UBINODES

GUIDE:

MATRIX ENCRYPTED CHAT SERVER

Guide: Matrix-encrypted chat servers

Matrix is a decentralized real-time communication protocol that operates as an open standard. It is implemented through distributed home servers across the internet, ensuring there is no single point of control or failure.

Contents of this article.

1. Matrix server installation guide.

Matrix operates as an open standard communication protocol for decentralized real-time communication. It is executed through distributed home servers across the internet, ensuring there is no single point of control or failure. Matrix provides a RESTful HTTP API for creating and managing distributed chat servers.

This includes tasks like sending and receiving messages, inviting and managing chat room members, handling user accounts, and offering advanced features such as VoIP and video calls. Matrix ensures secure synchronization between globally distributed home servers.

Synapse, developed by the Matrix team, serves as the implementation of the Matrix home server. Within the Matrix ecosystem, a network of federated home servers operates globally. Users connect to a chat client that interfaces with their home server, which in turn connects to the broader Matrix network. Each home server stores the chat history and login information for its users.

The next section will walk you through installing your Matrix reference server and connecting your initial user(s).

There are two basic things you need to run your private Matrix service:
• Domain name (e.g. ubinodes.org).
• A virtual server running Debian 8 on a cloud service (AWS, DigitalOcean, Vultr, etc.) or a physical server.
• Basic knowledge of the Linux CLI.

For this guide, we will use **ubinodes.org.**

2. DNS settings.

First, you have to register a domain name and fire up your DNS admin panel. You need to create a DNS record like this:

ubinodes.org 300 IN A 1.2.3.4

3. Installing Synapse.

After completing the previous step, the following guide will help you set up Synapse, which is Matrix's reference home server implementation.

3.1 Prepare your server.

• To begin, launch a virtual machine running **Debian 8** on your preferred cloud provider and SSH into the host. The following instructions assume that you are root on the server.
•Since the Matrix/Synapse package resides in a non-standard repository, we need to add this repository to our machine's package repository:

echo 'deb http://ftp.debian.org/debian jessie-backports main' >> /etc/apt/sources.list

• And then we need to make sure that Debian knows that the repo is there:

apt-get update && apt-get dist-upgrade -y

•Next, install a few essential packages that will be useful later. Since our VMs are basic by default, run the following command:

apt-get install -y apt-transport-https lsof curl python python-pip

apt-get install -y certbot -t jessie-backports

• At this point, we need to add another software repository. Create /etc/apt/sources.list.d/matrix.list and open this up in your preferred text editor.

• Inside /etc/apt/sources.list.d/matrix.list, add the following two lines:

deb https://matrix.org/packages/debian/ jessie main

deb-src https://matrix.org/packages/debian/ jessie main

3.2 Installing Synapse.

• With those packages installed, it's time to proceed with installing Matrix. Run the following command:

curl https://matrix.org/packages/debian/repo-key.asc | apt-key add -

apt-get update

apt-get install matrix-synapse -y

•With those packages installed, it's time to proceed with installing Matrix. Run the following command:
• If there's an issue with python-cffi, such as a package conflict error, it might cause the matrix-synapse installation to fail at this point.
• Simply run this command to install python-cffi from backports:

apt install python-cffi/jessie-backports

• Once the backported package is installed, try installing Synapse again:

apt-get install matrix-synapse -y

• You will be asked to provide a host name for your server, which in this tutorial we used myserver.example.com

4. Adding encryption support.

• Synapse should expose the Matrix service over SSL, so we need to request for a new certificate. You may reuse your existing SSL certificate if you already have one. For myserver.example.com. Otherwise, you can get a new one from Let's Encrypt.
• The next step is to use certbot to generate a Let's Encrypt certificate.

certbot certonly

• Choose the **"spin up a temporary web server"** option.
• The certificate is valid for three months. To configure auto-renewal, we need to add certbot to the system crontab file:

crontab -e

• Insert the following line:

@daily certbot renew --quiet --post-hook "systemctl reload nginx"

