Configuring Components

- Configuring Synapse
- Configuring Matrix Authentication Service
- Configuring Authentication
- Configuring Element Web
- Configuring Matrix RTC

Configuring Synapse

See how to download example files from the helm chart here.

Configuration

For a quick setup using the default settings, see the minimal fragment example in `charts/matrix-stack/ci/fragments/synapse-minimal.yaml`.

Configuring a postgresql database

If you want to use an external postgresql, see the following fragments examples:

- `charts/matrix-stack/ci/fragments/synapse-postgres.yaml`
- `charts/matrix-stack/ci/fragments/synapse-postgres-secrets-in-helm.yaml` or `charts/matrix-stack/ci/fragments/synapse-postgres-secrets-externally.yaml`

Credentials

Credentials are generated if possible. Alternatively they can either be provided inline in the values with `value` or if you have an existing `Secret` in the cluster in the same namespace you can use `secret` and `secretKey` to reference it.

If you dont want the chart to generate the secret, please refer to the following values fragments examples to see the secrets to configure.

Synapse requires `registrationSharedSecret`, `signingKey` and `macaroon` secrets:

- `charts/matrix-stack/ci/fragments/synapse-secrets-in-helm.yaml`
- `charts/matrix-stack/ci/fragments/synapse-secrets-externally.yaml`

If you are configuring S3 storage, see the following values fragments examples to see the secrets to configure:

- `charts/matrix-stack/ci/fragments/synapse-s3-secrets-in-helm.yaml`
- `charts/matrix-stack/ci/fragments/synapse-s3-secrets-externally.yaml`

Additional configuration

Additional Synapse configuration can be provided inline in the values as a string with

```
synapse:
  additional:
    ## Either reference config to inject by:
    1-custom-config:
      config: |
        admin_contact: "mailto:admin@example.com"
    ## Either reference an existing `Secret` by:
    2-custom-config:
      configSecret: custom-synapse-config
      configSecretKey: shared.yaml
```

Workers

The following Synapse workers are disabled by default and can be enabled on a per-worker basis :

- `appservice`
- `background`
- `client-reader`
- `encryption`
- `event-creator`
- `event-persister`
- `federation-sender`
- `initial-synchrotron`
- `media-repository`
- `presence-writer`
- `pusher`
- `receipts-account`
- `sliding-sync`
- `sso-login`
- `synchrotron`
- `typing-persister`
- `user-dir`

Synapse workers can be configured in the values with:

```
synapse:
  workers:
    <worker name>:
      enabled: true
```

Each worker comes with a different options (static replicas, horizontal scaling, resources, etc). These options can be seen under `synapse.workers.<name>` section of `helm show values` for this chart.

The following Synapse pro workers are enabled by default :

- `federation-reader`

They can be disabled in the values with :

```
synapse:
  workers:
    <worker name>:
      enabled: false
```

Full details on available configuration options can be found at https://element-hq.github.io/synapse/latest/usage/configuration/config_documentation.html

Disabling Synapse

Synapse is enabled for deployment by default can be disabled with the following values

```
synapse:
  enabled: false
```

Configuring Matrix Authentication Service

See how to download example files from the helm chart here.

Configuration

For a quick setup using the default settings, see the minimal fragment example in `charts/matrix-stack/ci/fragments/matrix-authentication-service-minimal.yaml`.

Using Element Web ingress

If Element Web is deployed, you can use the ingress host to access the Matrix Authentication Service. To do so, you can skip configuring `matrixAuthenticationService.ingress`. The chart will automatically expose the Matrix Authentication Service on the same ingress as Element Web, under the path `/account`.

Configuring a postgresql database

If you want to use an external postgresql database, merge 2 files to `postgresql.yaml`:

- `charts/matrix-stack/ci/fragments/matrix-authentication-service-postgres.yaml`
- `charts/matrix-stack/ci/fragments/matrix-authentication-service-postgres-secrets-in-helm.yaml` or `charts/matrix-stack/ci/fragments/matrix-authentication-service-postgres-secrets-externally.yaml`

Credentials

Credentials are generated if possible. Alternatively they can either be provided inline in the values with `value` or if you have an existing `Secret` in the cluster in the same namespace you can use `secret` and `secretKey` to reference it.

If you dont want the chart to generate the secret, please refer to the following values fragments examples to see the secrets to configure.

Matrix Authentication Service requires `encryptionSecret`, `synapseSharedSecret` and `synapseOIDCClientSecret` secrets:

- `charts/matrix-stack/ci/fragments/matrix-authentication-service-secrets-in-helm.yaml`
- `charts/matrix-stack/ci/fragments/matrix-authentication-service-secrets-externally.yaml`

If you are using LDAP Authentication, this will also need to configure `dex.masClientSecret`.

