matrix-org/synapse

Originalartikel

Backup

<html> Contents Introduction About Matrix Synapse Installation
Installing from source Configuring synapse Registering a user Setting up a TURN server IPv6 Running Synapse Connecting to Synapse from a client
Registering a new user from a client Security Note Platform-Specific Instructions
Debian Fedora ArchLinux FreeBSD OpenBSD NixOS Windows Install Troubleshooting
<a href=„https://github.com/matrix-org/synapse/blob/master/README.rst#troubleshooting-installation“

id=„user-content-id29“>Troubleshooting Installation Troubleshooting Running
ArchLinux   Upgrading an existing Synapse Setting up Federation
Troubleshooting Running a Demo Federation of Synapses Using PostgreSQL Using a reverse proxy with Synapse
Reverse-proxying the federation port Identity Servers URL Previews Password reset Synapse Development Building Internal API Documentation Help!! Synapse eats all my RAM! Matrix is an ambitious new ecosystem for open federated Instant Messaging and VoIP. The basics you need to know to get up and running are: Everything in Matrix happens in a room. Rooms are distributed and do not exist on any single server. Rooms can be located using convenience aliases like

```
#matrix:matrix.org
```

or

```
#test:localhost:8448
```

. Matrix user IDs look like

```
@matthew:matrix.org
```

(although in the future you will normally refer to yourself and others using a third party identifier (3PID): email address, phone number, etc rather than manipulating Matrix user IDs) The overall architecture is: <pre>client <—-> homeserver <==========================> homeserver <—-> client

```
    https://somewhere.org/_matrix      https://elsewhere.net/_matrix
```

</pre>

```
#matrix:matrix.org
```

is the official support room for Matrix, and can be accessed by any client from https://matrix.org/docs/projects/try-matrix-now.html or via IRC bridge at irc://irc.freenode.net/matrix. Synapse is currently in rapid development, but as of version 0.5 we believe it is sufficiently stable to be run as an internet-facing service for real usage! Matrix specifies a set of pragmatic RESTful HTTP JSON APIs as an open standard, which handle: Creating and managing fully distributed chat rooms with no single points of control or failure Eventually-consistent cryptographically secure synchronisation of room state across a global open network of federated servers and services Sending and receiving extensible messages in a room with (optional) end-to-end encryption[1] Inviting, joining, leaving, kicking, banning room members Managing user accounts (registration, login, logout) Using 3rd Party IDs (3PIDs) such as email addresses, phone numbers, Facebook accounts to authenticate, identify and discover users on Matrix. Placing 1:1 VoIP and Video calls These APIs are intended to be implemented on a wide range of servers, services and clients, letting developers build messaging and VoIP functionality on top of the entirely open Matrix ecosystem rather than using closed or proprietary solutions. The hope is for Matrix to act as the building blocks for a new generation of fully open and interoperable messaging and VoIP apps for the internet. Synapse is a reference „homeserver“ implementation of Matrix from the core development team at matrix.org, written in Python/Twisted. It is intended to showcase the concept of Matrix and let folks see the spec in the context of a codebase and let you run your own homeserver and generally help bootstrap the ecosystem. In Matrix, every user runs one or more Matrix clients, which connect through to a Matrix homeserver. The homeserver stores all their personal chat history and user account information - much as a mail client connects through to an IMAP/SMTP server. Just like email, you can either run your own Matrix homeserver and control and own your own communications and history or use one hosted by someone else (e.g. matrix.org) - there is no single point of control or mandatory service provider in Matrix, unlike WhatsApp, Facebook, Hangouts, etc. We'd like to invite you to join #matrix:matrix.org (via https://matrix.org/docs/projects/try-matrix-now.html), run a homeserver, take a look at the Matrix spec, and experiment with the APIs and Client SDKs. Thanks for using Matrix! [1] End-to-end encryption is currently in beta: blog post. Synapse is the reference python/twisted Matrix homeserver implementation. System requirements: POSIX-compliant system (tested on Linux & OS X) Python 2.7 At least 1GB of free RAM if you want to join large public rooms like #matrix:matrix.org <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Installing from source</h3> (Prebuilt packages are available for some platforms - see Platform-Specific Instructions.) Synapse is written in python but some of

