Element On-Premise Documentation

- Introduction to Element Enterprise
- Kubernetes Installations
- Single Node Installations
- Using the Installer in an Air-Gapped Environment
- Setting Up Jitsi and TURN With the Installer
- Setting up Permalinks With the Installer
- Setting Up Well Known Delegation
- Setting up Delegated Authentication With the Installer
- Setting up Group Sync with the Installer
- Setting Up the Integration Manager With the Installer
- Setting up GitLab, GitHub, JIRA and Webhooks Integrations With the Installer
- Setting up Adminbot and Auditbot
- Setting Up the Enterprise Admin Dashboard
- Setting Up Hydrogen
- Setting up On-Premise Metrics
- Setting Up the Telegram Bridge
- Setting Up the Teams Bridge
- Setting Up the IRC Bridge
- Setting Up the SIP Bridge
- Setting Up the XMPP Bridge

- Single Node Installs: Storage and Backup Guidelines
- On-Premise Support Scope of Coverage
- Troubleshooting
- Archived Documentation Repository
  - Documentation Covering Installers From 2022.07.03 to 2022.09.05
  - Documentation covering v1 and installers prior to 2022-07.03

Introduction to Element Enterprise

What is Element Enterprise?

https://www.youtube-nocookie.com/embed/1dXoajt6RCk

Element Enterprise provides an enterprise-grade secure communications platform that can be run either on your own premise or in our Element Cloud. Element Enterprise includes all of the security and privacy features that you get with Element:

- Built on the Matrix open communications standard.
- Provides end to end encrypted messaging, voice, and video through a consumer style messenger with the power of a collaboration tool.
- Delivers data sovereignty.
- Affords a high degree of flexibility that can be tailored to many use cases.
- Allows secure federation within a single organisation or across a supply chain or ecosystem.

and combines them with the following unique Enterprise specific features:

- Group Sync: Synchronize group data from your identity provider and map these into Element spaces.
- Adminbot: Give your server administrator the ability to be admin in any rooms on your homeserver.
- Auditbot: Have an auditable record of conversations conducted on your homeserver.
- Security and feature updates: Updates are easy to deploy and handled by our installer.
- Support: Access to the experts in federated, secure communications giving you confidence to deploy our platform for your most critical secure communications needs.

Given the flexibility afforded by this platform, ours has a number of moving parts to configure. This documentation will step you through architecting and deploying Element Enterprise On-Premise.

Deploying to a Single Node or Multiple Nodes?

Element Enterprise On-Premise can be deployed both to a single node or a set of multiple nodes. In the case of the multiple node deployment, this requires Kubernetes, a container orchestration platform. In the case of our single node deployment, our installer deploys microk8s (a smaller lightweight distribution of Kubernetes) and deploys our application to that microk8s instance.

In general, regardless of if you pick a single node deployment or a multiple node deployment, you will need a base level of hardware to support the application.

For scenarios that utilise closed federation, Element recommends a minimum of 6 vCPUs/CPUs and 16GB ram for the host(s) running synapse pods.

For scenarios that utilise open federation, Element recommends a minimum of 8 vCPUs/CPUs and 32GB ram for the host(s) running synapse pods.

Architecture

This document gives an overview of our secure communications platform architecture:

(Please click on the image to view it at 100%.)

Comprising our secure communications platform are the following components:

- synapse : The homeserver itself.
- element-web : The Element Web client.
- integrator: Our integration manager.
- synapse admin ui : Our Element Enterprise Administrator Dashboard.
- postgresql (Optional) : Our database. Only optional if you already have a separate PostgreSQL database.
- groupsync (Optional) : Our group sync software
- adminbot (Optional) : Our bot for admin tasks.
- auditbot (Optional) : Our bot that provides auditability.
- hookshot (Optional) : Our integrations with gitlab, github, jira, and custom webhooks.
- hydrogen (Optional) : A light weight alternative chat client.
- jitsi (Optional) : Our VoIP platform for group conferencing.
- coturn (Optional) : TURN server. Required if deploying VoIP.

