Ejabberd básico

por Ibon Castilla Varela
última modificación: 12 de noviembre de 2005

Este documento explica la sintaxis del fichero de configuración de este servidor de mensajería instantánea, así como da algunos ejemplos para llegar a instalar un servidor de forma eficiente. El documento está en constante evolución. Cualquier sugerencia o comentario será siempre bienvenido. Puedes contactar conmigo en estri arroba euskalnet punto net o visitar mi sitio web http://sinanimodelucro.dysn.cx o la bitácora http://sinanimodelucro.dyns.cx/cgi-bin/blosxom

---

Empecemos dando una visión general de lo que vamos a hacer. Por un lado el paquete en Debian nos deja un serie de ficheros en el arbol de directorios, pero la configuración se va a reducir a dos cosas básicamente. La primera consistirá en comprender la sintaxis del fichero de configuración de Ejabberd que el paquete Debian nos deja en "`/etc/ejabberd/ejabberd.cfg`", y la segunda en el uso de varias herramientas para crear cuentas nuevas y realizar algunas tareas de administración en nuestro servidor de mensajeria. Vamos a ello.

Como deciamos, la instalación del paquete en Debian nos deja en /etc/ejabberd/ un fichero de configuración inicial, que tendremos que adaptar a nuestras necesidades. Recomiendo hacer una copia del fichero, para evitar fallos irreparables en la configuración. Esta regla no es propia de Debian o de Ejabberd, si no una buena norma aplicacle a todo buen administrador de sistemas.

Esta es su sintaxis: las lineas con "%" son comentarios. Cada línea del fichero es una tupla, donde la primera parte es la opción y la segunda parte es el valor que puede tomar. Al comienzo del fichero es posible que existan alguna o las tres lineas que resalto a continuación. Se trata de una serie de líneas que se saltan la configuración almacenada en la base de datos que utiliza el servicio Ejabberd si fuera necesario.

```
override_global.
override_local.
override_acls.
```

Una vez puesto en marcha el servicio, podemos comentarlas, de tal manera que la próxima vez que reiniciemos el servicio, las opciones se lean de la base de datos. No necesitaremos descomentarlas, hasta que no volvamos a necesitar reescribir los valores de la base de datos.

Nombre del servidor Jabber: HOSTS

La opción host define el o los dominios que sirve ejabberd. Esta es una parte aún un poco misteriosa para mi, ya que como veremos mas adelante, el nombre de host debe ser capaz de ser resuelto por DNS. Varios ejemplos:

```
{hosts, ["ejemplo.org", "ejemplo.com"]}.
{hosts, ["ejemplo.org"]}.
```

Idioma de los mensajes: LANGUAGE

La opción languaje está por defecto en inglés. Para comprobar que existe una traducción de los mensajes de Ejabberd para los usuarios debemos mirar en: /`usr/lib/erlang/lib/ejabberd-0.9.1/priv/msgs/.` Dentro del directorio se encuentran los ficheros relacionados con las traducciones actualmente soportadas. Si existe un fichero con nuestro idioma podemos cambiar el lenguaje en la configuración, por ejemplo:

```
{language, "es"}.
```

Control de acceso: ACL y ACCESS

Este es uno de los grandes apartados de Ejabberd. Su sintaxis general es como sigue:

```
{acl, <nombreacl>, {<tipoacl>, ...}}.
```

Donde "tipoacl" puede ser:

- all: significa todos los elementos. Por ejemplo: {acl, all, all}.
- user: **{user, <nombreusuario>}** significa el usuario con nombre de usuario "nombreusuario", en el primer host virtual.**{user, <nombreusuario>, <servidor>}** significa el usuario con nombre de usuario "nombreusuario" en el servidor "servidor". Por ejemplo:

```
{acl, admin, {user, "juan"}}.

{acl, admin, {user, "juan", "servidor.org"}}.
```

- user_regexp: **{user_regexp, <expresiónregular>}** significa el usuario que cumpla con la expresión regular "expresiónregular". **{user_regexp, <expresiónregular> <servidor>}** significa el usuario que cumpla con la expresión regular "expresiónregular" en el servidor "servidor". Por ejemplo:

```
{acl, tests, {user, "^test[0-9]*$"}}.

{acl, tests, {user, "^test", "servidor.org"}}.
```

