Laboratorio de Sistemas Operativos y Redes

Trabajo Práctico

Docente

- José Luis Di Biase

Alumno

- Ramirez Brandan, Brian Leonel
- Kevin Paz

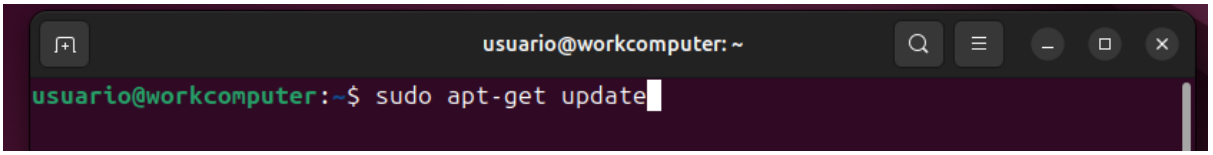
Introducción

ejabberd es un servidor de mensajería instantánea basado en el protocolo **XMPP** (Extensible Messaging and Presence Protocol), ampliamente utilizado para comunicaciones en tiempo real. Desarrollado con una arquitectura distribuida, altamente escalable y con soporte para múltiples plataformas, **ejabberd** es una solución ideal para implementar sistemas de chat seguros y robustos en entornos empresariales, educativos y personales.

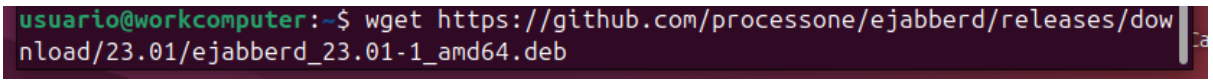
El objetivo principal de este trabajo práctico es proporcionar una guía práctica para instalar y configurar **ejabberd,** Cconectándolo con clientes de mensajería como **Pidgin** y **Gajim**. Además, se abordará la habilitación y configuración del módulo **mod_otr,** que permite el cifrado de mensajes punto a punto, mejorando la privacidad y seguridad de las comunicaciones.

Instalación de ejabberd

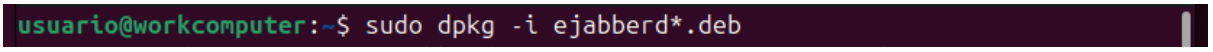
Lo primero será utilizar el comando update para actualizar los paquetes y evitar futuros conflictos.



Descargaremos ejabberd desde un repositorio porque la instalación por medio del comando sudo apt install presenta problemas al iniciar el servicio con los clientes.



Instalaremos el paquete descargado.



Confirmamos que el servicio de ejabberd está corriendo con el comando

systemctl status .ejabberd.service

```
usuario@workcomputer:~$ systemctl status ejabberd.service
● ejabberd.service - XMPP Server
     Loaded: loaded (/usr/lib/systemd/system/ejabberd.service; enabled; preset:>
     Active: active (running) since Wed 2024-12-04 22:17:46 UTC; 1min 11s ago
   Main PID: 4831 (ejabberdctl)
      Tasks: 25 (limit: 5989)
     Memory: 64.6M (peak: 87.5M)
        CPU: 2.662s
     CGroup: /system.slice/ejabberd.service
             ├─4831 /bin/sh /opt/ejabberd-23.01/bin/ejabberdctl foreground
             ├─4843 /opt/ejabberd-23.01/erts-12.3.2.5/bin/beam.smp -K true -P 2>
             ├─4860 erl_child_setup 65536
             ├─4875 /opt/ejabberd-23.01/lib/eimp-1.0.22/priv/bin/eimp
             ├─4876 /opt/ejabberd-23.01/lib/epam-1.0.12/priv/bin/epam
             ├─4877 inet_gethost 4
             ├─4878 inet_gethost 4
             └─4879 /opt/ejabberd-23.01/lib/os_mon-2.7.1/priv/bin/memsup
```

Vamos a editar el archivo de configuración en la siguiente ruta.

```
usuario@workcomputer:~$ sudo nano /opt/ejabberd/conf/ejabberd.yml
```

Añadimos el host workcomputer.local dentro de la configuración .

```
hosts:
  - workcomputer.local
```