- prometheus (Optional) : Provides metrics about the application and platform.
- grafana (Optional) : Graphs metrics to make them consumable.
- telegram bridge (Optional) : Bridge to connect Element to Telegram.
- teams bridge (Optional) : Bridge to connect Element to MS Teams.
- xmpp bridge (Optional) : Bridge to connect Element to XMPP.
- irc bridge (Optional) : Bridge to connect Element to IRC.
- sip bridge (Optional) : Bridge to connect Element to SIP.

For each of the components in this list (excluding postgresql, groupsync, adminbot, auditbot, and prometheus), you must provide a hostname on your network that meets this criteria:

- Fully resolvable to an IP address that is accessible from your clients.
- Signed PEM encoded certificates for the hostname in a crt/key pair. Certificates should be signed by an internet recognised authority, an internal to your company authority, or LetsEncrypt.

It is possible to deploy Element Enterprise On-Premise with self-signed certificates and without proper DNS in place, but this is not ideal as the mobile clients and federation do not work with self-signed certificates. Information on how to use self-signed certificates and hostname mappings instead of DNS can be found in How to Setup Local Host Resolution Without DNS

In addition to hostnames for the above, you will also need a hostname and PEM encoded certificate key/cert pair for your base domain. If we were deploying a domain called example.com and wanted to deploy all of the software, we would have the following hostnames in our environment that needed to meet the above criteria:

- example.com (base domain)
- synapse.example.com (homeserver)
- element.example.com (element web)
- integrator.example.com (integration manager)
- admin.example.com (admin dashboard)
- hookshot.example.com (Our integrations)
- hydrogen.example.com (Our light weight chat client)
- jitsi.example.com (Our VoIP platform)
- coturn.example.com (Our TURN server)
- grafana.example.com (Our Grafana server)
- telegrambridge.example.com (Our Telegram Bridge)
- teamsbridge.example.com (Our Teams Bridge)

As mentioned above, this list excludes postgresql, groupsync, adminbot, auditbot, and prometheus.

Further, if you want to do voice or video across network boundaries (ie: between people not on the same local network), you will need a TURN server. If you already have one, you do not have to set up coturn. If you do not already have a TURN server, you will want to set up coturn and if your server is behind NAT, you will need to have an external IP in order for coturn to work.

Installation

Software

To obtain our software, please visit our downloads page at: https://ems.element.io/on-premise/download

Prior to downloading our software, you'll need to determine if you want to do a multiple node deployment or a single node deployment and whether or not you'll be running in an airgapped environment.

Multiple Nodes

For a multiple node installation, make sure you have a Kubernetes platform deployed that you have access to and head over to Kubernetes Installations

Single Node

For a single node installation, please note that we support these on the following platforms:

- Ubuntu Server 20.04
- Enterprise Linux 8 (RHEL, CentOS Stream, etc.)

Once you have a server with one of these installed, please head over to Single Node Installations

Kubernetes Installations

How to Use the Installer

This video covers the single node installer, but many of the concepts are also applicable to our multi-node installer.

https://www.youtube-nocookie.com/embed/1dXoajt6RCk

Overview

Our Element Enterprise Kubernetes Installer can handle the installation of Element Enterprise into your production kubernetes (k8s) environment.

To get started with a kubernetes installation, there are several things that need to be considered and this guide will work through them:

- k8s Environments
- Postgresql Database
- TURN Server
- SSL Certificates
- Extra configuation items

Once these areas have been covered, you'll be able to install a production environment!

Unpacking the Installer

Please make sure that you unpack `element-enterprise-installer` onto a system that has access to your k8s environment. The directory that it unpacks into will be referenced in this document as the installer directory.

You will also need to create a directory for holding the configurations for the installer. This will be referenced as the config directory going forward.

```
mkdir  ~/.element-onpremise-config
```

k8s Environments

Element Enterprise Installer allows you to either deploy directly into a kubernetes environment or to render a set of manifests for a future deployment in a kubernetes environment.

