Self Hosted Services

Tutorials, Guides, Tips, and Tricks for Self Hosted Online Services.

- Matrix
  - How to Install Matrix Synapse on Debian 11

Matrix

A new basis for open, interoperable, decentralized real-time communication

How to Install Matrix Synapse on Debian 11

DEPRECIATED

THIS GUIDE IS NO LONGER RECOMMENDED OR MAINTAINED

Introduction

Matrix is an **open standard** for **interoperable, decentralised, real-time** communication over IP.

- there exists an **open standard** in the form of the Matrix Specification
- it's **interoperable**, meaning it is designed to interoperate with other communication systems, and being an Open Standard means it's easy to see how to interoperate with it
- Matrix is **decentralised**, which means there is no central point - anyone can host their own server and have control over their data
- it is designed to function in **real-time**, which means it is ideal for building systems that require immediate exchange of data, such as Instant Messaging

---

This guide will detail how to stand up your own Matrix homeserver from start to finish on Debian 11. We will configure Synapse and Nginx as a reverse proxy to implement HTTPS/TLS connections between clients and the homserver. We will also be configuring a TURN/STUN service to allow video, voice, and screen sharing. This guide will also detail how to set up a PostgreSQL database for enhanced performance.

Note: We make use of the Tor network to connect to our remote server. We do this for anonymity and to prevent network observers from determining ownership of the VPS. This may be overkill for your threat model, so modify the given commands to suit your needs.

Prerequisites

- VPS with capability to fresh install a Debian 11 ISO
- Root or Superuser Privileges
- A domain name for your homeserver
- VNC or noVNC access to terminal

Overview

- Debian 11 Expert Install with Full Disk Encryption
- Configure and Harden SSH
- Configure Firewall
- Configure DNS Records
- Install and Configure Synapse
- Install and Configure PostgreSQL
- Install and Configure Nginx
- Install and Configure SSL (ZeroSSL) Certs
- Install and Configure TURN/Coturn
- Install and Configure Matrix Registration for Token Based Registration
- Install and Configure Element Web

Debian 11 Expert Install with Full Disk Encryption

1. Select ***Advanced Options***
2. Select ***Expert Install***
3. For *Choose Language* select ***English***
4. For *Select your Location* select ***United States***
5. For *Configure locales* select ***United States - en_US.UTF-8***
6. Skip *Additional locales* by selecting ***Continue***
7. For *Configure the keyboard* select ***American English***
8. For *Detect and mount installaion media* select ***Continue***
9. For *Load installer components from installation media* select ***Continue***
10. For *Configure the network* select **Yes** under *Auto-configure networking*
11. Under *Wait time for link detection* select ***Continue*** with the default 3 seconds
12. For *Configure the network* under *hostname* enter ***your-hostname-here***
13. Leave the default domain name under domain name. **unknown** in this instance.
14. For *Set up users and passwords* under *Enable shadow passwords* select ***Yes***
15. For *Set up users and passwords* under *Allow login as root* select ***No***
16. For *Set up users and passwords* under *Full name for the new user* type ***superuser***
17. For *Set up users and passwords* under *Username for your account* type ***superuser***

18. For *Set up users and passwords* under *Choose a password for the new user* copy paste from ***KeePassXC*** entry
19. For *Configure the clock* under *Set the clock using NTP* select ***Yes***
20. For *Configure the clock* under *NTP server to use* leave the default ***0.debian.pool.ntp.org***
21. For *Configure the clock* under *Select your time zone* choose ***Coordinated Universal Time (UTC)***
22. For *Partition disks* under *Partitioning method* choose ***Guided - use entire disk and set up encrypted LVM***
23. For *Partition disks* under *Select disk to partition* select ***Virtual disk 1 (vda)***
24. For *Partition disks* under *Partitioning scheme* select ***All files in one partition***
25. For *Partition disks* under *Write the changes to disk and configure LVM* select ***Yes***
26. For *Partition disks* under *Enter LUKS Password* copy paste long and complex ***KeePassXC PASSPHRASE***
    - ***MAKE SURE YOU USE AN EASY TO TYPE PASS PHRASE!!!! I CAN'T STRESS THIS ENOUGH!!***
27. For *Partition disks* under *Name of the volume group* use ***your-hostname-vg***
28. For *Partition disks* under *Amount of volume group to use for guided partitioning* use ***##.# GB***
    - Use the default value to partition the entire drive
29. For *Partition disks* under *Overview* select ***Finish partitioning and write changes to disk***
30. For *Partition disks* under *Write the changes to disk* select ***Yes***
31. For *Install the base system* under *Kernel to install* select ***linux-image-amd64***
32. For *Install the base system* under *Drivers to include in the initrd* select ***generic: include all available drivers***
33. For *Configure the package manager* under *Scan extra installation media* select ***No***
34. For *Configure the package manager* under *Use a network mirror* select ***Yes***
35. For *Configure the package manager* under *Protocol for file downloads* select ***http***
36. For *Configure the package manager* under *Debian archive mirror country* select ***United States***
37. For *Configure the package manager* under *Debian archive mirror* select ***deb.debian.org***
38. For *Configure the package manager* under *HTTP proxy information* leave blank and select ***Continue***
39. For *Configure the package manager* under *Use non-free software* select ***No***
40. For *Configure the package manager* under *Use contrib software* select ***No***
41. For *Configure the package manager* under *Enable source repositories in APT* select ***Yes***
42. For *Configure the package manager* under *Services to use* select ***security updates*** and ***release updates***
43. For *Configuring discover* under *Updates management on this system* select ***No automatic updates***
    - Consider enabling unattended security upgrades, but it may (but probably won't) break something down the line
44. For *Configuring popularity-contest* under *Participate in the package usage survey* select ***No***
45. For *Software selection* under *Choose software to install* select ***standard system utilities*** and **deselect all other options**