5. Configuring nginx.

• To make this HTTPS-ready, we need to configure a reverse proxy. We will use nginx for this, so install it:

apt-get install nginx -y

• Then add the following configuration to /etc/nginx/conf.d/matrix.conf:

```
server {
listen 443 ssl;
server_name love4aviation.fr;

ssl_certificate /etc/letsencrypt/live/love4aviation.fr/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/love4aviation.fr/privkey.pem;
ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
ssl_ciphers HIGH:!aNULL:!MD5;
```

```
location /_matrix {
proxy_pass http://localhost:8008;
proxy_set_header X-Forwarded-For $remote_addr;
}
}
```

• Make sure you replace ubinodes.org with the relevant server name.
• Once that's saved, restart nginx by running:

systemctl restart nginx

6. Fine-tuning Synapse.

• Add a shared secret to the config file at /etc/matrix-synapse/homeserver.yaml:

Registration_shared_secret: <add random characters here, whatever you want your secret to be>

• Synapse caches conversation information in RAM where possible, and will use as much as you allow. For small implementations, (>50 users), you probably need about 512MB of RAM.
• You can configure this by adding the SYNAPSE_CACHE_FACTOR environment variable to /etc/default/matrix-synapse

`SYNAPSE_CACHE_FACTOR 0.02``

7. Run Synapse.

• Apply the settings by enabling and restarting the Synapse service:

systemctl restart matrix-synapse

systemctl enable matrix-synapse

8. Register your first Matrix user.

One of the major things you probably want this chat server for is a secure means of communication for your business. To do that, we need some user accounts, let's start by creating your own.
• Create a new user by running the following, and answering the prompts:

register_new_matrix_user -c /etc/matrix-synapse/homeserver.yaml https://localhost

- New user localpart [root]: {add your name/handle here}

- Password:

- Confirm password:

- Make admin [no]: yes

- Sending registration request...

- Success.

9. Enabling self-service user registrations.

Optional: to avoid having to register new users via CLI on your server every time, you can enable GUI user registration through the Riot client by editing /etc/matrix-synapse/homeserver.yaml and changing the following setting:

enable_registration: true

Otherwise, to register additional users, run register_new_matrix_user -c /etc/matrix-synapse/homeserver.yaml https://localhost again to manually configure more accounts. Make sure you don't set them all as admins.

Run your end-to-end encrypted chat server using Matrix and Riot.