To configure your kubernetes environment for a direct deployment, you need to :

- Configure a kubectl context able to connect to your kubernetes instance
- Copy `k8s.yml.sample` to `k8s.yml` in your config directory. Edit `k8s.yml` with the following values :
- `provider_storage_class_name`: The storage class to use when creating PVCs.
- `ingress_annotations`: The annotations to add to the ingresses created by the operator.
- `tls_managed_externally`: Should be true if you don't expect the operator to manage the certificates of your kubernetes deployment. In this case, you will be able to skip the * *Certificates*- chapter of the `CONFIGURE.md` file.
- `operator_namespace`: The namespace to create to deploy the operator.
- `element_namespace`: The namespace to create to deploy the element resources.
- `k8s_auth_context`: The value of the context used in kubectl. If you want to use cert-manager for your tls certificates, it needs to be already installed in the targeted k8s cluster.

An example k8s.yml file would look like:

```
provider_storage_class_name: gp8-delete  # select an available storage class
ingress_annotations: ## below are expected annotations for an aws deployment
    kubernetes.io/ingress.class: alb
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/group.name: global
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/ip-address-type: ipv4
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80},{"HTTPS": 443}]'
synapse_ingress_annotations:  # below are required annotations if using the NGINX ingress
controller
    nginx.ingress.kubernetes.io/proxy-body-size: "50m"
tls_managed_externally: true  # true if the certificates are managed externaly to k8s
security_context_force_uid_gid: true  # true to enable pod runAsUser and fsGroup in security
context. false if it should not be used, in the case of openshift for example.
security_context_set_seccomp: true  # true to enable RuntimeDefault pod seccomp. false if it
should not be used, in the case of openshift for example.
operator_namespace: <namespace to create to deploy the operator>
element_namespace: <namespace to create to deploy the element resources>
```

```
k8s_auth_context: <the k8s auth context>
out_dir: # Absolute path to the directory where to render manifests, if render mode is used
# operator_manager_limits:  # Can be used to defined upper limits if the default one are not
large enough for your operator deployment
#   cpu: "2"
#   memory: 8Gi
```

If you do not want to deploy directly to kubernetes, but wish to render manifests instead, set all of the above mentioned variables except for `k8s_auth_context` and define a value for the parameter `out_dir`, which specifies where to write the kubernetes manifests. Further, when you go to run the installer, you need to invoke it as such:

```
bash install.sh  ~/.element-onpremise-config --target render
```

Using the above syntax, you will have a set of manifests written out to `out_dir` that you can then deploy into your kubernetes environment.

N.B. You will need to set your ingress controller's upload size to be at least 50 Mb to match synapse's default upload size if you wish to be able to have users upload files up to 50 Mb in size. Instructions for doing this with nginx are included in the `parameters.yml` section below.

Postgresql Database

The installation requires that you have a postgresql database with a locale of C and UTF8 encoding set up. See https://matrix-org.github.io/synapse/latest/postgres.html#set-up-database for further details.

Please make note of the database hostname, database name, user, and password as you will need these to begin the installation.

TURN Server

For installations in which you desire to use video conferencing functionality, you will need to have a TURN server installed and available for Element to use.

If you do not have an existing TURN server, our installer can configure one for you by following the extra steps in Setting Up Jitsi and TURN With the Installer.

If you have an existing TURN server, please create a file called `synapse/turn.yml` in your config directory and put the following in it:

```
turn_uris: [ "turn:turn.matrix.org?transport=udp", "turn:turn.matrix.org?transport=tcp" ]
turn_shared_secret: "n0t4ctuAllymatr1Xd0TorgSshar3d5ecret4obvIousreAsons"
turn_user_lifetime: 86400000
turn_allow_guests: True
```

based on your TURN server specifics. This will allow the installer to configure synapse to use your TURN server.

A few notes on TURN servers:

- The TURN server has to be directly accessible by end-users. Normally this means a public IP, however if all the end-users are going to be on a VPN/private network then they just need to be able to access the private IP of the TURN server.
- The only reason to have TURN on a private network is if the private network disallows user <-> user traffic and only allows user <-> TURN server traffic. If user <-> user is allowed within the private network then a TURN server isn't needed.

SSL Certificates

For SSL Certificates, you have three options:

- Signed PEM encoded certificates from an internet recognized authority.
- Signed PEM encoded certificates from an internal to your company authority.
- LetsEncrypt

In the case of LetsEncrypt, your hostnames must be accessible on the internet.