the libraries it uses are written in C. So before we can install synapse itself we need a working C compiler and the header files for python C extensions. Installing prerequisites on Ubuntu or Debian: <pre>sudo apt-get install build-essential python2.7-dev libffi-dev \

```
                    python-pip python-setuptools sqlite3 \
                    libssl-dev python-virtualenv libjpeg-dev libxslt1-dev
```

</pre> Installing prerequisites on ArchLinux: <pre>sudo pacman -S base-devel python2 python-pip \

```
               python-setuptools python-virtualenv sqlite3
```

</pre> Installing prerequisites on CentOS 7 or Fedora 25: <pre>sudo yum install libtiff-devel libjpeg-devel libzip-devel freetype-devel \

```
                 lcms2-devel libwebp-devel tcl-devel tk-devel redhat-rpm-
config \
                 python-virtualenv libffi-devel openssl-devel
```

sudo yum groupinstall „Development Tools“ </pre> Installing prerequisites on Mac OS X: <pre>xcode-select –install sudo easy_install pip sudo pip install virtualenv brew install pkg-config libffi </pre> Installing prerequisites on Raspbian: <pre>sudo apt-get install build-essential python2.7-dev libffi-dev \

```
                    python-pip python-setuptools sqlite3 \
                    libssl-dev python-virtualenv libjpeg-dev
```

sudo pip install –upgrade pip sudo pip install –upgrade ndg-httpsclient sudo pip install –upgrade virtualenv </pre> Installing prerequisites on openSUSE: <pre>sudo zypper in -t pattern devel_basis sudo zypper in python-pip python-setuptools sqlite3 python-virtualenv \

```
               python-devel libffi-devel libopenssl-devel libjpeg62-devel
```

</pre> Installing prerequisites on OpenBSD: <pre>doas pkg_add python libffi py-pip py-setuptools sqlite3 py-virtualenv \

```
             libxslt
```

</pre> To install the synapse homeserver run: <pre>virtualenv -p python2.7 ~/.synapse source ~/.synapse/bin/activate pip install –upgrade pip pip install –upgrade setuptools pip install https://github.com/matrix-org/synapse/tarball/master </pre> This installs synapse, along with the libraries it uses, into a virtual environment under

```
~/.synapse
```

. Feel free to pick a different directory if you prefer. In case of problems, please see the Troubleshooting section below. Alternatively, Silvio Fricke has contributed a Dockerfile to automate the above in Docker at https://registry.hub.docker.com/u/silviof/docker-matrix/. Also, Martin Giess has created an auto-deployment process with vagrant/ansible, tested with

VirtualBox/AWS/DigitalOcean - see https://github.com/EMnify/matrix-synapse-auto-deploy for details. <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Configuring synapse</h3> Before you can start Synapse, you will need to generate a configuration file. To do this, run (in your virtualenv, as before): <pre>cd ~/.synapse python -m synapse.app.homeserver \

1. -server-name my.domain.name \
2. -config-path homeserver.yaml \
3. -generate-config \
4. -report-stats=[yes|no]

</pre> ... substituting an appropriate value for

```
--server-name
```

. The server name determines the „domain“ part of user-ids for users on your server: these will all be of the format

```
@user:my.domain.name
```

. It also determines how other matrix servers will reach yours for Federation. For a test configuration, set this to the hostname of your server. For a more production-ready setup, you will probably want to specify your domain (

```
example.com
```