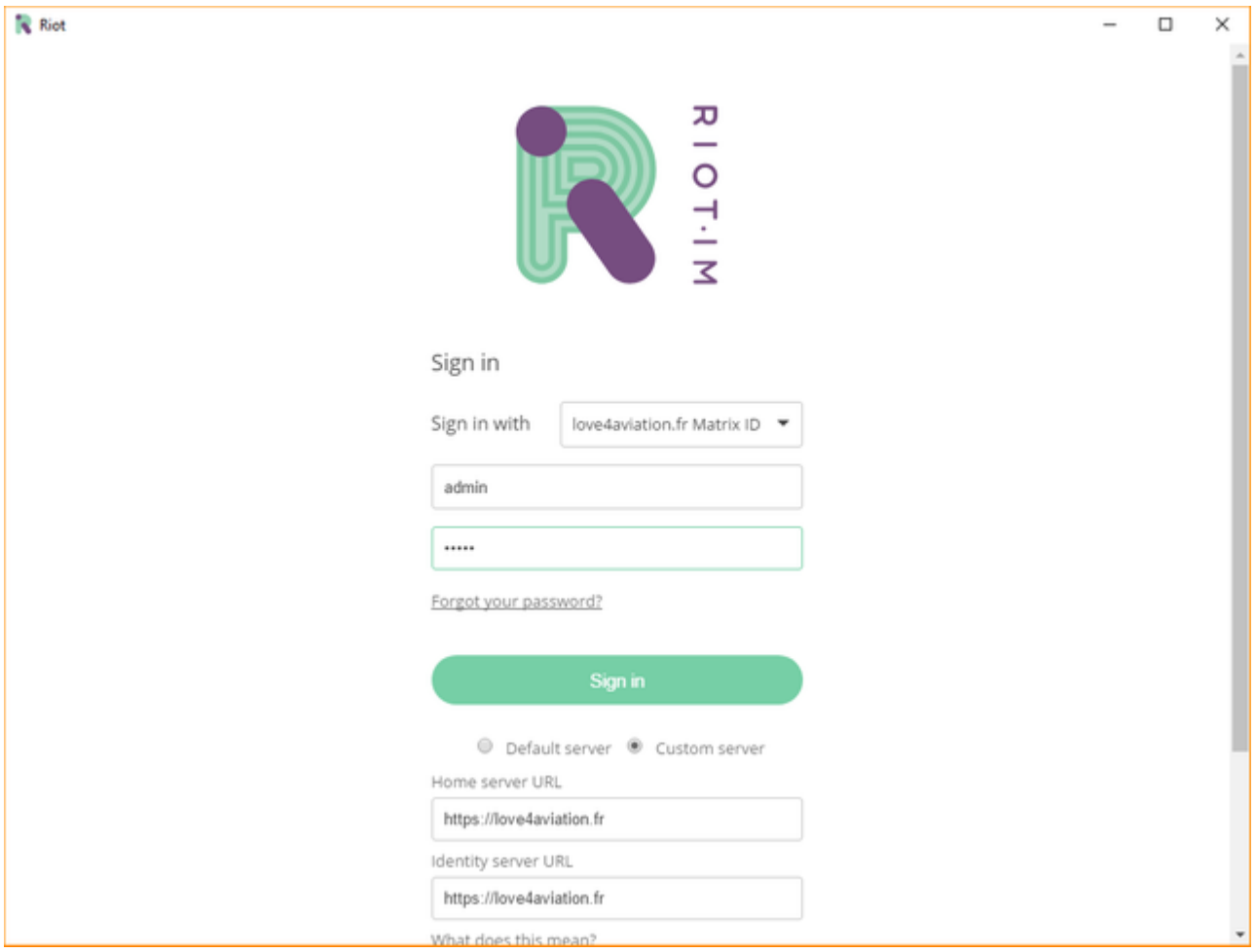
10. Running Riot.

