PC-Admin/PC-Admin-s-Synapse-Setup-Guide

Originalartikel

Backup

<html> <h3>README.md</h3><article class=„markdown-body entry-content“ itemprop=„text“> This guide covers complete Synapse setup for Debian 9. It includes the often missing sections on how to configure postgresql and coturn with Synapse. You can use this guide to make your own encrypted chat server. You will need at least a 1GB VPS although I recommend 2GB. You will also need a desired domain name. My guide will use ‘yourserver.org’ with Riot-Web hosted through NGINX on the same server. You may wish to have your matrix service hosted at another prefix like ‘matrix.yourserver.org’. Contact me at: perthchat@protonmail.com or at: @PC-Admin:perthchat.org if you get stuck or have an edit in mind. <hr/><h2>Server Setup</h2> Configure a Debian 9 server with auto-updates, security and SSH access. <hr/><h2>DNS Records</h2> Set up a simple A record. With ‘yourserver.org’ pointed to your servers IP. <hr/><h2>Prepare Server</h2>

```
$ sudo apt install -y apt-transport-https lsof curl python python-pip
```

 Inside /etc/apt/sources.list.d/matrix.list, add the following two lines: <pre>deb https://matrix.org/packages/debian/ stretch main deb-src https://matrix.org/packages/debian/ stretch main

</pre>

```
$ sudo nano /etc/apt/sources.list.d/matrix.list
```

 <hr/><h2>Installing Matrix</h2>

```
$ wget https://matrix.org/packages/debian/repo-key.asc
```

```
$ sudo apt-key add repo-key.asc
```

```
$ sudo apt update &amp;&amp; sudo apt upgrade &amp;&amp; sudo apt autoremove
```

```
$ sudo apt install matrix-synapse -y
```

 Asked to set name of your server, enter your desired URL here. (eg: yourserver.org) <hr/><h2>Configure Firewall</h2> Open the following ports: <pre>$ sudo ufw allow 443 $ sudo ufw allow 8448 $ sudo ufw allow 80

</pre> If you have an external firewall, open these ports there. <hr/><h2>Certbot Setup</h2>

```
$ sudo apt install certbot
```

 Test if server IP can be pinged first, if it can then run:

```
$ sudo certbot certonly
```

 Choose ‘spin up a temporary webserver’ enter a recovery email enter ‘yourserver.org’ as the domain <pre>Generating key (2048 bits): /etc/letsencrypt/keys/0000_key-certbot.pem Creating CSR: /etc/letsencrypt/csr/0000_csr-certbot.pem IMPORTANT NOTES: - Congratulations! Your certificate and chain have been saved at

```
 /etc/letsencrypt/live/yourserver.org/fullchain.pem. Your cert will
 expire on 2017-11-01.
```

</pre> for 3 month renewal, set a crontab:

```
$ sudo crontab -e
```

 Insert Line:

```
@monthly certbot renew --quiet --post-hook "systemctl reload nginx"
```

 ^ This doesn’t work. If anyone has the solution for auto-renewal please contact me. <pre>$ sudo ls /etc/letsencrypt/live/yourserver.org cert.pem chain.pem fullchain.pem privkey.pem README

</pre> <hr/><h2>Configure NGINX with A+ SSL</h2> Generate dhparam key and move it to your letsencrypt folder: <pre>$ openssl dhparam -out dhparam2048.pem 2048 $ sudo cp ./dhparam2048.pem /etc/letsencrypt/live/yourserver.org

</pre> Install NGINX and configure: <pre>$ sudo apt install nginx -y $ sudo nano /etc/nginx/conf.d/matrix.conf

</pre> Add: <pre>server {

```
    listen         80;
    server_name    yourserver.org;
    return         301 https://$server_name$request_uri;
```