) rather than a matrix-specific hostname here (in the same way that your email address is probably

```
user@example.com
```

rather than

```
user@email.example.com
```

) - but doing so may require more advanced setup - see Setting up Federation. Beware that the server name cannot be changed later. This command will generate you a config file that you can then customise, but it will also generate a set of keys for you. These keys will allow your Home Server to identify itself to other Home Servers, so don't lose or delete them. It would be wise to back them up somewhere safe. (If, for whatever reason, you do need to change your Home Server's keys, you may find that other Home Servers have the old key cached. If you update the signing key, you should change the name of the key in the

```
&lt;server name&gt;.signing.key
```

file (the second word) to something different. See the spec for more information on key management.) The default configuration exposes two HTTP ports: 8008 and 8448. Port 8008 is configured without TLS; it is not recommended this be exposed outside your local network. Port 8448 is configured to use TLS with a self-signed certificate. This is fine for testing with but, to avoid your clients complaining about the certificate, you will almost certainly want to use another certificate for production purposes. (Note that a self-signed certificate is fine for Federation). You can do so by changing

```
tls_certificate_path
```

,

```
tls_private_key_path
```

and

```
tls_dh_params_path
```

in

```
homeserver.yaml
```

; alternatively, you can use a reverse-proxy, but be sure to read Using a reverse proxy with Synapse when doing so. Apart from port 8448 using TLS, both ports are the same in the default configuration. <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Registering a user</h3> You will need at least one user on your server in order to use a Matrix client. Users can be registered either via a Matrix client, or via a commandline script. To get started, it is easiest to use the command line to register new users: <pre>$ source ~/.synapse/bin/activate $ synctl start # if not already running $ register_new_matrix_user -c homeserver.yaml https://localhost:8448 New user localpart: erikj Password: Confirm password: Make admin [no]: Success! </pre> This process uses a setting

```
registration_shared_secret
```

in

```
homeserver.yaml
```

, which is shared between Synapse itself and the

```
register_new_matrix_user
```

script. It doesn't matter what it is (a random value is generated by

```
--generate-config
```

), but it should be kept secret, as anyone with knowledge of it can register users on your server even if

```
enable_registration
```

is

```
false
```

. <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Setting up a TURN server</h3> For reliable VoIP calls to be routed via this homeserver, you MUST configure a TURN server. See docs/turn-howto.rst for details. <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>IPv6</h3> As of Synapse 0.19 we finally support IPv6, many thanks to @kyrias and @glyph for providing PR #1696. However, for federation to work on hosts with IPv6 DNS servers you must be running Twisted 17.1.0 or later - see https://github.com/matrix-org/synapse/issues/1002 for details. We can't make Synapse depend on Twisted 17.1 by default yet as it will break most older distributions (see https://github.com/matrix-org/synapse/pull/1909) so if you are using operating system dependencies you'll have to install your own Twisted 17.1 package via pip or backports etc. If you're running in a virtualenv then pip should have installed the newest Twisted automatically, but if your virtualenv is old you will need to manually upgrade to a newer Twisted dependency via: <blockquote> pip install Twisted>=17.1.0</blockquote> To actually run your new homeserver, pick a working directory for Synapse to run (e.g.

```
~/.synapse
```

), and: <pre>cd ~/.synapse source ./bin/activate synctl start </pre> The easiest way to try out your new Synapse installation is by connecting to it from a web client. The easiest option is probably the one at http://riot.im/app. You will need to specify a „Custom server“ when you log on or register: set this to

```
https://localhost:8448
```

- remember to specify the port (

```
:8448
```

) unless you changed the configuration. (Leave the identity server as the default - see Identity servers.) If all goes well you should at least be able to log in, create a room, and start sending messages. (The homeserver runs a web client by default at https://localhost:8448/, though as of the time of writing it is somewhat outdated and not really recommended - https://github.com/matrix-org/synapse/issues/1527). <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Registering a new user from a client</h3> By default, registration of new users via Matrix clients is disabled. To enable it, specify

```
enable_registration: true
```

in

```
homeserver.yaml
```

. (It is then recommended to also set up CAPTCHA - see docs/CAPTCHA_SETUP.rst.) Once

```
enable_registration
```

is set to

```
true
```

, it is possible to register a user via riot.im or other Matrix clients. Your new user name will be formed partly from the

```
server_name
```