46. For *Install the GRUB boot loader* under *Install the GRUB boot loader on your primary drive* select ***Yes***
47. For *Install the GRUB boot loader* under *Device for boot loader installation* select ***dev/vda***
48. For *Install the GRUB boot loader* under *Force GRUB installation to the EFI removable media path* select ***No***
49. For *Finish the installation* under *Is the system clock set to UTC* select ***Yes***
50. In the control panel shutdown the server, change boot order, and remove the ISO from the machine
51. Restart the server and enter the luks pass phrase.

Configure SSH

0. Update/Upgrade if needed
    - `sudo apt update`
    - `sudo apt upgrade -y`
1. Install SSH
    - `sudo apt update`
    - `sudo apt install ssh`
2. Change Default SSH Port
    - `sudo nano /etc/ssh/sshd_config`
    - `Port 55555`
        - Use any port you want, but we recommend high ports
        - For the rest of this guide, we'll be using port 55555 for reference

Create and Install SSH Keys

NOTE: THESE INSTRUCTIONS ARE FOR YOUR LOCAL HOST, NOT THE REMOTE SERVER!

1. Create a directory in a safe place on your local host to store the private key
    - **NOTE: CHANGE THESE DIRECTORY LOCATION TO SUIT YOUR NEEDS AND USERNAME**
    - `mkdir /home/user/ssh`
    - `cd /home/user/ssh`
2. Generate SSH Keys (ON YOUR LOCAL LINUX HOST)
    - `ssh-keygen-b 4096`
    - `NOTE: Make sure you use a password for your private key, store it in KeePassXC`
3. Copy public SSHkey from local host to remote server
    - **NOTE: CHANGE THIS IP TO YOUR HOMESERVER IP**
    - `torsocks ssh-copy-id -p 55555 -i matrix.pub superuser@88.777.66.55`
4. Test Login with SSH Keys
    - **NOTE: The 'matrix' key is the private SSH key / identity file, change this**
    - `torsocks ssh -p '55555' 'superuser@88.777.66.55' -i 'matrix'`
5. If test was successful, disable SSH password logins

- `sudo nano /etc/ssh/sshd_config`
- `PubKeyAuthentication yes`
- `PasswordAuthentication no`

6. Restart sshd
   - `sudo service sshd reload`
   - `sudo service sshd restart`

Install State(less) Firewall

Note: We chose ufw over iptables due to simplicity and lack of need for a stateful firewall

1. apt install ufw
   - `sudo apt update`
   - `sudo apt install ufw -y`
2. allow ports for services
   - Allow TURN ports
     - `sudo ufw allow 3478`
     - `sudo ufw allow 5349`
     - `sudo ufw allow 49152:65535/udp`
   - Allow SSH port
     - `sudo ufw allow 55555`
   - Allow Synapse Federation Port
     - `sudo ufw allow 8448`
   - Allow HTTP/S Traffic
     - `sudo ufw allow 443`
     - `sudo ufw allow 80`
   - Enable ufw
     - ***NOTE: MAKE SURE YOUR SSH PORT IS CORRECT, OTHERWISE YOU'LL BE LOCKED OUT***
     - `sudo ufw enable`

Create your DNS A record and point it towards your server

Install Synapse

1. Install the necessary packages
   - `sudo apt install -y lsb-release wget apt-transport-https`
2. Add the Matrix PGP key for APT
   - `sudo wget -O /usr/share/keyrings/matrix-org-archive-keyring.gpg https://packages.matrix.org/debian/matrix-org-archive-keyring.gpg`