} server {

```
  listen 443 ssl;
  server_name yourserver.org;
  ssl_certificate     /etc/letsencrypt/live/yourserver.org/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/yourserver.org/privkey.pem;
  ssl_protocols       TLSv1 TLSv1.1 TLSv1.2;
```

```
  ssl_ciphers         'ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-
POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-
ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-
SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-
SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-
SHA:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA:ECDHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA256:DHE-RSA-
AES256-SHA:ECDHE-ECDSA-DES-CBC3-SHA:ECDHE-RSA-DES-CBC3-SHA:EDH-RSA-DES-CBC3-
SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-
SHA:AES256-SHA:DES-CBC3-SHA:!DSS';
  ssl_dhparam         /etc/letsencrypt/live/yourserver.org/dhparam2048.pem;
  ssl_ecdh_curve      secp384r1;
  add_header Strict-Transport-Security "max-age=31536000; includeSubDomains"
always;
  location /_matrix {
      proxy_pass http://127.0.0.1:8008;
      proxy_set_header X-Forwarded-For $remote_addr;
  }

}
```

</pre> ^ Make sure to replace the server name here with yours! Restart service and renew SSL: <pre>$ sudo service nginx stop $ sudo certbot renew $ sudo service nginx start

</pre> <hr/><h2>Fine Tune Synapse</h2> Edit /etc/matrix-synapse/homeserver.yaml: <pre>$ sudo nano /etc/matrix-synapse/homeserver.yaml # A list of other Home Servers to fetch the public room directory from # and include in the public room directory of this home server # This is a temporary stopgap solution to populate new server with a # list of rooms until there exists a good solution of a decentralized # room directory. secondary_directory_servers:

1. matrix.org
2. vector.im

</pre> If you want you can also enable self registration and guest access: <pre>enable_registration: True # Allows users to register as guests without a password/email/etc, and # participate in rooms hosted on this server which have been made # accessible to anonymous users. allow_guest_access: True

</pre> There are other settings here you may want to adjust. I would do so one at a time, testing each change as you go. Also check environmental variables in

```
/etc/default/matrix-synapse
```

for a small server (<=2GB), you will want to edit in a low cache factor: <pre>$ sudo nano /etc/default/matrix-synapse # Specify environment variables used when running Synapse # SYNAPSE_CACHE_FACTOR=1 (default) SYNAPSE_CACHE_FACTOR=0.05

</pre> Then restart synapse and examine the RAM usage:

```
$ sudo service matrix-synapse restart
```

 <hr/><h2>Load Riot-Web client into NGINX</h2> NGINX content location:

/usr/share/nginx/html/index.html https://github.com/vector-im/riot-web/releases/latest Download latest riot-web and move contents into nginx folder: <pre>~/riot-web$ wget https://github.com/vector-im/riot-web/releases/download/v0.11.4/riot-v0.11.4.tar.gz $ tar -zxvf ./riot-v0.11.4.tar.gz $ sudo rm -r /usr/share/nginx/html/* $ sudo mv ./riot-v0.11.4/* /usr/share/nginx/html/

</pre> Create and edit config.yaml in nginx directory: <pre>$ sudo cp /usr/share/nginx/html/config.sample.json /usr/share/nginx/html/config.json $ sudo nano /usr/share/nginx/html/config.json {

```
  "default_hs_url": "https://yourserver.org",
  "default_is_url": "https://vector.im",
  "brand": "Riot",
  "integrations_ui_url": "https://scalar.vector.im/",
  "integrations_rest_url": "https://scalar.vector.im/api",
  "bug_report_endpoint_url": "https://riot.im/bugreports/submit",
  "enableLabs": true,
  "roomDirectory": {
      "servers": [
          "matrix.org"
      ]
  },
  "welcomeUserId": "@riot-bot:matrix.org",
  "piwik": {
      "url": "https://piwik.riot.im/",
      "siteId": 1
  }
```

}

</pre> Reset NGINX:

```
$ sudo systemctl restart nginx
```

 You should be able to view and use Riot-Web through your URL now, test it out. <hr/><h2>Configure TURN service:</h2> Your matrix server still cannot make calls across NATs (different routers), for this we need to configure coturn. Configure a simple A DNS record pointing turn.yourserver.org to your servers IP.

```
$ sudo apt install coturn
```

