1 Instalacion de Servidor de Mensajeria en Debian

1.1 Sumario

2 Servidor XMPP/EJABBERD

2.1 El protocolo XMPP.

Extensible Messaging and Presence Protocol, más conocido como **XMPP** (Protocolo extensible de mensajería y comunicación de presencia) (anteriormente llamado Jabber1), es un protocolo abierto y extensible basado en XML, originalmente ideado para mensajería instantánea.

Con el protocolo XMPP queda establecida una plataforma para el intercambio de datos XML que puede ser usada en aplicaciones de mensajería instantánea. Las características en cuanto a adaptabilidad y sencillez del XML son heredadas de este modo por el protocolo XMPP.

A diferencia de los protocolos propietarios de intercambio de mensajes como ICQ, Y! y Windows Live Messenger, se encuentra documentado y se insta a utilizarlo en cualquier proyecto. Existen servidores y clientes libres que pueden ser usados sin coste alguno.

Este es el protocolo que seleccionó Google para su servicio de mensajería Google Talk, así como Facebook y Tuenti entre otras para su chat.

2.1.1 Ventajas.

Descentralización
: La arquitectura de las redes XMPP es similar a la del correo electrónico; cualquiera puede poner en marcha su propio servidor XMPP, sin que haya ningún servidor central.

Estándares abiertos
: La **Internet Engineering Task Force** ha formalizado el protocolo XMPP como una tecnología de mensajería instantánea estándar, y sus especificaciones han sido publicadas como los RFC 3920 y RFC 3921. El desarrollo de esta tecnología no está ligado a ninguna empresa en concreto y no requiere el pago de regalías.

Historia
: Las tecnologías XMPP llevan usándose desde 1998. Existen múltiples implementar|implementaciones de los estándares XMPP para clientes, servidores, componentes y bibliotecas, con el apoyo de importantes compañías como Sun Microsystems y Google.

Seguridad
: Los servidores XMPP pueden estar aislados de la red pública XMPP, y poseen robustos sistemas de seguridad (como **SASL** y **Transport Layer Security**|**TLS**). Para apoyar la utilización de los sistemas de cifrado, la XMPP Standards Foundation pone a disposición de los administradores de servidores XMPP Autoridad de certificación en xmpp.net ofreciendo Criptografía asimétrica|certificados digitales gratis.

Flexibilidad
: Se pueden hacer funcionalidades a medida sobre XMPP; para mantener la interoperabilidad, las extensiones más comunes son gestionadas por la XMPP Software Foundation.

2.1.2 Desventajas.

Escalabilidad
: XMPP también sufre el mismo problema de redundancia en los servicios de chatroom y de suscripción. Actualmente se está trabajando en su solución.

Sin datos binarios
: XMPP es codificado como un único y largo documento XML, lo que hace imposible entregar datos binarios sin modificar. De todas formas, las transferencias de archivos se han solucionado usando otros protocolos como HTTP. Si es inevitable, XMPP también puede realizar transferencias codificando todos los datos mediante base64.

2.1.3 Proceso de entrega de mensajes.

Supongamos que *julieta@santiago.com* desea chatear con *romeo@madrid.net*. Julieta y Romeo tienen sus respectivas cuentas en los servidores santiago.com y madrid.net. Cuando Julieta escribe y envía su mensaje, entra en acción la siguiente secuencia de eventos:

1. El cliente de Julieta envía su mensaje al servidor santiago.com.
2. - Si el servidor santiago.com no tiene acceso al servidor madrid.net, el mensaje es desechado.
3. Si el servidor santiago.com tiene acceso con madrid.net, abre una conexión con ese servidor.
4. El servidor madrid.net entregará el mensaje a Romeo:
5. - Si Romeo no está conectado, el mensaje es guardado para su posterior entrega.