(see Configuring synapse), and partly from a localpart you specify when you create the account. Your name will take the form of: <pre>@localpart:my.domain.name </pre> (pronounced „at localpart on my dot domain dot name“). As when logging in, you will need to specify a „Custom server“. Specify your desired

```
localpart
```

in the 'User name' box. Matrix serves raw user generated data in some APIs - specifically the content repository endpoints. Whilst we have tried to mitigate against

possible XSS attacks (e.g. https://github.com/matrix-org/synapse/pull/1021) we recommend running matrix homeservers on a dedicated domain name, to limit any malicious user generated content served to web browsers a matrix API from being able to attack webapps hosted on the same domain. This is particularly true of sharing a matrix webclient and server on the same domain. See https://github.com/vector-im/vector-web/issues/1977 and https://developer.github.com/changes/2014-04-25-user-content-security for more details. <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Debian</h3> Matrix provides official Debian packages via apt from http://matrix.org/packages/debian/. Note that these packages do not include a client - choose one from https://matrix.org/docs/projects/try-matrix-now.html (or build your own with one of our SDKs :) <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Fedora</h3> Oleg Girko provides Fedora RPMs at https://obs.infoserver.lv/project/monitor/matrix-synapse <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>ArchLinux</h3> The quickest way to get up and running with ArchLinux is probably with the community package https://www.archlinux.org/packages/community/any/matrix-synapse/, which should pull in most of the necessary dependencies. If the default web client is to be served (enabled by default in the generated config), https://www.archlinux.org/packages/community/any/python2-matrix-angular-sdk/ will also need to be installed. Alternatively, to install using pip a few changes may be needed as ArchLinux defaults to python 3, but synapse currently assumes python 2.7 by default: pip may be outdated (6.0.7-1 and needs to be upgraded to 6.0.8-1): <pre>sudo pip2.7 install –upgrade pip </pre> You also may need to explicitly specify python 2.7 again during the install request: <pre>pip2.7 install https://github.com/matrix-org/synapse/tarball/master </pre> If you encounter an error with lib bcrypt causing an Wrong ELF Class: ELFCLASS32 (x64 Systems), you may need to reinstall py-bcrypt to correctly compile it under the right architecture. (This should not be needed if installing under virtualenv): <pre>sudo pip2.7 uninstall py-bcrypt sudo pip2.7 install py-bcrypt </pre> During setup of Synapse you need to call python2.7 directly again: <pre>cd ~/.synapse python2.7 -m synapse.app.homeserver \

1. -server-name machine.my.domain.name \

2. -config-path homeserver.yaml \
3. -generate-config

</pre> …substituting your host and domain name as appropriate. <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>FreeBSD</h3> Synapse can be installed via FreeBSD Ports or Packages contributed by Brendan Molloy from: <blockquote> Ports:

```
cd /usr/ports/net-im/py-matrix-synapse &amp;&amp; make install clean
```

 Packages:

```
pkg install py27-matrix-synapse
```

 </blockquote> <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>OpenBSD</h3> There is currently no port for OpenBSD. Additionally, OpenBSD's security settings require a slightly more difficult installation process. Create a new directory in

```
/usr/local
```

called

```
_synapse
```

. Also, create a new user called

```
_synapse
```

and set that directory as the new user's home. This is required because, by default, OpenBSD only allows binaries which need write and execute permissions on the same memory space to be run from

```
/usr/local
```

.

```
su
```

to the new

```
_synapse
```

user and change to their home directory. Create a new virtualenv:

```
virtualenv -p python2.7 ~/.synapse
```

 Source the virtualenv configuration located at

```
/usr/local/_synapse/.synapse/bin/activate
```

. This is done in

```
ksh
```

by using the

```
.
```

command, rather than

```
bash
```

's

```
source
```

. Optionally, use

```
pip
```

to install

```
lxml
```

, which Synapse needs to parse webpages for their titles. Use

```
pip
```

to install this repository:

```
pip install
https://github.com/matrix-org/synapse/tarball/master
```