- user_glob: **{user_glob, <glob>}** o **{user_glob, <glob>, <servidor>}** (ver node_glob)
- server: **{server, <servidor>}** significa el servidor con nombre "servidor". Por ejemplo:

```
{acl, jabberorg, {server, "servidor.org"}}.
```

- server_regexp: **{server_regexp, <expresiónregular>}** significa el servidor que cumpla con la expresión regular "expresiónregular". Por ejemplo:

```
{acl, icq, {server, "^icq\\."}}.
```

- server_glob: **{server_glob, <glob>}** (ver node_glob)
- node_regexp: **{node_regexp, <usuario_expresiónregular>, <servidor_expresiónregular>}** significa la combinación de usuario y servidor con sus respectivas expresiones regulares.
- node_glob: **{node_glob, <usuario_glob>, <servidor_glob>}** todos los "glob" son similares a las expresiones regulares, aunque se diferencian de estas últimas en que las expresiones "glob" siguen las reglas de la shell: "*", "?", o [...].

Una vez definida la ACL necesaria, se utilizan para la opción ACCESS: (pueden usarse “all” o “none”)

```
{access, <nombreacceso>, [{allow, <nombreacl>},
            {deny, <nombreacl>},
            ...
            ]}.
```

Por ejemplo:

```
{access, configure, [{allow, admin}]}.
{access, servicio, [{deny, chicos_malos},
            {allow, all}]}.
```

Limitación de conexiones: SHAPER

Con SHAPER se puede obligar a un tipo de tráfico en mi conexión: **{shaper, <shapername>, <kind>}.**

Actualmente solo existe un tipo de configuración implementada en Ejabberd: **{maxrate, <rate>}**

Donde “tasa” indica el número de bytes por segundo permitida de una conexión entrante. Por ejemplo:

```
{shaper, normal, {maxrate, 1000}}.
```

Sockets abiertos: LISTEN

Esta es otra de las grandes secciones de Ejabberd. En ella determinamos tres cosas: Número de puerto.

- Nombre del “módulo” que está escuchando en ese puerto.
- Opciones para ese “módulo”.

Actualmente estos son los módulos implementados:

- **ejabberd_c2s**

Este módulo sirve conexiones cliente servidor. Estas son las opciones de este módulo:

1. **{access, <regladeacceso>}** esta opción define el tipo de acceso de que disponen los usuarios al puerto C2S. El valor por defecto es “all”.
2. **{shaper, <reglaacceso>}** esta opción es similar a la anterior, pero utiliza “shapers” en vez de “allow” o “deny”. Por defecto esta a “none”.
3. **{ip, DirecciónIP}** esta opción especifica en que IP debe escuchar el módulo. Por ejemplo: `{ip, {192, 168, 1, 1}}.`
4. **inet6** esta opción hace compatible el socket con Ipv6.
5. **starttls** esta opción especifica que STARTTLS está disponible. Es necesaria la opción “certfile”.
6. **tls** esta opción especifica que la conexión va a pasar a ser segura nada mas

establecerse. Es necesaria la opción “certfile”.
7. **`ssl`** similar a la anterior. Se recomienda la opción “tls”.
8. **`{certfile, ruta}`** ruta hasta el certificado SSL.

- **`ejabberd_s2s_in`**

Este módulo sirve conexiones entrantes entre servidores.

- **`ejabberd_service`**

Este módulo sirve conexiones de los servicios Jabber. Estas son las opciones (las opciones `access`, `shaper`, `ip`, e `inet6` también son válidas):

1. **`{host, Hostname, [HostOptions]}`** esta opción define el nombre de host donde aparece el servicio, y las opciones correpondientes.
2. **`{hosts, [Hostnames], [HostOptions]}`** esta opción define el nombre de varios host donde aparece el servicio, y las opciones correspondientes.

- **`ejabberd_http`**

Este módulo sirve conexiones HTTP entrantes. Estas son sus opciones:

1. **`http_poll`** esta opción activa el JEP-0025 (http://www.jabber.org/jeps/jep-0025.html). Está disponible en `http://servidor:puerto/http-poll/`
2. **`web_admin`** esta opción activa la administración via interfaz web, disponible en `http://servidor:puerto/admin/.` Una vez accedido al sitio web se solicita un nombre de usuario y una contraseña que debe corresponderse con un usuario que este en la regla de acceso de “configure”. ¡MUY IMPORTANTE! (versión 0.9.)1: para acceder a la administración via web, el nombre de usuario es “usuario@host”, por ejemplo: prueba@servidor.servidor.org, de lo contrario puedes tirarte horas haciendo pruebas con el nombre de usuario sin el “arroba, nombre de host”, sin conseguir que funcione.