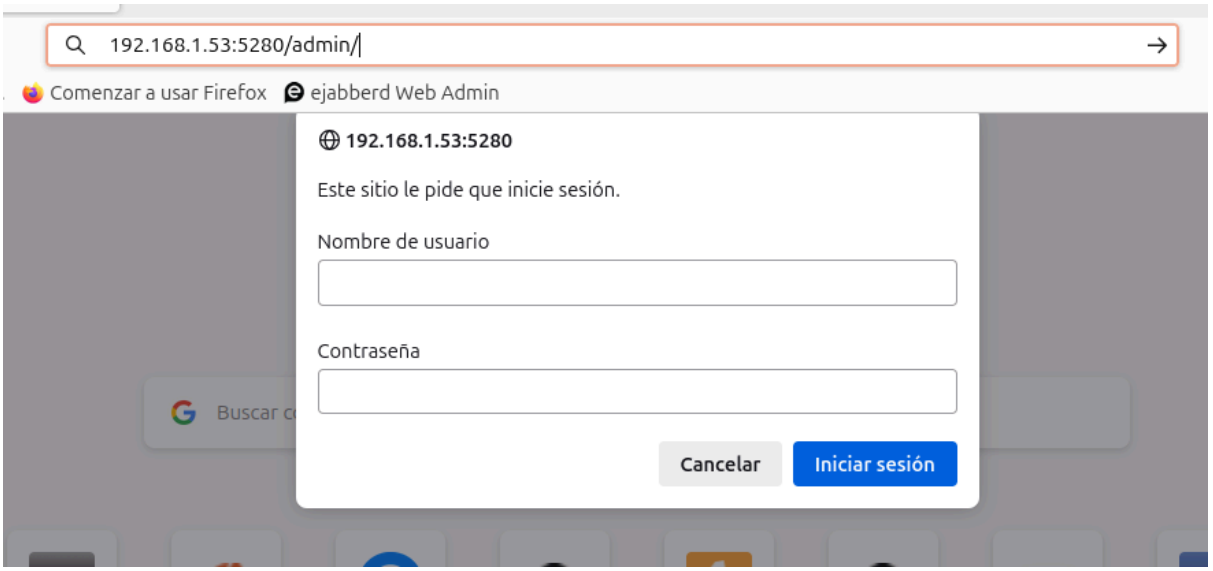
Cada vez que se realice algún cambio en la configuración del servicio es recomendable reiniciarlo.

```
usuario@workcomputer:~$ sudo systemctl restart ejabberd.service
```

Revisamos cuál es la ip del server por medio del terminal.

```
usuario@workcomputer:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gr
oup default qlen 1000
    link/ether 08:00:27:3d:b5:7d brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.53/24 brd 192.168.1.255 scope global dynamic noprefixroute en
p0s3
       valid_lft 27927sec preferred_lft 27927sec
    inet6 fe80::a00:27ff:fe3d:b57d/64 scope link
       valid_lft forever preferred_lft forever
```

En un navegador colocaremos la ip del server seguido del puerto 5280 y un /admin/ para acceder a la interfaz.



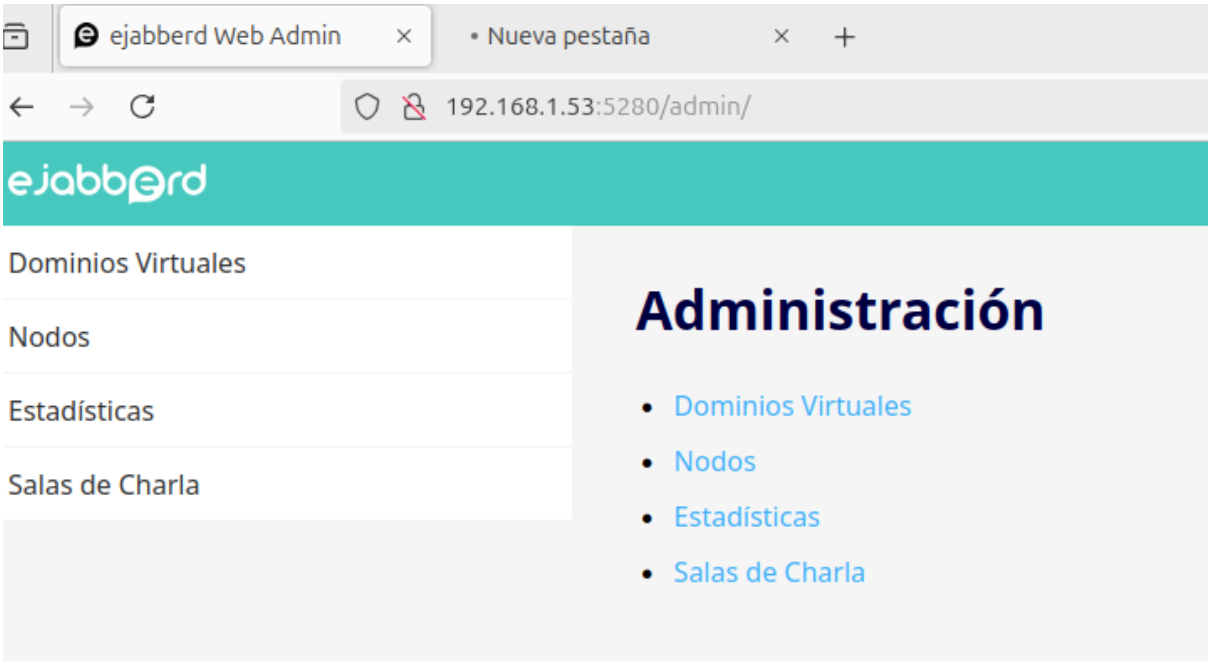
Tenemos que crear un usuario el cual llamaremos admin para esta prueba.

```
usuario@workcomputer:~$ sudo /opt/ejabberd-23.01/bin/ejabberdctl register admin
workcomputer.local admin1
User admin@workcomputer.local successfully registered
usuario@workcomputer:~$
```