 Generate a ‘shared-secret-key’, this can be done like so: <pre>$ < /dev/urandom tr -dc _A-Z-a-z-0-9 head -c64 V2OuWAio2B8sBpIt6vJk8Hmv1FRapQJDmNhhDEqjZf0mCyyIlOpf3PtWNT6WfWSh

</pre> Edited coturn configs: <pre>$ sudo nano /etc/turnserver.conf lt-cred-mech use-auth-secret static-auth-secret=[shared-secret-key] realm=turn.yourserver.org no-tcp-relay allowed-peer-ip=10.0.0.1 user-quota=16 total-quota=1200 min-port=49152 max-port=65535 $ sudo nano

/etc/default/coturn # # Uncomment it if you want to have the turnserver running as # an automatic system service daemon # TURNSERVER_ENABLED=1

</pre> Open port in firewall:

```
$ sudo ufw allow 3478
```

 Edit homeserver.yaml: <pre>$ sudo nano /etc/matrix-synapse/homeserver.yaml turn_uris: [„turn:turn.yourserver.org:3478?transport=udp“, „turn:turn.yourserver.org:3478?transport=tcp“] turn_shared_secret: shared-secret-key turn_user_lifetime: 86400000 turn_allow_guests: True

</pre> Restart both coturn and matrix-synapse and test: <pre>$ sudo systemctl start coturn $ sudo systemctl restart matrix-synapse

</pre> <hr/><h2>Configure PostgreSQL database</h2> By default synapse uses a sqlite3 database, performance and scalability is greatly improved by changing over to a PostgreSQL database. If you plan to ever have more than ~20 users I would recommend doing this. Install PostgreSQL

```
$ sudo apt install postgresql libpq-dev postgresql-client postgresql-client-common
```

 Create Role and Database

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

```
$ createuser synapse -P --interactive
```

 <pre>postgres@VM:~$ createuser synapse -P –interactive Enter password for new role: Enter it again: Shall the new role be a superuser? (y/n) n Shall the new role be allowed to create databases? (y/n) y Shall the new role be allowed to create more new roles? (y/n) y

</pre> Now we're back at $postgres. Let's create a database for Synapse with correct settings and set the owner to be the user we just created: Type:

```
psql
```

 ..And create the database as follows:

```
postgres=# CREATE DATABASE synapse WITH ENCODING 'UTF8' LC_COLLATE 'C' LC_CTYPE 'C' TEMPLATE template0 OWNER synapse;
```

 Exit from psql by typing

```
\q
```

 All done. Let's exit from postgres account by typing exit so land back at our own user. Next we modify postgres pg_hba.conf to allow all connections from localhost to the local

database server:

```
$ sudo nano /etc/postgresql/9.6/main/pg_hba.conf
```

 !NOTE „Paste it under the „Put your actual configuration here“

```
host all all 127.0.0.1/32 trust
```

 Restart postgresql after the change:

```
$ sudo service postgresql restart
```

 Shutdown matrix-synapse for now:

```
$ sudo service matrix-synapse stop
```

 Let's give the user ‘matrix-synapse’ access to bash temporary so we login to it's shell. The port process felt easier when I can actually work with the synapse user (python/envs/permissions work nicely) We will undo this change later:

```
$ sudoedit /etc/passwd
```

 !NOTE, I use „sudoedit“ by habit but you could also use „sudo nano /etc/passwd“ so it's up your preference. Change the shell for user matrix-synapse from /bin/false to /bin/bash, it's at the end of the row:

```
matrix-synapse:x:XXX:XXXXX::/var/lib/matrix-synapse:/bin/bash
```

 Now that Synapse is shutdown and we can login to matrix-synapse user:

```
$ sudo -i -u matrix-synapse
```

 You should land immediately to matrix-synapse's home directory which is /var/lib/matrix-synapse. Typing cd anytime brings you back here. Install psycopg2:

```
$ pip install psycopg2
```