3. Add the GPG key and Matrix repository to APT
   - `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`
4. Update and Install Synapse
   - `sudo apt update`
   - `sudo apt install matrix-synapse-py3`
5. Enter hostname from DNS A record
6. Select ***NO*** when asked to report anonyomous statistics

Install PostgreSQL

1. `sudo apt install postgresql`
2. Authenticate as database user
   - `sudo -u postgres bash`
3. Create postgres user and database
   - `createuser --pwprompt synapse_user`
   - `createdb --encoding=UTF8 --locale=C --template=template0 --owner=synapse_user synapse`
4. Setup access to postgresql service
   - `nano /etc/postgresql/13/main/pg_hba.conf`
5. Add the following line as seen below
   - `host  synapse  synapse_user  ::1/128  md5`

```
# Database administrative login by Unix domain socket
local   all             postgres                                peer
# TYPE  DATABASE        USER            ADDRESS                 METHOD
host    synapse     synapse_user    ::1/128     md5
# "local" is for Unix domain socket connections only
local   all             all                                     peer
# IPv4 local connections:
host    all             all             127.0.0.1/32            md5
# IPv6 local connections:
host    all             all             ::1/128                 md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication     all                                     peer
host    replication     all             127.0.0.1/32            md5
host    replication     all             ::1/128                 md5
```

6. Type `exit` to quit postgresql shell
7. Reload the Postgresql configuration
   - `sudo systemctl reload postgresql`
8. Edit homeserver.yaml

- `sudo nano /etc/matrix-synapse/homeserver.yaml`

OPTIONAL:

- install rsync on **SERVER** to edit homeserver.yaml
  - `sudo apt install rsync -y` (ON SERVER)
- install rsync on **YOUR PERSONAL COMPUTER**
  - `sudo apt install rsync -y`
- pull down homeserver.yaml to edit locally
  - `torsocks rsync -rvz -e "ssh -p 55555 -i /home/user/ssh/matrix" superuser@88.777.66.55:/etc/matrix-synapse/homeserver.yaml /home/user/matrix/configs/`

9. Comment out existing database settings in homeserver.yaml

```
#database:
#  name: sqlite3
#  args:
#    database: /var/lib/matrix-synapse/homeserver.db
```

10. Uncomment the entry for postgres configuration while adding postgresql password from earlier

```
database:
  name: psycopg2
  txn_limit: 10000
  args:
    user: synapse_user
    password: secretpassword
    database: synapse
    host: localhost
    port: 5432
    cp_min: 5
    cp_max: 10
```

11. ***OPTIONAL*** Configure homeserver.yaml settings for user privacy

```
# Whether to require authentication to retrieve profile data (avatars,
# display names) of other users through the client API. Defaults to
# 'false'. Note that profile data is also available via the federation
# API, unless allow_profile_lookup_over_federation is set to false.
#
require_auth_for_profile_requests: true
```

```
# Uncomment to require a user to share a room with another user in order
# to retrieve their profile information. Only checked on Client-Server
# requests. Profile requests from other servers should be checked by the
# requesting server. Defaults to 'false'.
#
limit_profile_requests_to_users_who_share_rooms: true
```

```
# Uncomment to prevent a user's profile data from being retrieved and
# displayed in a room until they have joined it. By default, a user's
# profile data is included in an invite event, regardless of the values
# of the above two settings, and whether or not the users share a server.
# Defaults to 'true'.
#
include_profile_data_on_invite: false
```

```
# If set to 'true', allows any other homeserver to fetch the server's public
# rooms directory via federation. Defaults to 'false'.
#
allow_public_rooms_over_federation: true
```

```
# How long to keep redacted events in unredacted form in the database. After
# this period redacted events get replaced with their redacted form in the DB.
#
# Defaults to `7d`. Set to `null` to disable.
#
redaction_retention_period: 1m
```

```
# How long to track users' last seen time and IPs in the database.
#
# Defaults to `28d`. Set to `null` to disable clearing out of old rows.
#
user_ips_max_age: 1m
```

```
retention:
  # The message retention policies feature is disabled by default. Uncomment the
  # following line to enable it.
  #
  enabled: true
```

```
  # Default retention policy. If set, Synapse will apply it to rooms that lack the
  # 'm.room.retention' state event. Currently, the value of 'min_lifetime' doesn't
  # matter much because Synapse doesn't take it into account yet.
  #
  default_policy:
    min_lifetime: 3d
    max_lifetime: 3w

  # Retention policy limits. If set, and the state of a room contains a
  # 'm.room.retention' event in its state which contains a 'min_lifetime' or a
  # 'max_lifetime' that's out of these bounds, Synapse will cap the room's policy
  # to these limits when running purge jobs.
  #
  allowed_lifetime_min: 3d
  allowed_lifetime_max: 3w
```

```
  purge_jobs:
    - longest_max_lifetime: 3w
      interval: 1d
    - shortest_max_lifetime: 3d
      interval: 1d
```

```
# The minimum TLS version that will be used for outbound federation requests.
#
# Defaults to `1`. Configurable to `1`, `1.1`, `1.2`, or `1.3`. Note
# that setting this value higher than `1.2` will prevent federation to most
# of the public Matrix network: only configure it to `1.3` if you have an
# entirely private federation setup and you can ensure TLS 1.3 support.
#
federation_client_minimum_tls_version: 1.2
```

```
# Uncomment to disable profile lookup over federation. By default, the
# Federation API allows other homeservers to obtain profile data of any user
# on this homeserver. Defaults to 'true'.
#
allow_profile_lookup_over_federation: false
```

```
# Uncomment to disable device display name lookup over federation. By default, the
# Federation API allows other homeservers to obtain device display names of any user
# on this homeserver. Defaults to 'true'.
```

```
#
allow_device_name_lookup_over_federation: false
```

```
## Metrics ###


# Enable collection and rendering of performance metrics
#
enable_metrics: false
```

```
user_directory:
  # Defines whether users can search the user directory. If false then
  # empty responses are returned to all queries. Defaults to true.
  #
  # Uncomment to disable the user directory.
  #
  enabled: false
```