2.1.4 Clientes de XMPP.

Algunos clientes que implementan el protocolo XMPP son:

- Pidgin (software): Uno de los clientes más usados en GNU/Linux, soporta otros protocolos como el de MSN, Yahoo.
- Google Talk: Implementación utilizada por Google en su sistema de Mensajería instantánea.
- Jabbear: Cliente de Mensajería instantánea incluido en el Sistema operativo Windows, Web.
- LJTalk: Cliente de Mensajería instantánea utilizado por el producto LiveJournal.
- Tkabber: Cliente con licencia Licencia pública general de GNU|GNU GPL escrito con Tcl/Tk que se puede utilizar bajo Windows y Linux.
- Champoo Chat: Cliente con licencia Licencia pública general de GNU|GNU GPL escrito en Java (lenguaje de programación), que provee una experiencia de chat tipo IRC, con soporte para canales, conferencias y chats privados. Posee ademas un cliente de Twitter muy simple, totalmente integrado con el sistema.

2.2 Instalación de Ejabberd.

Vamos a instalar el servidor de Jabber **ejabberd**.

```
# Instalamos el servidor Jabber.
 apt-get install ejabberd

# NOTA: cuando veáis en el fichero de configuración las siglas MUC significan (Multi User Chat).

# Para crear un usuario nuevo en el servidor localhost.
# ejabberdctl register (username) (server) (password)
ejabberdctl register veiga localhost xxxxxxxxxx

# Si queremos modificar la contraseña de un usuario.
# ejabberdctl change-password (username) (server) (newpassword)
ejabberdctl change-password veiga localhost xxxxxxxxxx

# El fichero de configuración se encuentra en:
nano /etc/ejabberd/ejabberd.cfg

# Si queremos que ese usuario tenga privilegios de administracion, tendremos que asignar
# ese usuario al ACL admin desde la sección ACCESS CONTROL LISTS:
{acl, admin, {user, "veiga", "localhost"}}.

# Si queremos más de un usuario en la ACL de admin, repetiremos la línea cambiando datos:
{acl, admin, {user, "martina", "pruebas.local"}}.

# Con ésto conseguimos tener dos usuarios en la ACL de administradores.

# Comprobar que sólo los miembros de la acl admin puedan acceder a la interfaz de configuración:
{access, configure, [{allow, admin}]}.
```

```
# Si nuestro servidor de ejabberd sólo va a funcionar en local podremos dejar la sección de
# hosts tal y como está, en cualquier otro caso si va a estar en un dominio es recomendable
# añadir el nombre del dominio que también va a ser gestionado por ejabberd.
# Hay que tener en cuenta que los usuarios creados son diferentes para cada dominio.
%% Hostname
{hosts, ["pruebas.local", "localhost"]}.

# Para el caso de que queramos integrarlo con LDAP, editamos el fichero de configuración
# para dar privilegios de administracion a una cuenta xxxx del LDAP:
{acl, admin, {user, "xxxx", "pruebas.local"}}.
{access, configure, [{allow, xxxx }]}.

# Una vez hechos los cambios reiniciamos el servicio:
service ejabberd restart
```

2.2.1 Puertos usados en Ejabberd.

El servidor Ejabberd utiliza los siguientes puertos:

- Para comunicaciones cliente-servidor: **5222**
- Para comunicaciones servidor-servidor: **5269**

- En la **sección de LISTENING PORTS** del fichero de configuración es dónde se configurarán todos los puertos, incluído el puerto de **configuración web del servidor ejabberd**, que viene configurado por defecto en el puerto **5280**.

```
{5280, ejabberd_http, [
                        %%{request_handlers,
                        %% [
                        %%  {["pub", "archive"], mod_http_fileserver}
                        %% ]},
                        %%captcha,
                        http_bind,
                        http_poll,
                        web_admin
                       ]}
```

2.2.2 Watchdog Admins en Ejabberd.

Si un proceso de ejabberd consume demasiada memoria, se puede configurar para que envíe una notificación instantánea a las cuentas que deseemos:

```
# Descomentaremos esta línea y configuraremos el nombre de la cuenta a la que enviar la notificación instantánea.
%%{watchdog_admins, ["bob@example.com"]}.
```

2.2.3 Sección MODULES en Ejabberd.

- En la **sección de MODULES** se configurarán todos los módulos que el servidor ejabberd tiene disponibles.