Additional configuration

Additional Matrix Authentication Service configuration can be provided inline in the values as a string with

```
matrixAuthenticationService:
  additional:
    ## Either reference config to inject by:
    1-custom-config:
      config: |
        admin_contact: "mailto:admin@example.com"
    ## Either reference an existing `Secret` by:
    2-custom-config:
      configSecret: custom-matrix-authentication-service-config
      configSecretKey: shared.yaml
```

Disabling Matrix Authentication Service

Matrix Authentication Service is enabled for deployment by default can be disabled with the following values

```
matrixAuthenticationService:
  enabled: false
```

Enable user registration

To allow users registration, you will need to configure MAS with SMTP. To do so, follow the steps in Configuring Matrix Authentication Service to inject additional email configuration.

Here is a sample minimal MAS configuration that allows user registration. You are encouraged to look through the MAS documentation linked above and customise the options to your requirements.

```
matrixAuthenticationService:
  additional:
    user-config.yaml:
      config: |
        email:
          from: '"Company Ink" <noreply@example.com>'
          reply_to: '"Company Ink" <noreply@example.com>'
          transport: smtp
          mode: starttls
          hostname: "smtp.example.com"
          port: 587
          username: smtpuser
          password: secretsmtppassword
        account:
          password_registration_enabled: true
          password_recovery_enabled: true
          account_deactivation_allowed: true
          login_with_email_allowed: true
        policy:
          data:
            emails:
              allowed_addresses:
                suffixes: ["@example.com"]
        rate_limiting:
          account_recovery:
            per_ip:
              burst: 3
              per_second: 0.0008
            per_address:
              burst: 3
              per_second: 0.0002
          login:
            per_ip:
              burst: 3
              per_second: 0.05
            per_account:
```

```
            burst: 1800
            per_second: 0.5
          registration:
            burst: 3
            per_second: 0.0008
```

Enable the MAS Admin API

To enable the MAS Admin API, you need to add some additional MAS configuration. There are two modes to use the Admin API. You can enable either one on its own or both as per your requirements. Note you will need to generate valid ULIDs for the client IDs below using a ULID generator like for example https://ulidgenerator.com/

1. Using the Swagger UI provided with MAS. An example is available on the MAS documentation page at https://element-hq.github.io/matrix-authentication-service/api/index.html. However, we encourage you to instead use the one hosted by your MAS instance at `https://your-mas-domain.tld/api/doc/`. `ULID_Admin_Client_1` in the below example enables authentication for graphical MAS clients like the Swagger UI.
2. Manually calling the API using a rest client, for example cURL or Bruno. This is documented in this example in the MAS documentation. This is `ULID_Admin_Client_2` in the below example.

> Ensure you protect the Client IDs and Secrets as these grant full access to manage all accounts on your server.

Example configuration:

```
matrixAuthenticationService:
  additional:
    user-config.yaml:
      config: |
        policy:
          data:
            admin_clients:
              - ULID_Admin_Client_1
              - ULID_Admin_Client_2
            admin_users:
              - your-admin-user
        clients:
          - client_id: ULID_Admin_Client_1
```

```
      client_auth_method: client_secret_post
      client_secret: A-secret
      redirect_uris:
        - https://account.example.com/api/doc/oauth2-callback
    - client_id: ULID_Admin_Client_2
      client_auth_method: client_secret_basic
      client_secret: Another-secret
```

Synapse Admin API

Access tokens returned by the above MAS Admin API configuration cannot be used with the Synapse Admin API. Long term, we plan to implement Personal Access Tokens in MAS. However, until that feature has landed, the only way to get an access token for the Synpse Admin API is using `mas-cli`.

```
kubectl exec --container matrix-authentication-service --namespace ess \
    --stdin --tty deploy/ess-matrix-authentication-service \
    -- mas-cli manage issue-compatibility-token \
    --yes-i-want-to-grant-synapse-admin-privileges \
    your-username
```

This will return a response similar to this

```
2025-05-21T11:11:53.564226Z  INFO mas_cli::commands::manage:320 Compatibility
token issued: mct_secret compat_access_token.id=ZI1UZZKCNWFOBFUUOQEYZBSIU8
compat_session.id=9X1BFZGXOYXGG5MDHPODT3ER6Q compat_session.device=MI71UWHZLG
user.id=QZEMHAYQCYXS8AYYQ3QWTRMNJZ user.username=your-username
```

In this example, `mct_secret` is your admin access token.