 Optionally, change

```
_synapse
```

's shell to

```
/bin/false
```

to reduce the chance of a compromised Synapse server being used to take over your box. After this, you may proceed with the rest of the install directions. <h3><a id=„user-

content-nixos“ class=„anchor“ href=„https://github.com/matrix-org/synapse/blob/master/README.rst#nixos“ aria-hidden=„true“><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>NixOS</h3> Robin Lambertz has packaged Synapse for NixOS at: https://github.com/NixOS/nixpkgs/blob/master/nixos/modules/services/misc/matrix-synapse.nix <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Windows Install</h3> Synapse can be installed on Cygwin. It requires the following Cygwin packages: gcc git libffi-devel openssl (and openssl-devel, python-openssl) python python-setuptools The content repository requires additional packages and will be unable to process uploads without them: libjpeg8 libjpeg8-devel zlib If you choose to install Synapse without these packages, you will need to reinstall

```
pillow
```

for changes to be applied, e.g.

```
pip uninstall pillow
```

```
pip install
pillow --user
```

 Troubleshooting: You may need to upgrade

```
setuptools
```

to get this to work correctly:

```
pip install setuptools --upgrade
```

. You may encounter errors indicating that

```
ffi.h
```

is missing, even with

```
libffi-devel
```

installed. If you do, copy the

```
.h
```

files:

```
cp /usr/lib/libffi-3.0.13/include/*.h /usr/include
```

 You may need to install libsodium from source in order to install PyNacl. If you do, you may need to create a symlink to

```
libsodium.a
```

so

```
ld
```

can find it:

```
ln -s /usr/local/lib/libsodium.a /usr/lib/libsodium.a
```

  <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Troubleshooting Installation</h3> Synapse requires pip 1.7 or later, so if your OS provides too old a version you may need to manually upgrade it: <pre>sudo pip install –upgrade pip </pre> Installing may fail with

```
Could not find any downloads that satisfy the requirement pymacaroons-pynacl
(from matrix-synapse==0.12.0)
```

. You can fix this by manually upgrading pip and virtualenv: <pre>sudo pip install –upgrade virtualenv </pre> You can next rerun

```
virtualenv -p python2.7 synapse
```

to update the virtual env. Installing may fail during installing virtualenv with

```
InsecurePlatformWarning: A true SSLContext object is not available. This
prevents urllib3 from configuring SSL appropriately and may cause certain
SSL connections to fail. For more information, see
https://urllib3.readthedocs.org/en/latest/security.html#insecureplatformwarn
ing.
```

You can fix this by manually installing ndg-httpsclient: <pre>pip install –upgrade ndg-httpsclient </pre> Installing may fail with

```
mock requires setuptools&gt;=17.1. Aborting installation
```

. You can fix this by upgrading setuptools: <pre>pip install –upgrade setuptools </pre> If pip crashes mid-installation for reason (e.g. lost terminal), pip may refuse to run until you remove the temporary installation directory it created. To reset the installation: <pre>rm -rf /tmp/pip_install_matrix </pre> pip seems to leak lots of memory during installation. For instance, a Linux host with 512MB of RAM may run out of memory whilst installing Twisted. If this

happens, you will have to individually install the dependencies which are failing, e.g.: <pre>pip install twisted </pre> On OS X, if you encounter clang: error: unknown argument: '-mno-fused-madd' you will need to export CFLAGS=-Qunused-arguments. <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Troubleshooting Running</h3> If synapse fails with

```
missing "sodium.h"
```