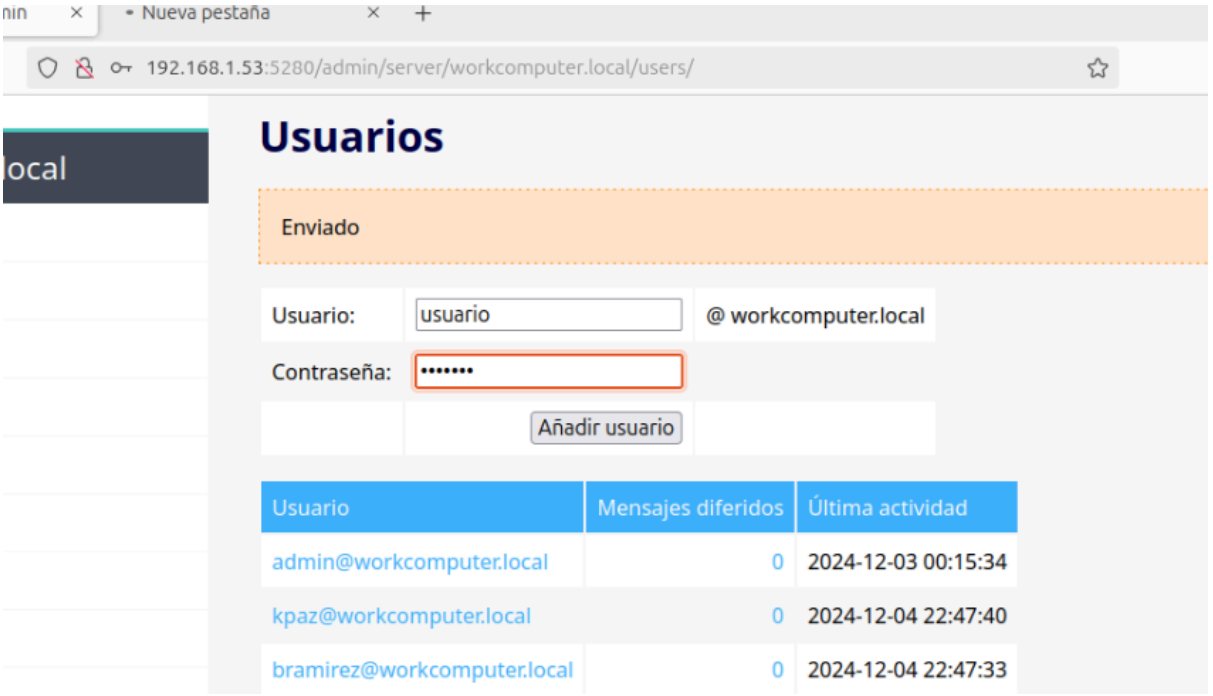
Nuevamente dentro del archivo de configuraciones donde antes añadimos el host ahora tenemos que crear el grupo admin y añadir los usuarios que tendrán privilegios de administrador, es importante una correcta indentación dentro del archivo.

```
acl:
  admin:
    user:
      - admin@workcomputer.local
  local:
    user_regexp: ""
  loopback:
    ip:
      - 127.0.0.0/8
      - ::1/128
```

Ahora con el usuario con privilegios de administrador podemos ver todas las opciones en la interfaz web.



Procedemos a crear los dos usuarios adicionales que utilizaran los clientes Pidgin y Gajim para enviar y recibir los mensajes.



Como nota adicional dentro del archivo de configuración podemos crear una sección para bloquear usuarios del servicio si así lo requerimos, no la