12. ***OPTIONAL*** Require Invite Tokens to Register

```
# Require users to submit a token during registration.
# Tokens can be managed using the admin API:
# https://matrix-org.github.io/synapse/latest/usage/administration/admin_api/registration_tokens.html
# Note that `enable_registration` must be set to `true`.
# Disabling this option will not delete any tokens previously generated.
# Defaults to false. Uncomment the following to require tokens:
#
registration_requires_token: true
```

```
# If set, allows registration of standard or admin accounts by anyone who
# has the shared secret, even if registration is otherwise disabled.
#
registration_shared_secret: "ENTER YOUR REGISTRATION SHARED SECRET HERE"
```

```
# 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: false
```

Install and configure nginx reverse proxy

1. `sudo apt install nginx -y`

Install ZeroSSL certs with acme.sh

- We ran into issues with WebRTC and certain devices rejecting LetsEncrypt certifications for TURN/STURN connections. We used ZeroSSL to prevent further issues.

1. Switch to root user
   - `sudo su`
2. Change to root home directory
   - `cd`
3. Install prerequisite packages
   - `apt install git curl`
4. Clone the acme.sh git
   - `git clone https://github.com/acmesh-official/acme.sh.git`
   - `cd acme.sh`
5. Enable execution permissions
   - `chmod +x acme.sh`
6. Run acme.sh and register an account using a valid email
   - `./acme.sh --register-account -m you@yoursite.com --server zerossl`
7. Take note of your ACCOUNT_TRUMBPRINT value
8. Standup an nginx site to validate domain ownership
   - `sudo nano /etc/nginx/conf.d/matrix.conf`

NOTE: acme.sh will modify this nginx site config to validate certificates, but add your ACCOUNT_THUMBPRINT value after the return 200 value if needed.

```
server {
    listen 80;
    listen [::]:80;
    server_name yoursite.com;
    location ~ "/.well-known/acme-challenge/" {
    default_type text/plain;
    return 200 "$1.YOUR_ACCOUNT_THUMBPRINT";
}
}
```

8. Create SSL Certs Directory
   - `mkdir -p /etc/zerossl/live/yoursite.com/`
9. Issue New Certs for your site
   - `./acme.sh --issue -d yoursite.com --nginx --server zerossl`

10. Copy and Install the Certificates (***Be sure to replace all the domains and paths***)
    - `./acme.sh --install-cert -d yoursite.com --cert-file /etc/zerossl/live/yoursite.com/cert.pem --key-file /etc/zerossl/live/yoursite.com/privkey.pem --fullchain-file /etc/zerossl/live/yoursite.com/fullchain.pem --ca-file /etc/zerossl/live/yoursite.com/chain.pem --reloadcmd "service nginx force-reload"`

You should see the following output:

```
Installing cert to: /etc/zerossl/live/yoursite.com/cert.pem
Installing CA to: /etc/zerossl/live/yoursite.com/chain.pem
Installing key to: /etc/zerossl/live/yoursite.com/privkey.pem
Installing full chain to: /etc/zerossl/live/yoursite.com/fullchain.pem
```

11. Configure nginx config file (completely replace the earlier conf lines)
    - `sudo nano /etc/nginx/conf.d/matrix.conf`