Por ejemplo podríamos configurar el mensaje de bienvenida del servidor Jabber en **mod_register** y también podríamos configurar un usuario que será notificado de los nuevos registros de forma automática (**registration watcher**):

```
{mod_register, [
                %%
                %% After successful registration, the user receives
                %% a message with this subject and body.
                %%
                {welcome_message, {"Welcome!",
                                   "Welcome to a Jabber service powered by Debian. "
                                   "For information about Jabber visit "
                                   "http://www.jabber.org"}},
                %% Replace it with 'none' if you dont want to send such message:
                %%{welcome_message, none},

                %%
```

```
    %% When a user registers, send a notification to
    %% these Jabber accounts.
    %%
    %%{registration_watchers, ["admin1@example.org"]},

    {access, register}
   ]},
```

- Para visualizar los logs de error de ejabberd consultar el fichero: /var/log/ejabberd/ejabberd.log

```
tail /var/log/ejabberd/ejabberd.log
```

- Para que la comunicación entre servidores y cliente-servidor funcione correctamente es muy importante tener una buena **configuración en el DNS**:

```
#Tendremos que editar el servidor de DNS, para agregar unos registros de servicio (SRV) para Jabber:
# Creamos un registro de tipo A o cname para nuestro servidor de mensajeria.
# Creamos un CNAME para el servidor conference en nuestro dominio:

mensajeria.pruebas.local.       IN      A       192.168.1.25
conference.pruebas.local.       IN      CNAME   mensajeria

# Registros a crear en el DNS:
# 5 es la prioridad.
# 0 es el peso.
# 5269 es el puerto.

_jabber._tcp.pruebas.local.     IN      SRV     5 0 5269 mensajeria
_xmpp-server._tcp.pruebas.local.        IN      SRV     5 0 5269 mensajeria
_xmpp-client._tcp.pruebas.local.        IN      SRV     5 0 5222 mensajeria

# Si al aplicar cambios nos da algún error de rndc connection refused ejecutar:
rndc reload

# Una vez agregados los registros y aplicados cambios se puede comprobar su funcionamiento con dig:
dig _jabber._tcp.pruebas.local srv
dig _xmpp-client._tcp.pruebas.local srv
dig _xmpp-server._tcp.pruebas.local srv
```

2.2.4 Activación de comunicación segura SSL/TLS.

Por defecto cuando instalamos ejabberd ya crea un certificado y ya escuchando en el puerto 5222 para establecer una conexión segura con los clientes.

Si quisiéramos crear otro certificado diferente podríamos hacerlo con las siguientes instrucciones:

```
cd /etc/ejabberd

openssl req -new -x509 -newkey rsa:1024 -days 3650 -keyout privkey.pem -out servidor.pem
openssl rsa -in privkey.pem -out privkey.pem
cat privkey.pem >> servidor.pem
rm privkey.pem
```

Ahora editamos el fichero de configuración /etc/ejabberd/ejabberd.cfg y especificamos la localización de nuestro certificado servidor.pem:

```
nano /etc/ejabberd/ejabberd.cfg

{listen, [{5222, ejabberd_c2s,     [{access, c2s}, {shaper, c2s_shaper},
                                    starttls, {certfile, "/etc/ejabberd/servidor.pem"}]},

%%% El uso del puerto 5223 está en desuso.
%%% Esta opción la activaríamos para dar soporte SSL antiguo a clientes.
          {5223, ejabberd_c2s,     [{access, c2s}, {shaper, c2s_shaper},
                                    tls, {certfile, "/etc/ejabberd/servidor.pem"}]},
          ...
]}.
```

```
{s2s_use_starttls, true}.
{s2s_certfile, "/etc/ejabberd/servidor.pem"}.
```