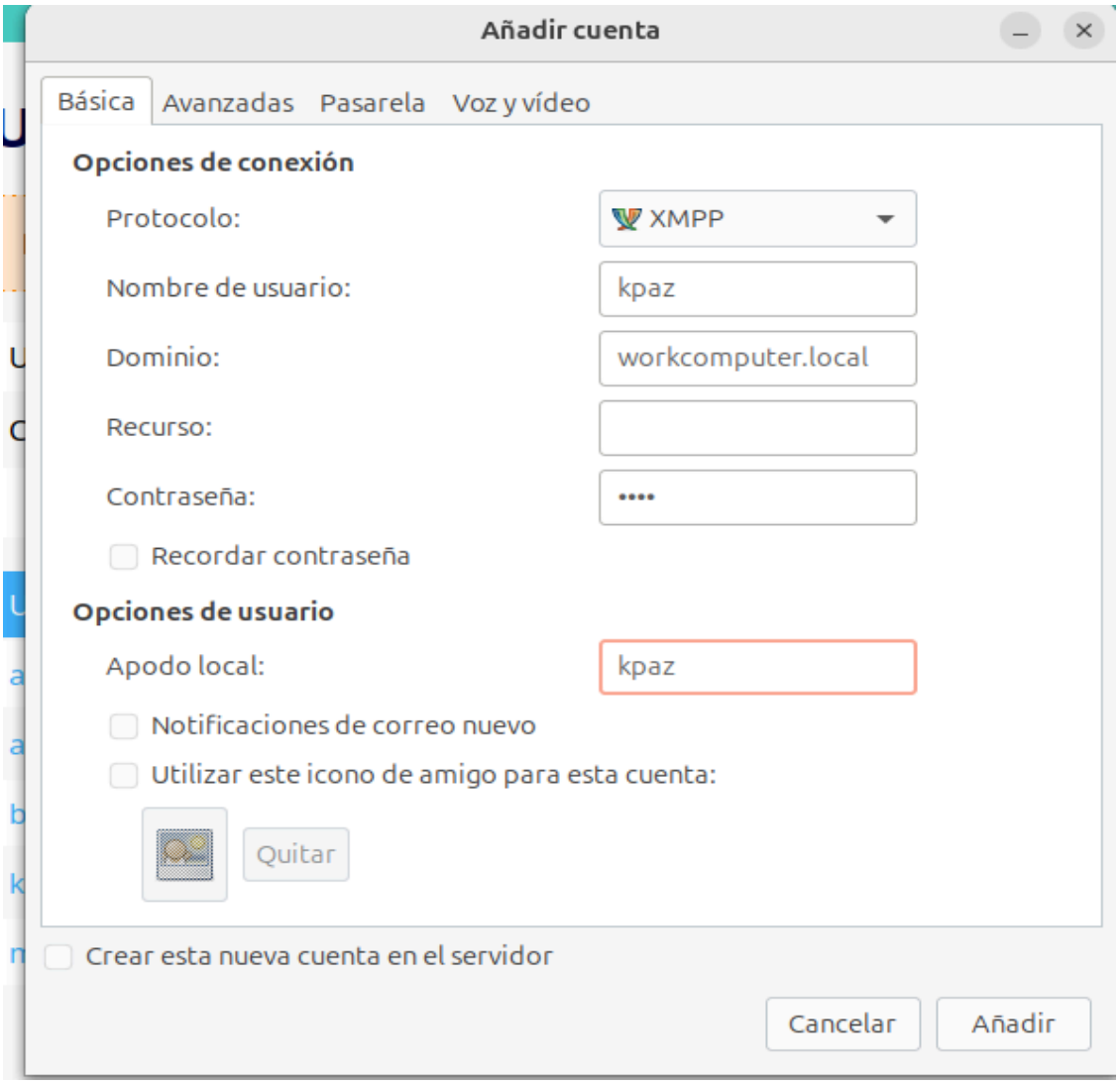
utilizaremos en este caso.

```
acl:
  admin:
    user:
      - admin@workcomputer.local
  local:
    user_regexp: ""
  loopback:
    ip:
      - 127.0.0.0/8
      - ::1/128
  blocked:
    user:
      - usuarioejemplobloq@workcomputer.local
```

Ahora podemos instalar los clientes Pidgin y Gajim en el terminal con los comandos apt install respectivamente.

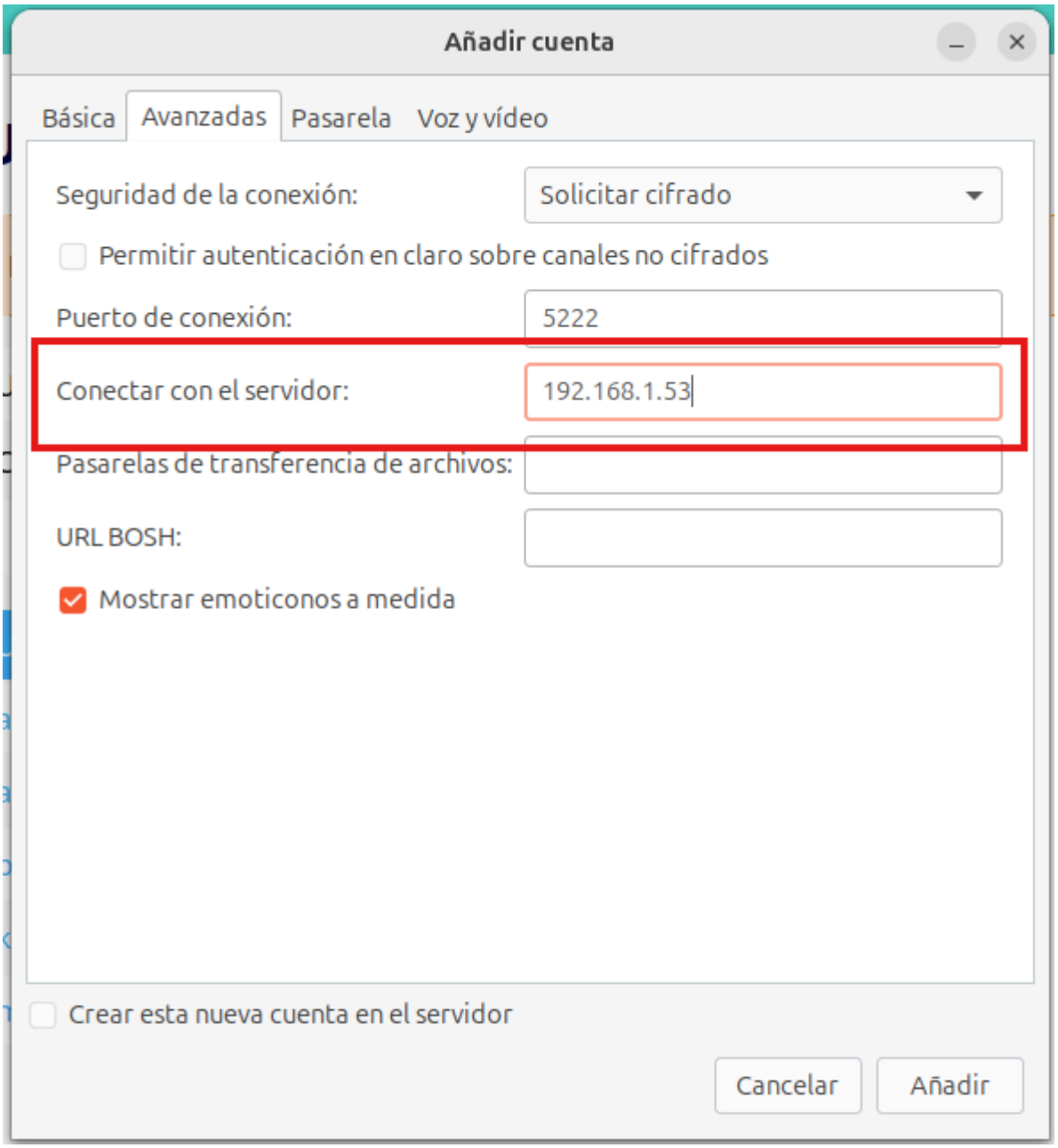
```
usuario@workcomputer:~$ sudo apt install pidgin
usuario@workcomputer:~$ sudo apt install gajim
```

