Matrix Synapse

Matrix is an open standard for interoperable, decentralised, real-time communication over IP. It can be used to power Instant Messaging, VoIP and Internet of Things communication - or anywhere you need a standard HTTP API for publishing and subscribing to data whilst tracking the conversation history.

Synapse is a reference homeserver implementation from the core development team at matrix.org, written in Python/Twisted.

In this guide, we will show you step-by-step how to install and configure Synapse on Ubuntu 18.04. We will configure Synapse and the Nginx web server as a reverse proxy for it and implement the HTTPS connection between clients and the front-end Nginx web server. We will also show how to set up a PostgreSQL database for better performance.

This guide explains one way to setup a Synapse server. There are many other correct ways to setup a Matrix server and that is the reason why there are so many guides. Feel free to choose the guide that suits your setup the best.

How to install Synapse on Ubuntu 18.04 LTS

Prerequisites

- Ubuntu 18.04 secured with basic security
- Root privileges
- A domain name for your server

What we will do

- Update and Upgrade System
- Install Synapse
- Configure Synapse
- Generate SSL certificates using Let's Encrypt
- Install and configure Nginx as a reverse proxy
- Install and configure Postgres instead of SQLite (optional but highly recommended, SQLite should not be used in production)
- Setup UFW Firewall
- Create a new Matrix user on your server
- Check federation
- Test the installation

Step 1 - Update and Upgrade System

Read the whole tutorial before starting to install the server.

Login to your Ubuntu server and add the repository key to make sure any installations and updates have been signed by the developers and to stop any unauthorized packages from being installed on your server.

```
sudo apt install -y lsb-release wget apt-transport-https
sudo wget -O /usr/share/keyrings/matrix-org-archive-keyring.gpg
https://packages.matrix.org/debian/matrix-org-archive-keyring.gpg
echo "deb [signed-by=/usr/share/keyrings/matrix-org-archive-keyring.gpg]
https://packages.matrix.org/debian/ $(lsb_release -cs) main" | sudo tee
/etc/apt/sources.list.d/matrix-org.list
```

Update the repository and upgrade all packages using the apt command below.

```
sudo sh -c 'apt update && apt upgrade'
```

Step 2 - Install Synapse

Install matrix-synapse using the apt command as below. (You can add the option `-y` to assume “yes” as answer to all prompts and run non-interactively) The name is `matrix-synapse-py3` because there is already another package name `synapse`.There is also a `matrix-synapse` package available but this uses Python 2 and it will stop being updated soon as Python 2 reaches end of life.

```
sudo apt install matrix-synapse-py3
```

During the installation, it will ask you about the matrix server name - type in your domain `example.com`. (We will not use `matrix.example.com`, because we also don't use `mail.example.com` for our E-Mails. This will work with well.known, SRV-records and nginx.

Don't leave the hostname blank during setup.

If you want to provide the team with information about your setup with an anonymous data report, choose 'Yes', otherwise leave it at 'No'.

When the Synapse installation is complete, start the service and enable it to launch everytime at system boot.

```
sudo systemctl start matrix-synapse.service
sudo systemctl enable matrix-synapse.service
```

Synapse is now up and running using the default configuration on port '8008' and '8448'. Check the open ports using ss (former netstat) command.

```
ss -plntu
```

Set up well.known

On your webserver a file at `/.well-known/matrix/server` has to be set up with the following

content

```
{
    "m.server": "synapse.example.com:443"
}
```

Where / is the root of your webserver. So if you navigate to https://example.com/.well-known/matrix.server it may try to download the `server` file or show it directly.

Step 3 - Configure Synapse

After the Synapse installation, we will configure it to run under the local IP address, disable Synapse registration, and enable the registration-shared-secret.

Before editing the home server configuration, we need to generate the shared secret key with the following command.

```
cat /dev/urandom | tr -dc 'a-zA-Z0-9' | fold -w 32 | head -n 1
```

And you will get a generated key. We will disable the registration for now and then copy the key into the homeserver configuration file. To disable the Synapse registration, uncomment the `registration_shared_secret` (Delete the `#` and don't leave a whitespace)

```
sudoedit /etc/matrix-synapse/homeserver.yaml
```

```
enable_registration: False

registration_shared_secret: [shared_secred_key]
```

Check ports

The best is to leave it default as it comes delivered (watch here https://github.com/matrix-org/synapse/blob/master/docs/sample_config.yaml), so check if it matches the follwing:

```
sudoedit /etc/matrix-synapse/homeserver.yaml
```

```
- port: 8008
    tls: false
    bind_addresses: ['::1', '127.0.0.1']
    type: http
x_forwarded: true
```

Be aware that indentation is important in *.yaml files!