- **NOTE**: this nginx configuration enforces the following modifications
  - Access and Error logs turned off for user privacy
  - Server tokens turned off to prevent nginx fingerprinting
  - Mozilla 'modern' recommended SSL configuration
  - TLS 1.3 and TLS 1.2 with most secure ciphers with perfect forward security
  - Configured for an A+ on SSL Labs

```
server {
    listen 80;
    listen [::]:80;
    server_name yoursite.com;

    access_log off;
    error_log off;
    server_tokens off;

    return 301 https://$host$request_uri;
}

server {
  server_name www.yoursite.com;
  return 301 $scheme://yoursite.com$request_uri;

  error_log off;
  access_log off;
  server_tokens off;
}
```

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

    ssl on;
    ssl_certificate /etc/zerossl/live/yoursite.com/fullchain.pem;
    ssl_certificate_key /etc/zerossl/live/yoursite.com/privkey.pem;
    ssl_session_timeout 1d;
    ssl_session_cache shared:MozSSL:10m;  # about 40000 sessions
    ssl_session_tickets off;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_prefer_server_ciphers on;
    ssl_ciphers "ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES256-GCM-SHA384";

    access_log off;
    error_log off;
    server_tokens off;

# NOTE THE BELOW LOCATION IS THERE IF YOU WANT TO RUN A HOME PAGE LOCATED AT /var/www/yoursite.com
    location / {
add_header Strict-Transport-Security "max-age=31536000; includeSubdomains" always;
        root /var/www/yoursite.com;
        gzip on;
        gzip_disable "msie6";
        gzip_vary on;
        gzip_proxied any;
        gzip_min_length 1024;
        gzip_comp_level 6;
        gzip_buffers 16 8k;
        gzip_http_version 1.1;
        gzip_types text/plain text/css application/json application/javascript text/xml application/xml application/xml+rss text/javascript;

    }
```

```
# NOTE THE BELOW LOCATION IS CONFIGURED FOR MATRIX REGISTRATION
    location /register {
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
add_header Strict-Transport-Security "max-age=31536000; includeSubdomains" always;
proxy_pass http://localhost:5000;
        gzip on;
        gzip_disable "msie6";
        gzip_vary on;
        gzip_proxied any;
        gzip_min_length 1024;
        gzip_comp_level 6;
        gzip_buffers 16 8k;
        gzip_http_version 1.1;
        gzip_types text/plain text/css application/json application/javascript text/xml application/xml
application/xml+rss text/javascript;
    }

# NOTE THE BELOW LOCATION IS CONFIGURED FOR MATRIX REGISTRATION
    location /static {
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_pass http://localhost:5000;
        gzip on;
        gzip_disable "msie6";
        gzip_vary on;
        gzip_proxied any;
        gzip_min_length 1024;
        gzip_comp_level 6;
        gzip_buffers 16 8k;
        gzip_http_version 1.1;
        gzip_types text/plain text/css application/json application/javascript text/xml application/xml
application/xml+rss text/javascript;
    }

# NOTE THE BELOW LOCATION IS CONFIGURED FOR MATRIX REGISTRATION
    location /api {
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_pass http://localhost:5000;
    }
# NOTE THE BELOW LOCATION IS REQUIRED FOR MATRIX/SYNAPSE
    location ~ ^(/_matrix|/_synapse/client) {
```

```
        # note: do not add a path (even a single /) after the port in `proxy_pass`,
        # otherwise nginx will canonicalise the URI and cause signature verification
        # errors.
        proxy_pass http://localhost:8008;
        proxy_set_header X-Forwarded-For $remote_addr;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header Host $host;

        # Nginx by default only allows file uploads up to 1M in size
        # Increase client_max_body_size to match max_upload_size defined in homeserver.yaml
        client_max_body_size 50M;
    }
}

server {
    listen 8448 ssl http2 default_server;
    listen [::]:8448 ssl http2 default_server;
    server_name yoursite.com;

    ssl on;
    ssl_certificate /etc/zerossl/live/yoursite.com/fullchain.pem;
    ssl_certificate_key /etc/zerossl/live/yoursite.com/privkey.pem;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_session_timeout 1d;
    ssl_session_cache shared:MozSSL:10m;  # about 40000 sessions
    ssl_session_tickets off;
    ssl_prefer_server_ciphers on;
    ssl_ciphers "ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512:ECDHE-RSA-AES256-GCM-
SHA384:DHE-RSA-AES256-GCM-SHA384";

    add_header Strict-Transport-Security "max-age=31536000; includeSubdomains; always";

    access_log off;
    error_log off;
    server_tokens off;

    location ~ ^(/_matrix|/_synapse/client) {
        # note: do not add a path (even a single /) after the port in `proxy_pass`,
        # otherwise nginx will canonicalise the URI and cause signature verification
        # errors.
```