Dentro del cliente Pidgin colocamos el usuario de la cuenta (kpaz) junto con el dominio workcomputer.local para este caso



En la configuración avanzada podemos colocar la ip del servidor la cual sabemos porque lo consultamos previamente, si tenemos la opción de configurar el host para nuestro local resolver podemos poner

workcomputer.local en vez de una dirección IP.



Si tenemos la posibilidad de acceder a los host dentro de la máquina donde tenemos los clientes haremos nano /etc/hosts para modificar el archivo.
Si ese es el caso procedemos con lo siguiente, si no podemos, se coloca la IP como se mencionó con anterioridad.

```
usuario@workcomputer:~$ sudo nano /etc/hosts
```

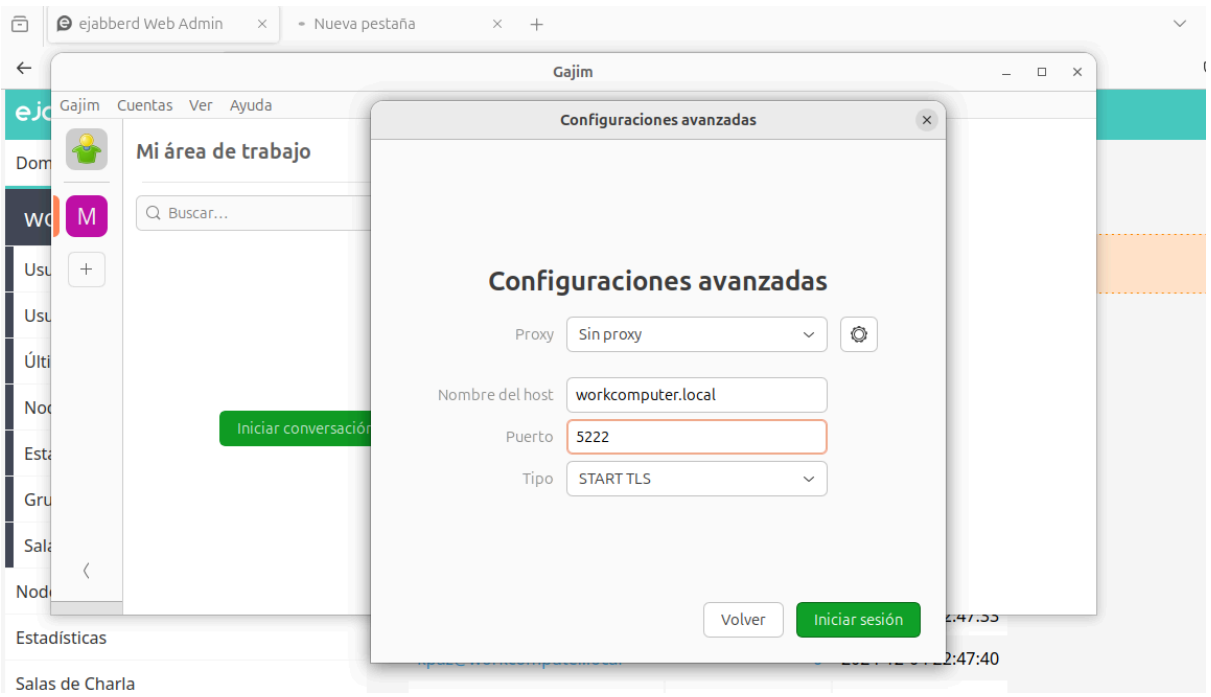
En la última línea tendremos que añadir la ip del server con su dominio.

```
# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
192.168.1.53 workcomputer.local
```