Save and exit.

Note: `registration_shared_secret`: **If set allows registration by anyone who also has the shared secret, even if registration is disabled.**

Now restart the Synapse services.

```
sudo systemctl restart matrix-synapse.service
```

Check the homeserver service with the following command

```
ss -plntu
```

You will get the Synapse service is now on the local IP address. And we have completed the Synapse installation and configuration.

Step 4 - Generate SSL Letsencrypt Certificates

In this tutorial, we will enable HTTPS for the Nginx reverse proxy, and we will generate the SSL certificate files from Letsencrypt. So, start with installing the letsencrypt tool. (it is possible to add -y again)

```
sudo apt install letsencrypt
```

If nginx is installed first, lets stop nginx so certbot can listen to port 80

```
sudo systemctl stop nginx.service
```

Install the most recent certbot

```
sudo add-apt-repository ppa:certbot/certbot
sudo apt-get install certbot python-certbot-nginx
```

Generate the SSL certificate files for the matrix domain name `example.com` using the certbot command as shown below.

```
sudo certbot --nginx
```

The Letsencrypt tool will generate SSL certificate files by running the 'standalone' temporary web server for verification. When it's complete, you will get the information that its done and where the certificates are stored. Usally the SSL certificate files for the Synapse domain name `example.com` are generated inside the `/etc/letsencrypt/live/` directory.

```
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator standalone, Installer None
Obtaining a new certificate
Performing the following challenges:
http-01 challenge for example.com
Waiting for verification...
Cleaning up challenges
```

```
IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at:
   /etc/letsencrypt/live/example.com/fullchain.pem
   Your key file has been saved at:
   /etc/letsencrypt/live/example.com/privkey.pem
   Your cert will expire on 2019-03-03. To obtain a new or tweaked
   version of this certificate in the future, simply run certbot
   again. To non-interactively renew *all* of your certificates, run
   "certbot renew"
 - Your account credentials have been saved in your Certbot
   configuration directory at /etc/letsencrypt. You should make a
   secure backup of this folder now. This configuration directory will
   also contain certificates and private keys obtained by Certbot so
   making regular backups of this folder is ideal.
 - If you like Certbot, please consider supporting our work by:

   Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
   Donating to EFF:                    https://eff.org/donate-le
```

There should already be a cronjob generater for automatic renewal of the certs, as they last only 90 days. To check if the cron is up

```
sudo certbot renew --dry-run
```

Step 5 - Install and configure Nginx as a reverse proxy

Now install the Nginx web server and configure it as a reverse proxy for the homeserver that is running on the port '8008'. Start with installing the Nginx web server using the apt command below. (it is possible to add -y again)

```
sudo apt install nginx
```

After the installation is complete, start the service and enable it to launch everytime at system boot

```
sudo systemctl start nginx.service
sudo systemctl enable nginx.service
```

Next, we will create a new virtual host configuration for the domain name `example.com`. Go to the '/etc/nginx' configuration directory and create a new virtual host file 'matrix'.

```
sudoedit /etc/nginx/sites-available/matrix
```

Paste the following configuration there, changing the domain `example.com` to your own:

```
server {
       listen 80;
       server_name example.com;
       return 301 https://$server_name$request_uri;
}
```

```
server {
    listen 443 ssl;
    listen [::]:443 ssl;
    server_name example.com;

    ssl_certificate /etc/letsencrypt/live/example.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/example.com/privkey.pem;

    # If you don't wanna serve a site, comment this out
    root /var/www/html;
    index index.html index.htm;

    location /_matrix {
      proxy_pass http://127.0.0.1:8008;
      proxy_set_header X-Forwarded-For $remote_addr;
    }
    location /.well-known/matrix/server {
      return 200 '{"m.server": "example.com:443"}';
      add_header Content-Type application/json;
    }
    location /.well-known/matrix/client {
      return 200 '{"m.homeserver": {"base_url":
"https://example.com"},"m.identity_server": {"base_url":
"https://vector.im"}}';
      add_header Content-Type application/json;
      add_header "Access-Control-Allow-Origin" *;
    }
}
```

`location ~ ^/.well-known/matrix/client$ {` might need to escape the . Check it!