2.2.5 Configuración de Ejabber + LDAP Active Directory AD.

- Aquí se muestra una configuración de ejemplo para que Ejabberd se valide con un AD de Windows:

```
%% Admin user
{acl, admin, {user, "xxxx ", "sanclemente.local"}}.
{access, configure, [{allow, xxxx }]}.

%% Hostname
{hosts, ["sanclemente.local"]}.

%% {auth_method, internal}.
{auth_method, ldap}.
{ldap_servers, ["10.0.4.1"]}.
{ldap_encrypt, none}.
{ldap_port, 389}.
{ldap_rootdn, "CN=ldap,OU=Especiais,OU=SC-Usuarios,DC=sanclemente,DC=local"}.
{ldap_password, "xxxxxxxxxxx"}.
{ldap_base, "OU=Profes,OU=SC-Usuarios,DC=sanclemente,DC=local"}.
{ldap_uids, [{"sAMAccountName", "%u"}]}.
```

2.2.6 Registro y bloqueo de usuarios.

```
# Hemos visto como se podían registrar usuarios desde la línea de comandos.
ejabberdctl register (username) (server) (password)

# Por ejemplo:
ejabberdctl register veiga localhost xxxxxxxxxx

# Otra forma posible es registrar los usuarios via web, a través de la dirección de gestión:
http://IP:5280/admin

# Pero existe otro modo de registro automático de usuarios.
# Los clientes se podrán registrar automáticamente con el usuario que ellos elijan y con la
# contraseña que ellos quieran. Para permitir estos registros automáticos,
# tendremos que modificar la opción siguiente en la sección de "in-band registration":
{access, register, [{deny, all}]}.

#por la opción
{access, register, [{allow, all}]}.

# Acordaros de reiniciar el servicio, una vez hechas las modificaciones:
service ejabberd restart

# Si deseamos bloquear un usuario lo haremos desde la sección ACCESS CONTROL LISTS.
# Descomentaremos la línea siguiente y añadiremos el usuario que queremos bloquear:
%%{acl, blocked, {user, "baduser", "example.org"}}.
```

2.2.7 Creación de Grupos (Shared Roster) en Ejabberd.

Por ejemplo si en tu empresa sois unos pocos trabajadores, y te gustaría que ciertos grupos de usuarios se tuvieran entre sí como contactos, sin tener que añadirse entre ellos uno a uno. Lo que necesitarías configurar es Shared Roster Groups: Ver Capturas de Shared Roster Groups

Para activar los shared roster groups tienes que editar el fichero de configuración:

```
nano /etc/ejabberd/ejabberd.cfg

# Buscamos la siguiente línea:
%%{mod_shared_roster,[]},

# y la descomentamos:
{mod_shared_roster,[]},
```

Luego desde la página web crearemos los grupos y añadiremos los usuarios que deseemos. Consulta el siguiente enlace para ver algunos ejemplos: Ver Capturas de Shared Roster Groups

Tendremos que cubrir 4 secciones en el formulario de Shared Roster Groups:

- Nombre: (nombre del grupo que se mostrará en el pidgin)
- Descripcion: (información extra del grupo)
- Miembros: (quienes pertenecen al grupo).
- Mostrar Grupos: (qué grupos queremos que vea este grupo)

Y si quieres que todos los usuarios de tu servidor puedan ver a todos los usuarios de tu servidor, puedes poner como miembro del grupo: **@all@**

Recomendable visitar la siguiente dirección para ver la configuración de grupos: All Users in a Shared Roster Group.

2.3 Uso de Ejabberd.

- Para administrar el servidor de mensajería lo podemos hacer vía web a través de la página:

http://IP_Servidor:5280/admin

Accederemos con el usuario administrador que hemos creado y su contraseña.

```
Por ejemplo:
veiga@localhost
xxxxxxxxxx
```

- Para parar o arrancar el servidor o comprobar su status:

```
service ejabberd start | stop | restart | live
```

- El servidor de ejabberd también se puede controlar desde la línea de comandos con **ejabberdctl**:

Para más información sobre las opciones de ejabberdctl visitar: http://www.digipedia.pl/man/doc/view/ejabberdctl.8/