Un ejemplo de todo esto podría ser:

```
{acl, blocked, {user, “maloso”}}.
{access, c2s, [{deny, blocked},
{allow, all}]}.
{shaper, normal, {maxrate, 1000}}.
{access, c2s_shaper, [{none, admin},
{normal, all}]}.
{listen,
  [
    {5222, ejabberd_c2s,[{access, c2s}, {shaper, c2s_shaper}]},
    {5223, ejabberd_c2s,[{access, c2s}, ssl, {certfile, “/ruta/al/fichero/ssl.pem”}]},
    {5269, ejabberd_s2s_in,  []},
    {5280, ejabberd_http,    [http_poll, web_admin]},
    {5233, ejabberd_service, [{host, “aim.ejemplo.org”, [{password, “aimsecreto”}]}]},
    {5234, ejabberd_service, [{hosts, [”icq.ejemplo.org”, “sms.ejemplo.org”],
[{password, “jitsecreto”}]}]}
  ]
}.
```

Todo esto traducido significa:

- El servicio C2S se establece en el puerto 5222 y 5223 para SSL y se deniega para el

usuario “maloso”.

- El servicio S2S se establece en el puerto 5269.
- El servicio HTTP se establece en el puerto 5280 y poseemos acceso para administrar y http_poll.
- Todos los usuarios, excepto admin, tienen acceso limitado a 1000B/s.
- Los servicios Jabber están disponibles en los puertos especificados con las contraseñas especificados.

MODULES

Los módulos dan funcionalidad a Ejabberd para hacer multitud de cosas. Estos módulos no tienen nada que ver con los módulos de la sección inmediatamente anterior.

Ejemplo:

```
{modules,
 [
  {mod_register,  []},
  {mod_roster,    []},
  {mod_privacy,   []},
  {mod_configure, []},
  {mod_disco,     []},
  {mod_stats,     []},
  {mod_vcard,     []},
  {mod_offline,   []},
  {mod_announce,  [{access, announce}]},
  {mod_echo,      [{host, “echo.ejemplo.org”}]},
  {mod_private,   []},
  {mod_irc,       []},
  {mod_muc,       []},
  {mod_pubsub,    []},
  {mod_time,      [{iqdisc, no_queue}]},
  {mod_last,      []},
  {mod_version,   []}
 ]
}.
```

Para ver el mas detalles acerca de estos módulos consultar el apéndice A (al final de este documento).

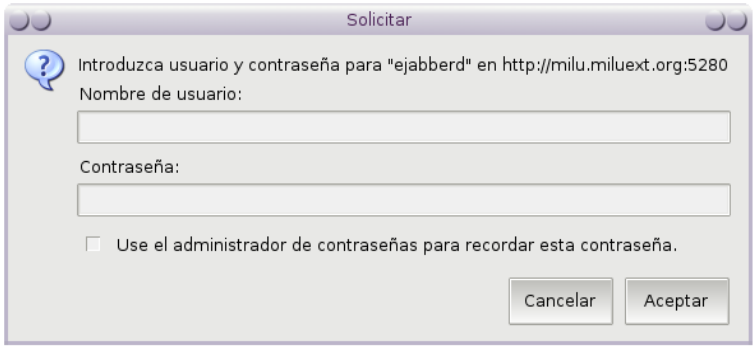
CONFIGURACIÓN ON-LINE Y MONITORIZACIÓN

Este es uno de los métodos de administración mas prácticos, aunque nos el primero que vamos a poder utilizar, al menos como punto de partida. En el fichero de administración debemos determinar un usuario administrador, y para poder acceder a esta herramienta de administración es necesario haberlo creado primero. Para resolver esta especie de bucle sin fin, tenemos una herramienta que veremos un poco mas adelante. Terminemos primero con esta otra. Para acceder a la configuración on-line debemos tener activo el servicio “ejabberd_http”, que veíamos en la sección anterior. Por ejemplo:

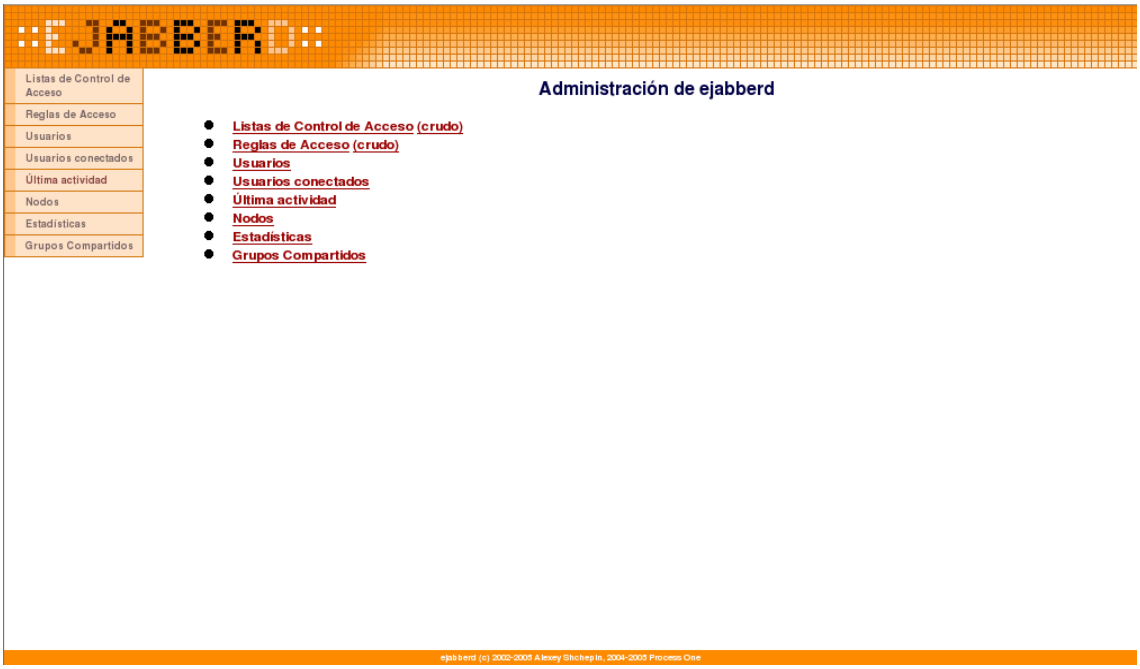
```
{host, "ejemplo.org"}.
{listen,
 [
  ...
  {5280, ejabberd_http, [web_admin]},
  ...
 ]
}.
```

Cuando este listo, debes introducir la siguiente dirección en el navegador:
http://ejemplo.org:5280/admin/

Y podrás ver una pantalla como esta:



En esta pantalla acuérdate de poner el nombre de usuario en la forma "usuario@host". Si introduces tus datos de forma correcta puedes ver una pantalla como esta:



Desde esta interfaz se pueden realizar multitud de tareas de administración como las que aparecen en los menus. Quizá las mas interesantes son la posibilidad de editar la configuración, modificar cuentas de usuario, etc.

La segunda herramienta que comentaba anteriormente es menos vistosa pero igual o mas potente. Se trata del comado EJABBERDCTL: Esta herramienta de línea de comando, nos permite interactuar con el servidor para ejecutar varias acciones:

```
status                      obtiene el esto del servidor ejabberd.
stop                              detiene el servidor ejabberd.
restart                           reinicia el servidor ejabberd.
reopen-log                        reabre el fichero de log.
register user server password     registra a un usuario en un servidor.
unregister user server            elimina a un usuario de un servidor.
backup file                       almacena la base de datos en un fichero.
restore file                restaura la base de datos desde un fichero.
install-fallback file             instala una base de datos desde un fichero.
dump file                         vuelca el contenido de la base de datos a un fichero
de texto.
load file                         restaura la base de datos de un fichero de texto.
import-file file                  importa la base de datos de usuario de un fichero
spool.
import-dir dir                    importa la base de datos de usuario de un directorio
de spool.
registered-users                  lista todos los usuarios registrados.
delete-expired-messages           elimina todos los mensajes de “fuera de linea”, que
han expirado, del servidor.
```

Es posible por ejemplo hacer copias de seguridad de la base de datos, gestionar cuentas de usuario,

reiniciar el servidor o pararlo, etc. En esta herramienta es donde aún no veo claro su funcionamiento en lo referente al nombre de host que comentaba con anterioridad: si el fichero de configuración dice:

```
{host, “servidor”}
```

y utilizamos la herramienta para dar de alta a un usuario, tal que así:

```
ejabberdctl register-user usuario servidor contraseña
```

Parece que no funciona. Debe ser porque el nombre debe poder ser resuelto por DNS, y no vale ni siquiera editar el fichero de hosts. De todas maneras ya investigaré para ver como funciona realmente.