> Ensure you protect the access token as this grants full access to manage your server.

Configuring Authentication

See how to download example files from the helm chart here.

Overview

The chart components authentication is managed by the top-level key `authentication`.

The configuration is similar if you use standalone Synapse (legacy authentication) or if you enable Matrix Authentication Service.

You can find configurations examples in `charts/matrix-stack/ci/fragments/authentication-secrets-externally.yaml` and `charts/matrix-stack/ci/fragments/authentication-secrets-in-helm.yaml`.

Registration and Password Authentication

The charts come with :

- registration disabled by default
- password authentication enabled by default

To change this default behaviour, you will have to configure it through the `synapse.additional` or `matrixAuthenticationService.additional` key. See Synapse documentation or Matrix Authentication Service documentation for more details.

Configuring OIDC

You can configure a list of OIDC providers to use in the chart. Please refer to the description of the `authentication.oidc` key in the values file for details.

Configuring LDAP

You can configure a list of LDAP providers to use in the chart. Please refer to the description of the `authentication.ldap` key in the values file for details.

If LDAP is configured, and Advanced Identity Management is enabled, it will use the first LDAP provider configured in the list as the source of its users.

Configuring Element Web

See how to download example files from the helm chart here.

Configuration

For a quick setup using the default settings, see the minimal fragment example in `charts/matrix-stack/ci/fragments/element-web-minimal.yaml`.

Additional configuration

Additional Element Web configuration can be provided inline in the values as a json string with

```
elementWeb:
  additional:
    user-config.json: |
      {
        "some": "settings"
      }
```

Disabling Element Web

Matrix Authentication Service is enabled for deployment by default can be disabled with the following values

```
elementWeb:
  enabled: false
```

Configuring Matrix RTC

See how to download example files from the helm chart here.

Configuration

For a quick setup using the default settings, see the minimal fragment example in `charts/matrix-stack/ci/fragments/matrix-rtc-minimal.yaml`.

Credentials

Credentials are generated if possible. Alternatively they can either be provided inline in the values with `value` or if you have an existing `Secret` in the cluster in the same namespace you can use `secret` and `secretKey` to reference it.

If you dont want the chart to generate the secret, please refer to the following values fragments examples to see the secrets to configure.

Matrix RTC requires `livekitAuth.secret` secret:

- `charts/matrix-stack/ci/fragments/matrix-rtc-secrets-in-helm.yaml`
- `charts/matrix-stack/ci/fragments/matrix-rtc-secrets-externally.yaml`

SFU Networking

The matrix RTC SFU networking relies on NodePort by default. This means that the node but be reachable from outside of the cluster. Default ports are :

- RTC TCP: 30000/TCP
- RTC Muxed UDP : 30001/UDP

This can be configured using `matrixRTC.sfu.exposedServices`.

The default SFU networking relies on STUN to discover its public IP. It will automatically advertise it to the clients. The STUN servers can be configured in LiveKit configuration using the `additional` section :

```
matrixRTC:
  sfu:
    additional: |
      rtc:
        stun_servers:
          - ip:port
          - ip:port
          - ...
```

Accessing from behind a Load Balancer

If you are behind a Load Balancer, you must forward the ports from the Load Balancer to the nodes. The ports must be the same on the Load Balancer and the nodes. In this situation, the SFU cannot discover the Load Balancer public IP using the STUN method. Instead, you must use set the env variable `NODE_IP` :

```
matrixRTC:
  sfu:
    extraEnv:
    - name: NODE_IP
      value: 1.2.3.4

    additional: |
      rtc:
        use_external_ip: false
        # To workaround https://github.com/livekit/livekit/issues/2088
        # Any IP address is acceptable, it doesn't need to be a correct one,
        # it just needs to be present to get LiveKit to skip checking all local interfaces
        # We assign here a TEST-NET IP which is
        # overridden by the NODE_IP env var at runtime
        node_ip: 198.51.100.1
```

Additional SFU configuration

Additional Matrix RTC SFU configuration can be provided inline in the values as a string with

```
matrixRTC:
  sfu:
    additional:
      ## Either reference config to inject by:
```

```
      1-custom-config:
        config: |
          admin_contact: "mailto:admin@example.com"
      ## Either reference an existing `Secret` by:
      2-custom-config:
        configSecret: custom-matrix-rtc-config
        configSecretKey: shared.yaml
```

Disabling Matrix RTC

Matrix RTC is enabled for deployment by default can be disabled with the following values

```
matrixRTC:
  enabled: false
```