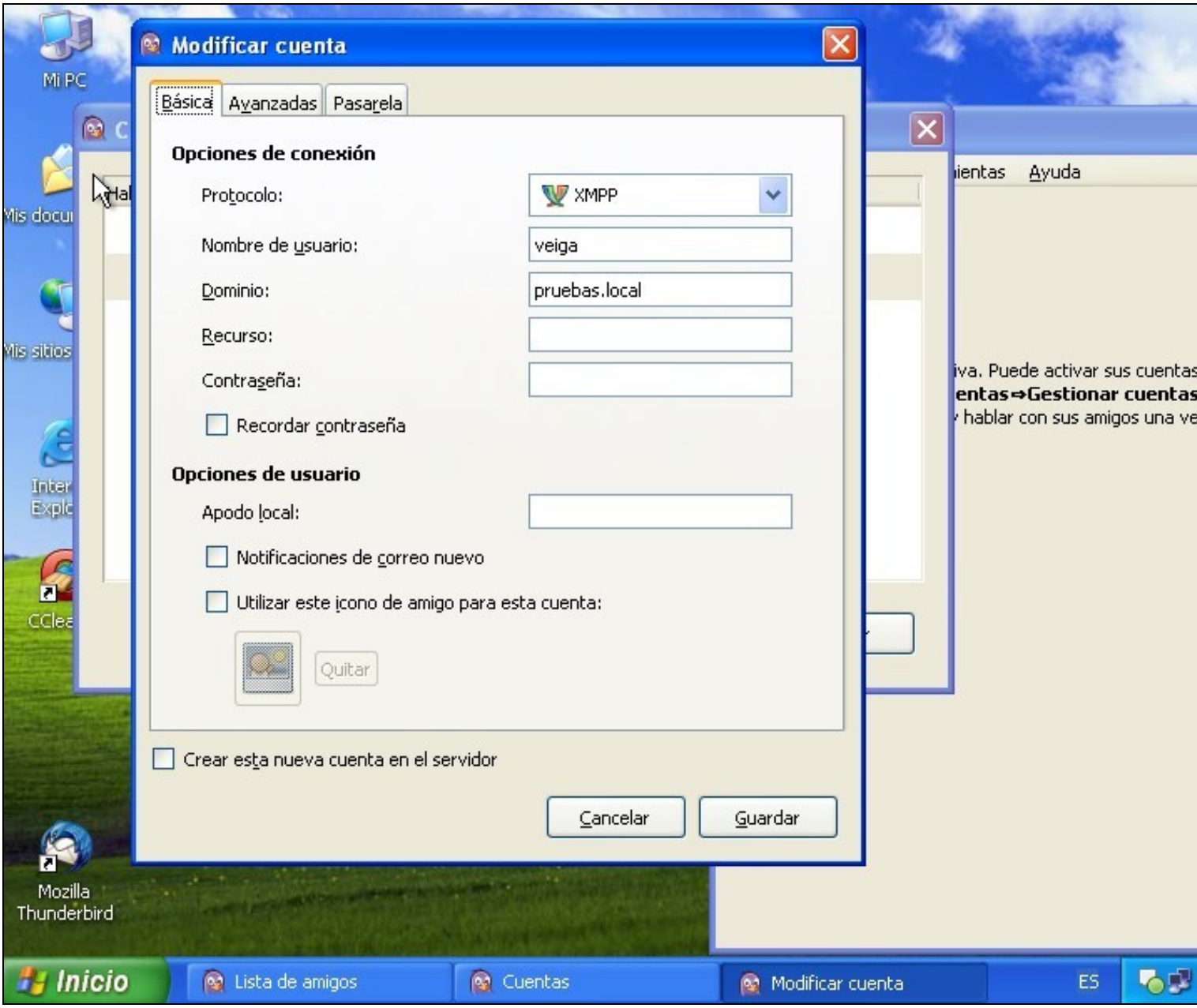
```
# Prueba de envío de mensajes desde la línea de comandos a otros usuarios:
ejabberdctl send_message_chat marga@pruebas.local  adolfo@pruebas.local  "Hola colegas !!"
```

2.4 Instalación de Cliente XMPP.

En Windows podemos instalarnos el cliente PIDGIN: http://www.pidgin.im/

Para crear nuevas cuentas lo haremos desde la sección:

Cuentas --> Gestionar Cuentas --> Añadir...



--Rafael Veiga 20:40 11 mar 2012 (GMT)