Un par de ejemplos mas:

```
ejabberdctl restart
ejabberdctl --node ejabberd@host restart
```

Apéndice A

A.1 Opciones generales

Las opciones que aparecen a continuación son comunes a todos los módulos, por lo que se describen a continuación.

A.1.1 `iqdisc`

Varios módulos definen "conectores" para manejar consultas IQ de diferentes áreas a este servidor o usuario (por ej: a `ejemplo.org` o a `usuario@ejemplo.org`). Esta opción define la norma a seguir en estos casos. Los posibles valores son:

`no_queue`
: Todas las consultas de área con esta norma son procesadas inmediatamente. Esto significa que ningún otro paquete es procesado hasta que esto finalice. De ahí que esta norma no sea recomendable si el procesado de la consulta puede llevar bastante tiempo.

`one_queue`
: En este caso se crea un cola que almacena estas consultas y se procesa a la par que el resto de paquetes. Esta es la norma mas recomendable.

`parallel`
: En este caso todos los paquetes con esta norma generan un proceso Erlang separado, de tal manera qu son procesados en paralelo. Aunque la generación de procesos Erlang tiene un costo relativamente bajo, el trabajo habitual del servidor puede verse interrumpido, porque el emulador Erlang limita el número de procesos a 32000 (por defecto).

Ejemplo:

```
{modules,
 [
  ...
  {mod_time, [{iqdisc, no_queue}]},
  ...
 ]}.
```

A.1.2 `host`

Esta opción define de forma explícita el nombre de host para el módulo que actúa como servicio:

Ejemplo:

```
{modules,
 [
  ...
  {mod_echo, [{host, "echo.ejemplo.org"}]},
  ...
 ]}.
```

A.1.3 `hosts`

Esta opción define una lista de nombres de host, para el módulo que actúa como servicio

Ejemplo:

```
{modules,
 [
  ...
  {mod_echo, [{hosts, ["echo.ejemplo.org", "echo.ejemplo.com"]}]},
  ...
 ]}.
```

mod_announce

Este módulo añade soporte para los mensajes de anuncio de broadcast y para MOTD. Cuando el módulo se carga maneja los mensajes enviados a los JIDes que se comentan a continuación (suponiendo que el servidor tiene la dirección"ejemplo.org"):

ejemplo.org/announce/all
: El mensaje se envía a todos los usuarios registrados en `ejemplo.org`. Si el usuario está conectado y conectado a diversos recursos, únicamente el recurso con mayor prioridad recibirá el mensaje. Si el usuario no está conectado, el mensaje se almacenará en "fuera de línea" (si este está disponible).

ejemplo.org/announce/online
: El mensaje es enviado a todos los usuarios conectados a `ejemplo.org`. Si el usuario está conectado a varios recursos el mensaje se recibirá en todos ellos.

ejemplo.org/announce/all-hosts/online
: El mensaje se envía a todos los usuarios conectados a cualquier host virtual. Si el usuario está conectado a varios recursos el mensaje se recibirá en todos ellos.

ejemplo.org/announce/motd
: El mensaje se trata como MOTD (Mensaje del día) y se envía a los usuarios del servidor `ejemplo.org` nada mas hacer login. Además el mensaje es enviado a todos los usuarios conectados de forma similar a announce/online.

ejemplo.org/announce/motd/update
: El mensaje se trata como MOTD (Mensaje del dia) y es enviado a todos los usuarios de `ejemplo.org` en cuanto hagan login. El mensaje no se envia a los usuarios ya conectados.

ejemplo.org/announce/motd/delete
: Cualquier mensaje mensaje enviado a este JID elimina el mensaje del día.

Opciones:

access
: Especifica quién está autorizado a enviar mensajes. Por defecto es `none`.

Ejemplo:

```
% Only admins can send announcement messages:
{access, announce, [{allow, admin}]}.


{modules,
 [
  ...
  {mod_announce, [{access, announce}]},
  ...
 ]}.
```

Para postear un nuevo anuncio es suficiente con GAIM, por ejemplo, abrir una ventana de envio de mensaje, teclear la dirección a la que se envia (los JID's que se acaban de mencionar) y pulsar en

enviar. Eso si, quién lo haga tiene que tener permiso para hacerlo.

En breve, mas por llegar.