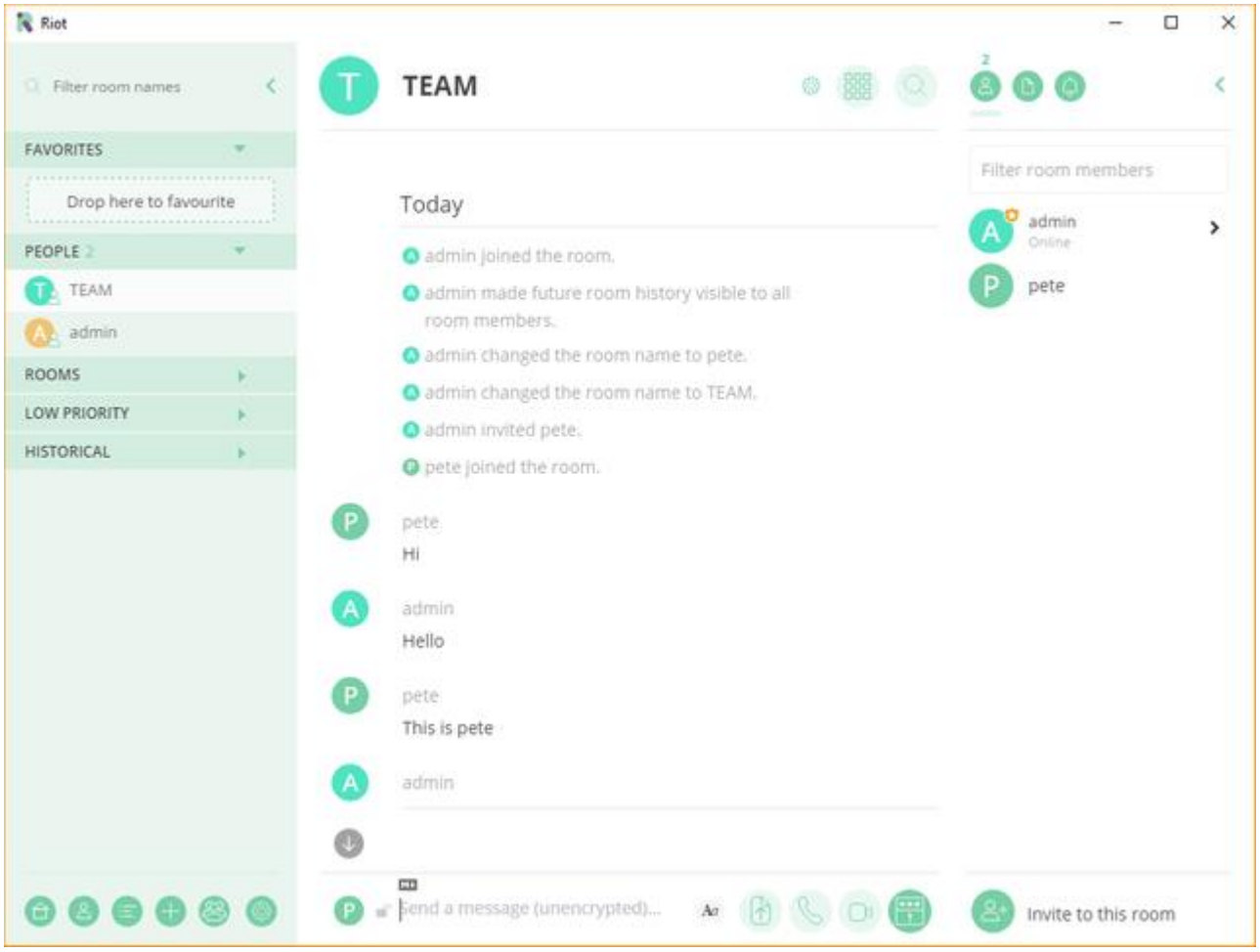
Riot is the front-end client for the server we just set up. If you don't have it already, you can download the app for your OS of choice at https://riot.im/
If Riot tries to auto-connect you to their default servers, log out. We need the Riot login screen for the next step.

Next, let's connect Riot to the server we just configured.

Add your hostname (either your custom hostname or the provided hostname from earlier):

Home server URL: https://ubinodes.org
Identity server URL: https://ubinodes.org





You can now join any room on the Matrix network. Here is our public room: **#foo:Ubinodes.org.**

11. Pros of running your own server.

• Retain control over your data by running a script to clean up deleted rooms as needed.

•Enjoy improved privacy settings that allow you to choose what information you share, giving you a clear advantage over users on vector.im.

• Opt for a personal server for identity, avoiding reliance on vector.im's server. This permits the use of your domain name for team member identification or LDAP integration. An example of such an identity service is mxisd:

• Demonstrates notably faster performance than utilizing Matrix's free server.

• Enables businesses to authenticate users using their own domain, enhancing security. In public rooms with diverse participants, restricting registrations to the organization's domain helps prevent social engineering by ensuring that only team members can register with their associated domain.

12. Cons of running your own server.

• Challenging to set up, requiring meticulous firewall configuration, and necessitating a skilled system administrator for ongoing server maintenance.

• Despite claiming decentralization, it operates on a federated model. To achieve redundancy, a minimum of two servers is essential—one for hosting rooms and another as the user gateway. This configuration ensures data from the central room is pushed to edge rooms, creating redundancy. However, if the central room experiences downtime, other rooms will be affected.

• Operating your identity server necessitates connection with vector.im. If the server encounters issues, users won't be able to reuse their ID to reconnect to Matrix.

• Managing your ID server involves setting up and maintaining your plugin, as opposed to using Vector.im directly, which may not be a worthwhile endeavor.

13. Conclusion.

Running Matrix home servers on a dedicated domain is crucial to mitigate potential attacks on web applications hosted on the same domain. This helps by limiting the exposure of malicious user-generated content served through a Matrix API. This recommendation is especially important when both a Matrix web client and server are shared on the same domain.