```
        proxy_pass http://localhost:8008;
        proxy_set_header X-Forwarded-For $remote_addr;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header Host $host;

        # Nginx by default only allows file uploads up to 1M in size
        # Increase client_max_body_size to match max_upload_size defined in homeserver.yaml
        client_max_body_size 50M;
    }
}
```

Install and Configure Turn Server

1. `sudo apt install coturn`
2. Create coturn readable ssl keys
   - `mkdir -p /etc/coturn/certs`
   - `sudo chown -R turnserver:turnserver /etc/coturn/`
   - `chmod -R 700 /etc/coturn/`
3. Create your static-auth-secret for TURN using pwgen
   - `pwgen -s 64 1`
4. Modify /etc/turnserver.conf
   - `sudo nano /etc/turnserver.conf`
5. Add the following lines to the bottom of the turnserver.conf file

```
use-auth-secret
static-auth-secret=YOUR-STATIC-AUTH-SECRET-HERE
realm=yoursite.com:5349
no-tcp-relay
denied-peer-ip=10.0.0.0-10.255.255.255
denied-peer-ip=192.168.0.0-192.168.255.255
denied-peer-ip=172.16.0.0-172.31.255.255
allowed-peer-ip=10.0.0.1
user-quota=12
total-quota=1200
cert=/etc/coturn/certs/fullchain.pem
pkey=/etc/coturn/certs/privkey.pem
```

6. Restart turn server
   - `sudo systemctl restart coturn`
7. Modify homeserver.yaml to reflect new turn server

```
turn_uris: [ "turns:yoursite.com:5349?transport=udp", "turns:yoursite.com:5349?transport=tcp" ]`
turn_shared_secret: "YOUR-STATIC-AUTH-SECRET-HERE"
turn_user_lifetime: 86400000
turn_allow_guests: false
```

8. Enable Cert support for coturn
    - `sudo cp /etc/zerossl/live/yoursite/fullchain.pem /etc/coturn/certs/`
    - `sudo cp /etc/zerossl/live/yoursite/privkey.pem /etc/coturn/certs/`
    - `sudo chown turnserver:turnserver -R /etc/coturn/`
    - `sudo service coturn force-reload`
    - `sudo service coturn restart`

Enable Token Based Registrations with Matrix-Registration

1. Create Admin User
    - `register_new_matrix_user -c homeserver.yaml http://localhost:8008`
    - Make sure you register user 'admin' as an admin
    - Store the Username/Password in KeePassXC
2. Create Registration User
    - `register_new_matrix_user -c homeserver.yaml http://localhost:8008`
    - Make sure you register user 'registration' as an admin
    - Store the Username/Password in KeePassXC
3. Login to your Matrix Homeserver as 'registration' using a client (like Element) and take note of your access token (starts with syt_)
    - This will be the access token used by the matrix registration service to register new accounts
    - Save this access token in your KeePassXC notes.
4. Install Matrix Registration
    - `sudo apt install python3-pip`
    - `pip3 install matrix-registration==1.0.0.dev7`
    - `pip3 install psycopg2-binary`
5. Update Flask_limiter usage
    - `https://github.com/zeratax/matrix-registration/pull/83/commits/92087a6e221482194b42b63848735e9e6d92b1f6`
    - (replace get_ipaddr with get_remote_address)
        - line 9 of matrix_registration/app.py
        - line 60 of matrix_registration/app.py
6. Create empty postgresql database
    - `sudo -u postgres bash`
    - `createuser --pwprompt matrix_reg_user`
        - Store this Password in KeePassXC, you'll need it later
    - `createdb --owner=matrix_reg_user matrix_reg`

7. Setup access to postgresql service
   - `nano /etc/postgresql/13/main/pg_hba.conf`
8. Add the following line below to pg_hba
   - `host  matrix_reg  matrix_reg_user  ::1/128  md5`
9. Restart postgresql and verify pg_hba accepted
   - `sudo service postgresql status`
10. Create the following **config.yaml** file for matrix-registration