Same for `location ~ ^/.well-known/matrix/server$ {`?

Save and exit.

Activate the virtual host file and test the configuration.

```
sudo ln -s /etc/nginx/sites-available/matrix /etc/nginx/sites-enabled/
```

```
sudo nginx -t
```

If everything is fine, you should see the following output:

```
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

Make sure there is no error, then restart the Nginx services.

```
sudo systemctl restart nginx.service
```

Nginx installation and configuration as a reverse proxy for the Synapse homeserver has been completed.

(Optional) Step 6 - PostgreSQL instead of sqlite

While the step is marked as optional, it is **strongly encouraged** for any server that isn't purely for testing.

Initial PostgreSQL setup

```
sudo apt install postgresql
```

```
sudo -i -u postgres
```

```
psql
```

```
postgres=# CREATE USER "username" WITH PASSWORD 'password';
```

```
postgres=# CREATE DATABASE synapse ENCODING 'UTF8' LC_COLLATE='C'
LC_CTYPE='C' template=template0 OWNER "username";
```

Where username can be `synapse_user`, and password is a new strong password you set for postgresql.

To end the postgre line just type in `\q` and close the postegre-usershell with `exit`

Set up PostgreSQL for Synapse

```
sudo apt install python3-psycopg2
```

Afterwards edit in the homeserver.yaml the database section

```
sudoedit /etc/matrix-synapse/homeserver.yaml
```

```
database:
    name: psycopg2
    args:
        user: <user>
        password: <pass>
        database: <db>
        host: <host>
        cp_min: 5
        cp_max: 10
```

- `user` is in this case `synapse_user`
- `Database` should be the above created db, example `synapse`
- `Host` is the postgre hostname, usally `/var/run/postgresql/` or `127.0.0.1`

Now restart the Synapse services.

```
sudo systemctl restart matrix-synapse.service
```

Migrating from SQlite to PostgreSQL

Assuming you already followed step 6, there is no need for a migration. If you already used your Synapse and want to migrate, please refer to https://github.com/matrix-org/synapse/blob/master/docs/postgres.md

Step 7 - Setup UFW Firewall

Open the needed ports for our services. We will only allow SSH, HTTP, HTTPS and 8448 (for federation) connection on the UFW firewall configuration. To add them to the UFW firewall configuration, run the following commands.

```
sudo ufw allow ssh
sudo ufw allow http
sudo ufw allow https
sudo ufw allow 8448
```

Now enable the UFW firewall service and then check the status.

```
sudo ufw enable
sudo ufw status
```

Step 8 - Create a New Matrix User

At this stage, the Synapse homeserver installation and configuration is complete. And in this step, we need to add a new matrix user from the command line on the server. To create a new matrix user, run the command below.

```
register_new_matrix_user -c /etc/matrix-synapse/homeserver.yaml
http://localhost:8008
```

Now you need to input the user name, password, and decide whether the user will have the admin privileges or not. And we have created a new matrix user with admin privilege.

Step 9 - Testing

If you have used Riot with the desktop application before you may not want to log out, so it is better to go to https://riot.im/app/ and press “Launch now”. If you have used the web client before, download the Riot desktop application, install it and open the Riot software. With both you will get the Matrix login page now. Type the matrix username and password, then choose the 'Custom server' option and type the domain name from your server `example.com` in. Click the `Sign In` button and you will get to the Riot Dashboard.

The Synapse homeserver is up and running under the Nginx reverse proxy HTTPS connection, and the user is now logged in to the Synapse homeserver using the Riot application.

If you need two instances of riot instead, you can start it with argument, refer to Riot.im.

For another way to test it, go to `https://example.com/_matrix/static/` and you will be presented with a **`It works! Synapse is running`** screen or go to `https://example.com/_matrix/client/versions` and the output should look like the following:

```
unstable_features
m.lazy_load_members true
versions
0   "r0.0.1"
1   "r0.1.0"
2   "r0.2.0"
3   "r0.3.0"
```

Step 10 - Federation

You can test if federation is working using https://federationtester.matrix.org. If any of the checks show an error then federation won't work. Other federation-testers include:

- https://fed.mau.dev/

Explanations

Presence

Unfortunately presence is right now broken and generates a high load. It is possible to deactivate it, but the user avatars will be grey afterwards on the homeserver. To deactivate, open `homeserver.yaml` and add

```
sudoedit /etc/matrix-synapse/homeserver.yaml
```

```
use_presence: False
```

Do i need a TURN-Server (ex. COTURN)

It's only necessary when both parties are behind NAT. Otherwise 1-on-1 communication should work fine. Group-Calls via Riot will be handled with jitsi.riot.im and are not handled by the homeserver.

Port 8008 and 8448

TCP port 8008 is the port for clients, TCP port 8448 is the federation port for HTTPS.

Signature errors

Don't be worried about signature errors when joining rooms, timeouts from random domain names, and failed requests to random domain names.

Certificate errors

Certificates and LetsEncrypt

CLIENT and **FEDERATION** ports are **DIFFERENT**, they do not use the same port.

- **TCP 8448 (Default, can change):** Federation, HTTPS, original generated self-signed certificate, directly exposing port TCP 8448 of synapse (NO reverse proxy, NO replace certificate)
- **TCP 443:** Clients, HTTPS, regular certificate (e.g. Let's Encrypt), reversed proxy to port TCP 8008 of synapse

The self-signed certificate of synapse **SHOULD NOT** be replaced and port 8448 should only be used for federation (server traffic) and directly exposed publicly. For clients connections, a reverse proxy should be reachable publicly with a regular certificate (e.g. Let's Encrypt) on port 443 that goes to the port 8008 of synapse.

Why are certificate errors actually perfectly safe?

Because matrix (at this point) uses perspectives to validate certificates so there is no need to validate a certificate by an certificate authority. Tl;dr: Other matrix server look at the cert, and if they see the same cert your server does, you're not being MITM'ed (Man-in-the-middle), a bit like peer validation. It is possible to configure which peers are trusted in `homeserver.yaml`, by default it's just `matrix.org`.

Optional settings

Disable presence

Add `use_presence: False` in the `homeserver.yaml` to deactivate presence. (Improves the

performance dramstically at this moment, because presence is not working quite well).

Autojoin a room on registration

There is a setting for that.

```
sudoedit /etc/matrix-synapse/homeserver.yaml
```

```
# Users who register on this homeserver automatically join
# to these rooms
auto_join_rooms:
```

Troubleshooting

If your need help, get as much information as possible (Installed version, ...) and join https://matrix.to/#/#synapse:matrix.org. If it worked before, try to remember what was changed.

Whats my version

- https://example.com/_matrix/federation/v1/version

Location of logs

Check matrix with `journalctl -xe` and `systemctl status matrix-synapse`

A good way to check the logs is `tail -20 [PATH]`. `tail` will show the last lines of a file, with `-20` it is possible to see the last 20 lines.

Matrix

```
/var/log/matrix-synapse/homeserver.log
```

Nginx

```
/var/log/nginx/error.log
/var/log/nginx/application.log
```

Wipe Synapse

In case there is a new installation needed for whatever reason.

Stop the Synapse server

```
sudo systemctl stop matrix-synapse
```

Purge Synapse itself and everything related to it.

```
sudo apt purge matrix-synapse
```

```
sudo rm -r /var/log/matrix-synapse/ && sudo rm -r /var/lib/matrix-synapse/
&& sudo rm -r /etc/matrix-synapse/
```

Also, delete the Synapse PostgreSQL user.

Move Synapse to another server

In order to move to another server the following is needed:

- database (at `/var/lib/matrix-synapse/`)
- config files (*.yaml) (at `/etc/matrix-synapse/`
    - log.config
- server keys
- media store (at `/var/lib/matrix-synapse/`)

Wipe History of a room

It is not possible because it is a federated system. It is possible to redact messages but other servers need to be trusted to actually redact the messages. Think of Matrix like email in sense that once someone has a copy of a message its not possible to force them to do anything with it.

About this guide

For feedback about this guide or tips on how to improve it visit https://matrix.to/#/#synapseguide:matrix.org