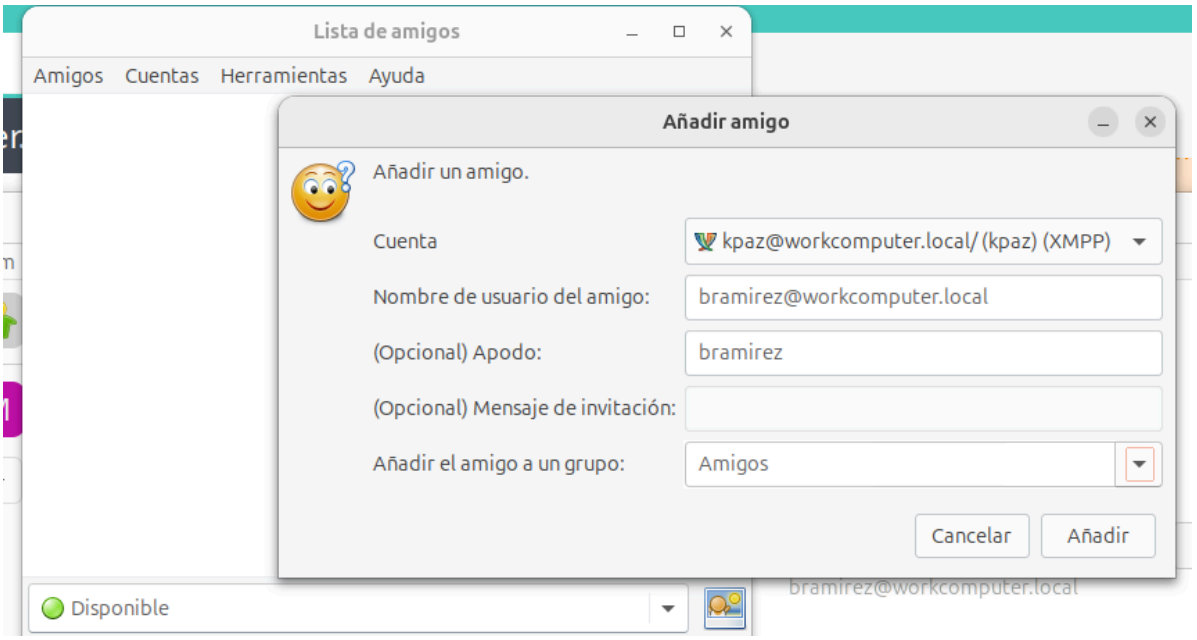
Usamos el comando ping al dominio para verificar que esté recibiendo los paquetes.

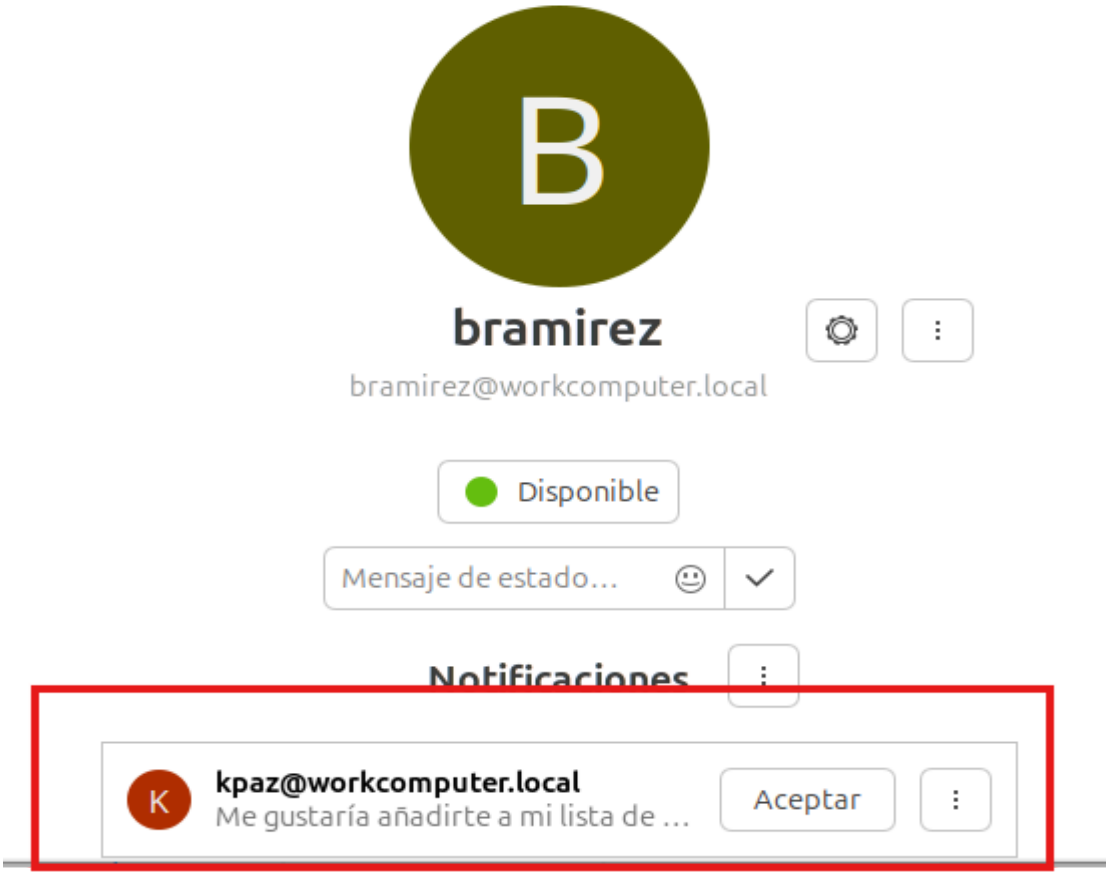
```
usuario@workcomputer:~$ sudo nano /etc/hosts
usuario@workcomputer:~$ ping workcomputer.local
PING workcomputer.local (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.014 ms
64 bytes from localhost (127.0.0.1): icmp_seq=2 ttl=64 time=0.016 ms
64 bytes from localhost (127.0.0.1): icmp_seq=3 ttl=64 time=0.021 ms
64 bytes from localhost (127.0.0.1): icmp_seq=4 ttl=64 time=0.017 ms
64 bytes from localhost (127.0.0.1): icmp_seq=5 ttl=64 time=0.019 ms
64 bytes from localhost (127.0.0.1): icmp_seq=6 ttl=64 time=0.020 ms
64 bytes from localhost (127.0.0.1): icmp_seq=7 ttl=64 time=0.019 ms
```