```
server_location: 'http://localhost:8008'
server_name: 'yoursite.com'
registration_shared_secret: 'YOUR-REGISTRATION-SHARED-SECRET' # see your synapse's homeserver.yaml
admin_api_shared_secret: 'syt_YOUR-REGISTRATION-ACCESS-TOKEN' # to generate tokens via the web api
base_url: '/' # e.g. '/element' for https://example.tld/element/register
client_redirect: 'https://element.yoursite.com'
client_logo: 'static/images/element-logo.png' # use '{cwd}' for current working directory
db: 'postgresql://matrix_reg_user:YOUR-matrix_reg_user-PASSWORD@localhost:5432/matrix_reg'
host: 'localhost'
port: 5000
rate_limit: ["100 per day", "10 per minute"]
allow_cors: false
ip_logging: false
logging:
  disable_existing_loggers: true
  version: 1
  root:
    level: DEBUG
    handlers: [console, file]
  formatters:
    brief:
      format: '%(name)s - %(levelname)s - %(message)s'
    precise:
      format: '%(asctime)s - %(name)s - %(levelname)s - %(message)s'
  handlers:
    console:
      class: logging.StreamHandler
      level: INFO
      formatter: brief
      stream: ext://sys.stdout
    file:
      class: logging.handlers.RotatingFileHandler
      formatter: precise
```

```
    level: INFO
    filename: m_reg.log
    maxBytes: 10485760 # 10MB
    backupCount: 3
    encoding: utf8
# password requirements
password:
  min_length: 14
# username requirements
username:
  validation_regex: [ '[a-zA-Z0-9]' ]
  invalidation_regex: [
'(info|admin|null|123456|mail|fuck|webmaster|root|test|guest|adm|mysql|user|administrator|oracle|ftp|pi|puppet
|ansible|ec2-user|vagrant|azureuse|mod|moderator|host|postgres|synapse|support)' ]
```

- Note: The above regexes can be changed to enforce username restrictions.

Create Matrix Registration Service

1. Create new user
   - `sudo useradd -m registration`
     - Store Password in KeePassXC
     - NOTE: Make sure matrix-registration files are available in the /home/registration/ user folders as referenced in the below service file
2. Create service file
   - `sudo nano /etc/systemd/system/matrix-registration.service`

```
[Unit]
Description=Matrix Registration
After=matrix-synapse.service
BindsTo=matrix-synapse.service

[Service]
Type=simple
User=registration
Group=registration
WorkingDirectory=/home/registration/matrix-registration
ExecStart=/home/registration/.local/bin/matrix-registration --config-path /home/registration/matrix-
registration/config.yaml serve
Restart=always
```

```
[Install]
WantedBy=multi-user.target
```

3. Enable Service
   - `systemctl daemon-reload`
   - `systemctl --user enable matrix-registration`
   - `systemctl --user start matrix-registration`
4. Confirm matrix-registration is available by browsing to https://yoursite.com/register
5. Create an Invite Token and test matrix-registration is working

```
curl -X POST \
    -H "Authorization: SharedSecret syt_your_shared_secret_token" \
    -H "Content-Type: application/json" \
    -d '{"max_usage": 1, "expiration_date": "2023-01-01"}' \
    http://localhost:5000/api/token
```

Install Element Web

1. Add new elementweb user
   - `sudo useradd -m elementweb`
   - Store password in KeepassXC
2. Change directories in the elementweb home dir
   - `cd /home/elementweb/`
3. Download the latest element-web release (v1.10.12 of today)
   - `sudo --user elementweb wget https://github.com/vector-im/element-web/releases/download/v1.10.12/element-v1.10.12.tar.gz`
   - `sudo --user elementweb tar -xvf element-v1.10.12.tar.gz`
4. Create config.json file
   - Make sure to replace yoursite.com with your actual site domain

```
{
   "default_server_config": {
      "m.homeserver": {
         "base_url": "https://element.yoursite.com",
         "server_name": "yoursite.com"
      },
      "m.identity_server": {
         "base_url": "https://vector.im"
      }
   },
   "disable_custom_urls": true,
```

```
    "disable_guests": true,
    "disable_login_language_selector": false,
    "disable_3pid_login": true,
    "brand": "Element",
    "integrations_ui_url": "https://scalar.vector.im/",
    "integrations_rest_url": "https://scalar.vector.im/api",
    "integrations_widgets_urls": [
        "https://scalar.vector.im/_matrix/integrations/v1",
        "https://scalar.vector.im/api",
        "https://scalar-staging.vector.im/_matrix/integrations/v1",
        "https://scalar-staging.vector.im/api",
        "https://scalar-staging.riot.im/scalar/api"
    ],
    "bug_report_endpoint_url": "https://element.io/bugreports/submit",
    "uisi_autorageshake_app": "element-auto-uisi",
    "default_country_code": "GB",
    "show_labs_settings": false,
    "features": { },
    "default_federate": true,
    "default_theme": "dark",
    "room_directory": {
        "servers": [
            "yoursite.com"
        ]
    },
    "enable_presence_by_hs_url": {
        "https://yoursite.com": false,
        "https://element.yoursite.com": false
    },
    "setting_defaults": {
        "breadcrumbs": true
    },
    "jitsi": {
        "preferred_domain": "meet.element.io"
    },
    "map_style_url": "https://api.maptiler.com/maps/streets/style.json?key=iCksBQ1EHZmZ8OkgfZvV"
}
```

5. Add a DNS record for element.yoursite.com
6. Generate ZeroSSL Certs for element.yoursite.com
    1. `sudo nano /etc/nginx/sites-enabled/element.yoursite.com`