crypto errors, you may need to manually upgrade PyNaCL, as synapse uses NaCl (http://nacl.cr.yp.to/) for encryption and digital signatures. Unfortunately PyNACL currently has a few issues (https://github.com/pyca/pynacl/issues/53) and (https://github.com/pyca/pynacl/issues/79) that mean it may not install correctly, causing all tests to fail with errors about missing „sodium.h“. To fix try re-installing from PyPI or directly from (https://github.com/pyca/pynacl): <pre># Install from PyPI pip install –user –upgrade –force pynacl # Install from github pip install –user https://github.com/pyca/pynacl/tarball/master </pre> <h4><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>ArchLinux</h4> If running $ synctl start fails with 'returned non-zero exit status 1', you will need to explicitly call Python2.7 - either running as: <pre>python2.7 -m synapse.app.homeserver –daemonize -c homeserver.yaml </pre> …or by editing synctl with the correct python executable. The instructions for upgrading synapse are in UPGRADE.rst. Please check these instructions as upgrading may require extra steps for some versions of synapse. Federation is the process by which users on different servers can participate in the same room. For this to work, those other servers must be able to contact yours to send messages. As explained in Configuring synapse, the

```
server_name
```

in your

```
homeserver.yaml
```

file determines the way that other servers will reach yours. By default, they will treat it as a hostname and try to connect to port 8448. This is easy to set up and will work with the default configuration, provided you set the

```
server_name
```

to match your machine's public DNS hostname. For a more flexible configuration, you can set up a DNS SRV record. This allows you to run your server on a machine that might not have the same name as your domain name. For example, you might want to run your server at

```
synapse.example.com
```

, but have your Matrix user-ids look like

```
@user:example.com
```

. (A SRV record also allows you to change the port from the default 8448. However, if you are thinking of using a reverse-proxy, be sure to read Reverse-proxying the federation port first.) To use a SRV record, first create your SRV record and publish it in DNS. This should have the format

```
_matrix._tcp.&lt;yourdomain.com&gt; &lt;ttl&gt; IN SRV 10 0 &lt;port&gt; &lt;synapse.server.name&gt;
```

. The DNS record should then look something like: <pre>$ dig -t srv _matrix._tcp.example.com _matrix._tcp.example.com. 3600 IN SRV 10 0 8448 synapse.example.com. </pre> You can then configure your homeserver to use

```
&lt;yourdomain.com&gt;
```

as the domain in its user-ids, by setting

```
server_name
```

: <pre>python -m synapse.app.homeserver \

1. -server-name <yourdomain.com> \
2. -config-path homeserver.yaml \
3. -generate-config

python -m synapse.app.homeserver –config-path homeserver.yaml </pre> If you've already generated the config file, you need to edit the

```
server_name
```

in your

```
homeserver.yaml
```

file. If you've already started Synapse and a database has been created, you will have to recreate the database. If all goes well, you should be able to connect to your server with a client, and then join a room via federation. (Try

```
#matrix-dev:matrix.org
```

as a first step. „Matrix HQ“'s sheer size and activity level tends to make even the largest boxes pause for thought.) <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Troubleshooting</h3> The typical failure mode with federation is that when you try to join a room, it is rejected with „401: Unauthorized“. Generally this means that other servers in the room couldn't access yours. (Joining a room over federation is a complicated dance which requires connections in both directions). So, things to check are: If you are trying to use a reverse-proxy, read Reverse-proxying the federation port. If you are not using a SRV record, check that your

```
server_name
```

(the part of your user-id after the

```
:
```

) matches your hostname, and that port 8448 on that hostname is reachable from outside your network. If you are using a SRV record, check that it matches your

```
server_name
```

(it should be

```
_matrix._tcp.&lt;server_name&gt;
```

), and that the port and hostname it specifies are reachable from outside your network. <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Running a Demo Federation of Synapses</h3> If you want to get up and running quickly with a trio of homeservers in a private federation, there is a script in the

```
demo
```

directory. This is mainly useful just for development purposes. See demo/README. As of Synapse 0.9, PostgreSQL is supported as an alternative to the SQLite database that Synapse has traditionally used for convenience and simplicity. The advantages of Postgres include: significant performance improvements due to the superior threading and caching model, smarter query optimiser allowing the DB to be run on separate hardware allowing basic active/backup high-availability with a „hot spare“ synapse pointing at the same DB master, as well as enabling DB replication in synapse itself. For information on how to install and