 !NOTE Ignore any traceback errors if you get and do not use sudo as this is not an admin user! You should land immediately to matrix-synapse's home directory which is /var/lib/matrix-synapse. Typing cd anytime brings you back here. This location has the original SQLite homeserver.db, which we want to snapshot(copy) now, when Synapse is turned off. Let's take a snapshot:

```
$ cp homeserver.db homeserver.db.snapshot
```

 !NOTE, no need to use sudo anytime when you are logged in as matrix-synapse. This user is not an admin(in sudoers file) and it already has correct permissions for the needed files/db's/directories's. <pre>$ ls homeserver.db media uploads

</pre> Restart service for now: <pre>$ exit $ sudo service matrix-synapse start

</pre> Login back to matrix-synapse account:

```
$ sudo -i -u matrix-synapse
```

 Make a copy of the homeserver.yaml configuration file to be modified for our postgresql database settings:

```
$ cp /etc/matrix-synapse/homeserver.yaml /etc/matrix-synapse/homeserver-
postgres.yaml
```

 Modify the postgres database settings to the new homeserver-postgres.yaml -file:

```
$ nano /etc/matrix-synapse/homeserver-postgres.yaml
```

 Fill in the database section as follows, commenting out the old section: <pre>database:

```
  name: psycopg2
  args:
      user: synapse
      password: YOUR_SICK_DB_PASSWORD_PLEASE_SAVE_THIS_SOMEWHERE
      database: synapse
      host: localhost
      cp_min: 5
      cp_max: 10
```

</pre> !NOTE user,password,database are the values we created with psql before. Download synapse_port_db.py: https://github.com/matrix-org/synapse/blob/master/scripts/synapse_port_db Set excecute permissions to the synapse_port_db.py -script:

```
$ chmod +x synapse_port_db.py
```

 Now we are ready to try the port script against the homeserver.db.snapshot:

```
$ python synapse_port_db.py --sqlite-database homeserver.db.snapshot --
postgres-config /etc/matrix-synapse/homeserver-postgres.yaml --curses -v
```

 This should run a long time if you've used SQLite DB for a while. The –curses and -v flags at the end help you visualize what's going on. It will show you in real time what data is migrated from the homeserver.db.snapshot to your new postgresql database. At the end the screen should be pretty much all green (I think I had like 2 „events“ missing). Press any key.. Almost at the finale. To complete the conversion shut down the synapse server and run the port script one last time, e.g. if the SQLite database is at homeserver.db: Move back to your normal user account (eg. exit from matrix-synapse): <pre>$ exit $ sudo service matrix-synapse stop

</pre> Change user back to matrix-synapse:

```
$ sudo -i -u matrix-synapse
```

 And let's run the portscript again to bring the latest changes to postgresql:

```
$ python synapse_port_db.py --sqlite-database homeserver.db --postgres-
config /etc/matrix-synapse/homeserver-postgres.yaml --curses -v
```

 This shouldn't take so long as it quickly figures to import incrementally (e.g) only the data that has changed during Synapse was up. Last step is to rename our new homeserver-postgresql.yaml to homeserver.yaml e.g: <pre>$ cd /etc/matrix-synapse/ $ mv homeserver.yaml homeserver.yaml.old $ mv homeserver-postgres.yaml homeserver.yaml

</pre> And restart Synapse: <pre>$ exit (from matrix-synapse user) $ sudo service matrix-synapse start

</pre> Synapse should now be running against PostgreSQL, awesome! Final thing is to deny shell from matrix-synapse, like it was before: <pre>$ sudoedit /etc/passwd ... matrix-synapse:x:XXX:XXXXX::/var/lib/matrix-synapse:/bin/*false*

</pre> Done! Cleanup these old files after testing: <pre>/var/lib/matrix-synapse/homeserver.db /var/lib/matrix-synapse/homeserver.db.snapshot /var/lib/matrix-synapse/port-synapse.log /var/lib/matrix-synapse/synapse_port_db.py

</pre></article> </html>