```
server {
    listen 80;
    listen [::]:80;
    server_name element.yoursite.com;
    location ~ "/.well-known/acme-challenge/" {
    default_type text/plain;
    return 200 "$1.YOUR_ACCOUNT_THUMBPRINT";
}
}
```

7. Install SSL certs
   - `sudo mkdir /etc/zerossl/live/element.yoursite.com/`
   - `./acme.sh --install-cert -d element.yoursite.com --cert-file /etc/zerossl/live/element.yoursite.com/cert.pem --key-file /etc/zerossl/live/element.yoursite.com/privkey.pem --fullchain-file /etc/zerossl/live/element.yoursite.com/fullchain.pem --ca-file /etc/zerossl/live/element.yoursite.com/chain.pem --reloadcmd "service nginx force-reload"`
8. Create Element Web Directory
   - `mkdir /var/wwww/element`
   - `sudo --user elementweb mv /home/elementweb/element /var/www/`
9. Update element nginx conf
   - `sudo nano /etc/nginx/sites-enabled/element.yoursite.com`

```
server {
    listen 80;
    listen [::]:80;
    server_name element.yoursite.com;

    return 301 https://$host$request_uri;

    access_log off;
    error_log off;
    server_tokens off;
}

server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name element.yoursite.com;

    ssl on;
    ssl_certificate /etc/zerossl/live/element.yoursite.com/fullchain.pem;
```

```
    ssl_certificate_key /etc/zerossl/live/element.yoursite.com/privkey.pem;

    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_prefer_server_ciphers on;
    ssl_ciphers "ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512:ECDHE-RSA-AES256-GCM-
SHA384:DHE-RSA-AES256-GCM-SHA384";
    ssl_session_timeout 1d;
    ssl_session_cache shared:MozSSL:10m;  # about 40000 sessions
    ssl_session_tickets off;

    gzip on;
    gzip_disable "msie6";
    gzip_vary on;
    gzip_proxied any;
    gzip_min_length 1024;
    gzip_comp_level 6;
    gzip_buffers 16 8k;
    gzip_http_version 1.1;
    gzip_types text/plain text/css application/json application/javascript text/xml application/xml
application/xml+rss text/javascript;

    access_log off;
    error_log off;
    server_tokens off;

    add_header X-Frame-Options SAMEORIGIN;
    add_header X-Content-Type-Options nosniff;
    add_header X-XSS-Protection "1; mode=block";
    add_header Content-Security-Policy "frame-ancestors 'none'";
    add_header Strict-Transport-Security "max-age=63072000" always;

    index index.html;
    root /var/www/element;

    location / {
        try_files $uri $uri/ =404;
        proxy_set_header X-Forwarded-For $remote_addr;
    }
}
```

10. Restart nginx
    - `sudo service nginx restart`

Automate Cert Renewals using DNS API

- NOTE: I used Njalla API in this demo, replace with your domain provider and follow instructions from this wiki.
- Issue the certs

1. `/root/acme.sh/./acme.sh --issue --dns dns_njalla -d yoursite.com --server zerossl && /root/acme.sh/./acme.sh --install-cert -d yoursite.com --cert-file /etc/zerossl/live/yoursite.com/cert.pem --key-file /etc/zerossl/live/yoursite.com/privkey.pem --fullchain-file /etc/zerossl/live/yoursite.com/fullchain.pem --ca-file /etc/zerossl/live/yoursite.com/chain.pem --reloadcmd "service nginx force-reload && cp /etc/zerossl/live/yoursite.com/fullchain.pem /etc/coturn/certs/ && cp /etc/zerossl/live/yoursite.com/privkey.pem /etc/coturn/certs/ && service coturn force-reload && service coturn restart"`
2. `/root/acme.sh/./acme.sh --issue --dns dns_njalla -d element.yoursite.com --server zerossl && /root/acme.sh/./acme.sh --install-cert -d element.yoursite.com --cert-file /etc/zerossl/live/element.yoursite.com/cert.pem --key-file /etc/zerossl/live/element.yoursite.com/privkey.pem --fullchain-file /etc/zerossl/live/element.yoursite.com/fullchain.pem --ca-file /etc/zerossl/live/element.yoursite.com/chain.pem --reloadcmd "service nginx force-reload"`
3. acme cron should now do the renewals automatically

Congrats! Enjoy your new server!