use PostgreSQL, please see docs/postgres.rst. It is possible to put a reverse proxy such as nginx, Apache or HAProxy in front of Synapse. One advantage of doing so is that it means that you can expose the default https port (443) to Matrix clients without needing to run Synapse with root privileges. The most important thing to know here is that Matrix clients and other Matrix servers do not necessarily need to connect to your server via the same port. Indeed, clients will use port 443 by default, whereas servers default to port 8448. Where these are different, we refer to the 'client port' and the 'federation port'. The next most important thing to know is that using a reverse-proxy on the federation port has a number of pitfalls. It is possible, but be sure to read Reverse-proxying the federation port. The recommended setup is therefore to configure your reverse-proxy on port 443 for client connections, but to also expose port 8448 for server-server connections. All the Matrix endpoints begin

```
/_matrix
```

, so an example nginx configuration might look like: <pre>server {

```
  listen 443 ssl;
  listen [::]:443 ssl;
  server_name matrix.example.com;
  location /_matrix {
      proxy_pass http://localhost:8008;
      proxy_set_header X-Forwarded-For $remote_addr;
  }
```

} </pre> You will also want to set

```
bind_addresses: ['127.0.0.1']
```

and

```
x_forwarded: true
```

for port 8008 in

```
homeserver.yaml
```

to ensure that client IP addresses are recorded correctly. Having done so, you can then use

```
https://matrix.example.com
```

(instead of

```
https://matrix.example.com:8448
```

) as the „Custom server“ when Connecting to Synapse from a client. <h3><svg aria-hidden=„true“ class=„octicon octicon-link“ height=„16“ version=„1.1“ viewbox=„0 0 16 16“ width=„16“/>Reverse-proxying the federation port</h3> There are two issues to consider before using a reverse-proxy on the federation port: Due to the way SSL certificates are managed in the Matrix federation protocol (see spec), Synapse needs to be configured with the path to the SSL certificate, even if you do not terminate SSL at Synapse. Synapse does not currently support SNI on the federation protocol (bug #1491), which means that using name-based virtual hosting is unreliable. Furthermore, a number of the normal reasons for using a reverse-proxy do not apply: Other servers will connect on port 8448 by default, so there is no need to listen on port 443 (for federation, at least), which avoids the need for root privileges and virtual hosting. A self-signed SSL certificate is fine for federation, so there is no need to automate renewals. (The certificate generated by

```
--generate-config
```

is valid for 10 years.) If you want to set up a reverse-proxy on the federation port despite these caveats, you will need to do the following: In

```
homeserver.yaml
```

, set

```
tls_certificate_path
```

to the path to the SSL certificate file used by your reverse-proxy, and set

```
no_tls
```

to

```
True
```

. (

```
tls_private_key_path
```

will be ignored if

```
no_tls
```

is

```
True
```

.) In your reverse-proxy configuration:
If there are other virtual hosts on the same port, make sure that the default one uses the certificate configured above. Forward

```
/_matrix
```

to Synapse. If your reverse-proxy is not listening on port 8448, publish a SRV record to tell other servers how to find you. See Setting up Federation. When updating the SSL certificate, just update the file pointed to by

```
tls_certificate_path
```

: there is no need to restart synapse. (You may like to use a symbolic link to help make this process atomic.) The most common mistake when setting up federation is not to tell Synapse about your SSL certificate. To check it, you can visit

```
https://matrix.org/federationtester/api/report?server_name=&lt;your_server_name&gt;
```

. Unfortunately, there is no UI for this yet, but, you should see

```
"MatchingTLSFingerprint": true
```

. If not, check that

```
Certificates[0].SHA256Fingerprint
```

(the fingerprint of the certificate presented by your reverse-proxy) matches

```
Keys.tls_fingerprints[0].sha256
```