Dentro del otro cliente (Gajim) usaremos la otra cuenta que creamos como administradores (bramirez@workcomputer) siguiendo los mismos pasos que con la cuenta anterior pero para este cliente.

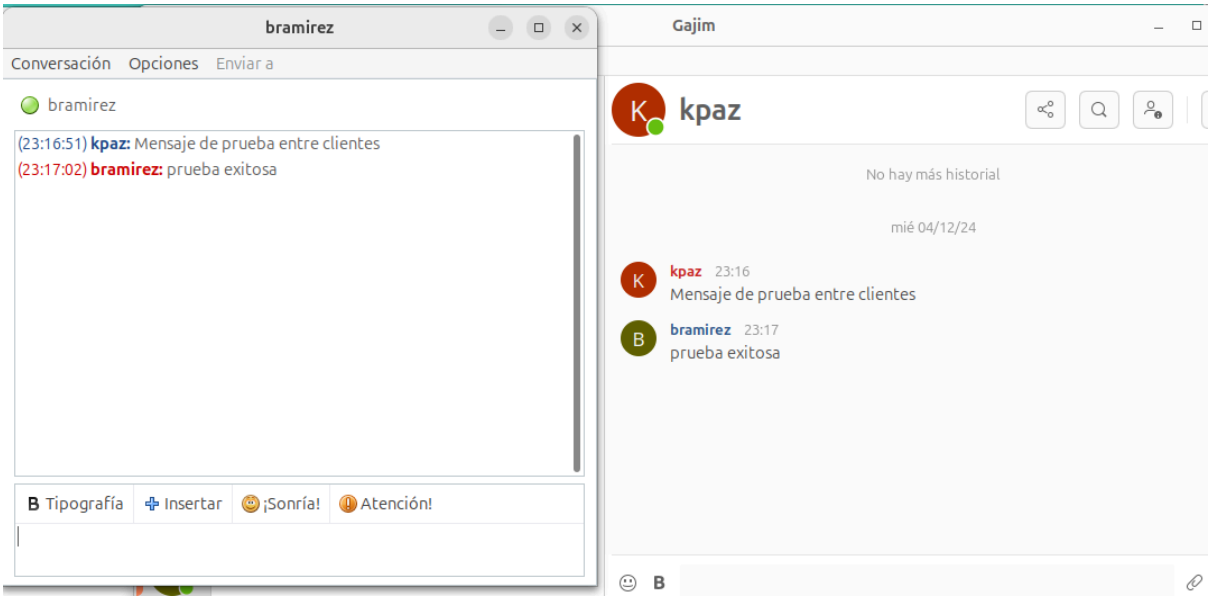


Regresamos al cliente Pidgin y añadiremos como amigo a la cuenta del cliente Gajim, kpaz@workcomputer.local le envía una solicitud a bramirez@workcomputer.local y este la verá reflejada en su cliente.





Una vez agregados como amigos los dos usuarios podrán establecer comunicación en tiempo real independientemente del cliente que estén utilizando.



Complicaciones encontradas

El módulo **mod_otr** utilizado para cifrado fue descontinuado por la comunidad y eliminado de los repositorios oficiales por lo que no encontramos una versión compatible para instalar las dependencias necesarias.