(the fingerprint of the certificate Synapse is using). Identity servers have the job of mapping email addresses and other 3rd Party IDs (3PIDs) to Matrix user IDs, as well as verifying the ownership of 3PIDs before creating that mapping. They are not where accounts or credentials are stored - these live on home servers. Identity Servers are just for mapping 3rd party IDs to matrix IDs. This process is very security-sensitive, as there is obvious risk of spam if it is too easy to sign up for Matrix accounts or harvest 3PID data. In the longer term, we hope to create a decentralised system to manage it (matrix-doc #712), but in the meantime, the role of managing trusted identity in the Matrix ecosystem is farmed out to a cluster of known trusted ecosystem partners, who run 'Matrix Identity Servers' such as Sydent, whose role is purely to authenticate and track 3PID logins and publish end-user public keys. You can host your own copy of Sydent, but this will prevent you reaching other users in the Matrix ecosystem via their email address, and prevent them finding you. We therefore recommend that you use one of the centralised identity servers at

```
https://matrix.org
```

or

```
https://vector.im
```

for now. To reiterate: the Identity server will only be used if you choose to associate an email address with your account, or send an invite to another user via their email address. Synapse 0.15.0 introduces a new API for previewing URLs at

```
/_matrix/media/r0/preview_url
```

. This is disabled by default. To turn it on you must enable the

```
url_preview_enabled: True
```

config parameter and explicitly specify the IP ranges that Synapse is not allowed to spider for previewing in the

```
url_preview_ip_range_blacklist
```

configuration parameter. This is critical from a security perspective to stop arbitrary Matrix users spidering 'internal' URLs on your network. At the very least we recommend that your loopback and RFC1918 IP addresses are blacklisted. This also requires the optional lxml and netaddr python dependencies to be installed. If a user has registered an email address to their account using an identity server, they can request a password-reset token via clients such as Vector. A manual password reset can be done via direct database access as follows. First calculate the hash of the new password: <pre>$ source ~/.synapse/bin/activate $ ./scripts/hash_password Password: Confirm password: $2a$12$xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx </pre> Then update the users table in the database: <pre>UPDATE users SET password_hash='$2a$12$xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'

```
  WHERE name='@test:test.com';
```

</pre> Before setting up a development environment for synapse, make sure you have the system dependencies (such as the python header files) installed - see Installing from source. To check out a synapse for development, clone the git repo into a working directory of your choice: <pre>git clone https://github.com/matrix-org/synapse.git cd synapse </pre> Synapse has a number of external dependencies, that are easiest to install using pip and a virtualenv: <pre>virtualenv -p python2.7 env source env/bin/activate python synapse/python_dependencies.py | xargs pip install pip install lxml mock </pre> This will run a process of downloading and installing all the needed dependencies into a virtual env. Once this is done, you may wish to run Synapse's unit tests, to check that everything is installed as it should be: <pre>PYTHONPATH=„.“ trial tests </pre> This should end with a 'PASSED' result: <pre>Ran 143 tests in 0.601s PASSED (successes=143) </pre> Before building internal API documentation install sphinx and sphinxcontrib-napoleon: <pre>pip install sphinx pip install sphinxcontrib-napoleon </pre> Building internal API documentation: <pre>python setup.py build_sphinx </pre> Synapse's architecture is quite RAM hungry currently - we deliberately cache a lot of recent room

data and metadata in RAM in order to speed up common requests. We'll improve this in future, but for now the easiest way to either reduce the RAM usage (at the risk of slowing things down) is to set the almost-undocumented

```
SYNAPSE_CACHE_FACTOR
```

environment variable. Roughly speaking, a SYNAPSE_CACHE_FACTOR of 1.0 will max out at around 3-4GB of resident memory - this is what we currently run the matrix.org on. The default setting is currently 0.1, which is probably around a ~700MB footprint. You can dial it down further to 0.02 if desired, which targets roughly ~512MB. Conversely you can dial it up if you need performance for lots of users and have a box with a lot of RAM.</p> </html>