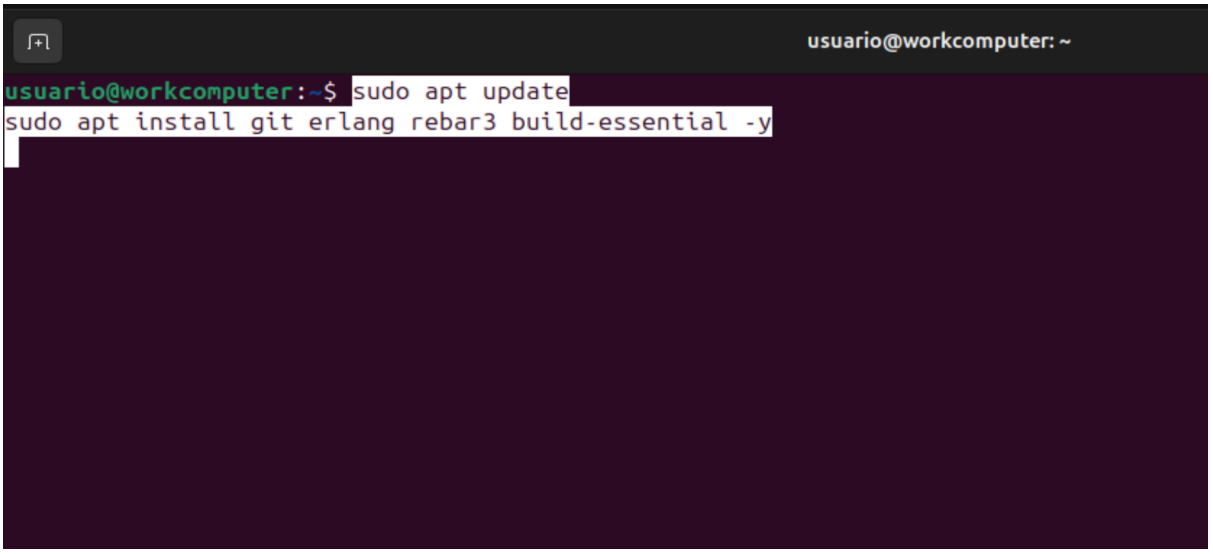
La instalación de ejabberd mediante el comando ‘sudo apt install ejabberd’ no lograba levantar el servicio correctamente generando problemas con el registro de los usuarios, por lo que se instaló mediante un repositorio la versión 23.01 del servicio.

Las pocas aplicaciones móviles disponibles presentaban problemas en la configuración de la cuenta, estaban dadas de baja de las páginas oficiales o eran de pago, lo que limitó la experiencia de prueba para usuarios móviles.

Instalar **mod_otr** para conexiones seguras

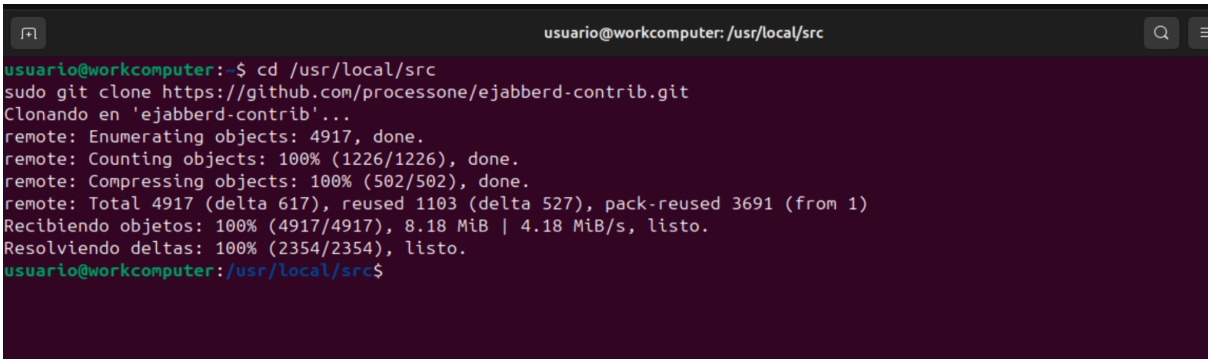
Instalar dependencias necesarias:

Son necesarias para poder compilar el módulo



- **Git**: Para clonar el repositorio de módulos contribuidos.
- **Erlang**: Ejabberd está basado en Erlang.
- **Rebar3**: Herramienta de construcción para proyectos Erlang.
- **Build-essential**: Para compilar módulos.

Clonar repositorio del módulo



Compilar módulo

Para este punto, al intentar compilar el módulo hemos notado que el directorio de **mod_otr** no se encontraba dentro del repositorio. Luego de una investigación averiguamos que OTR actualmente se encuentra en desuso.
Actualmente en lugar de OTR se usa un nuevo módulo llamado OMEMO, el cual ya viene instalado por defecto.
Solamente es necesario un cliente que lo soporte. Por ejemplo, Gajim soporta conexiones seguras con OMEMO, solamente hay que configurarlo en la ventana de chat.