You will need to configure certificates for the following names:

- fqdn.tld
- element.fqdn.tld
- synapse.fqdn.tld
- dimension.fqdn.tld
- hookshot.fqdn.tld

Using our example hosts, this would mean that we need certificates for:

- local
- element.local
- synapse.local
- dimension.local
- hookshot.local

Certificates without LetsEncrypt

If you have certificates for all of the aforementioned host names, then you can simply place the `.crt` and `.key` files in the `certs` directory under the config directory. Certificates in the `certs` directory must take the form of `fqdn.cert` and `fqdn.key`.

Certificates with LetsEncrypt

Our installer also supports using LetsEncrypt to build certificates for your host names and automatically install them into your environment. If your hosts are internet accessible, this is the easiest method and only requires an admin email address to provide to LetsEncrypt.

parameters.yml

Now it is time to set `parameters.yml`. A sample has been provided and to get started, it is easiest to do:

```
cp config-sample/parameters.yml.sample ~/.element-onpremise-config/parameters.yml
```

Using the example hostnames of `element.local` and `synapse.local`, we would set the following parameters first in `parameters.yml`:

```
domain_name: local
element_fqdn: element.local
synapse_fqdn: synapse.local
```

Next, we need to set the variables related to Postgres. As you are installing into kubernetes, you will need to set the following for your Postgres database:

```
postgres_create_in_cluster: false
postgres_fqdn: `Postgres Server`
postgres_user: `Postgres User`
postgres_db: `Postgres Database for Element`
```

The next line states:

```
media_size: "50Gi"
```

You wll want to adjust that to match the size of storage you've allocated for your media. It must be at least 50Gb.

The next section pertains to certmanager. If you are using your own certificates, please leave these items both blank, as such:

```
certmanager_issuer:
certmanager_admin_email:
```

If you have chosen to use letsencrypt, please specify “letsencrypt” for the certmanager_issue and an actual email address for who should manage the certificates for certmanager_admin_email:

```
certmanager_issuer: 'letsencrypt'
certmanager_admin_email: 'admin@mydomain.com'
```

Starting with installer 2022-08.02, we have added two mandatory variables related to telemetry data. These are `max_mau_users` and `strict_mau_users_limit`. You should set `max_mau_users` to the value defined in your contract with Element. If you set this number above your contractual limit, then the software will allow you to exceed your contractual limit and Element will bill you appropriately for the overage.

Setting `strict_mau_users_limit` to `true` forces synapse to cap the number of monthly active users to the value defined in `max_mau_users`. Say for example, you've paid Element for 1,000 monthly active users and don't want to exceed that, you would set:

```
max_mau_users: 1000
strict_mau_users_limit: true
```

Let's say that you paid Element for 1,000 monthly active users, but didn't mind going over provided that you didn't exceed 2,000 monthly active users. In this scenario, you would set:

```
max_mau_users: 2000
strict_mau_users_limit: true
```

For more information on the data that Element collects, please see: What Telemetry Data is Collected by Element?

You will also see two paths:

```
# media_host_data_path: "/mnt/data/synapse-media"
# postgres_data_path: "/mnt/data/synapse-postgres"
```

For kubernetes installations, please make sure these are commented out.

You will also notice two lines towards the end regarding `synapse_registration` and `tls_managed_externally`. In most cases, you can leave these alone, but if you wish to close synapse

registration or have your TLS managed externally, you may set them at this time.

If you are using nginx as your ingress controller and wish to send files up to 50 Mb in size, please add these two lines to parameters.yml:

```
synapse_ingress_annotations:
  nginx.ingress.kubernetes.io/proxy-body-size: "50m"
```

secrets.yml

Now we move on to configuring `secrets.yml`. You will need the following items here:

- A Macaroon key
- Your postgres password for the user specified in parameters.yml
- A Registration Shared Secret
- A signing Key
- An EMS Image Store username and token, which will have been provided to you by Element.

To build a `secrets.yml` with the macaroon key, the registration shared secret, the generic shared secret, and the signing key already filled in, please run (in the installer folder):

```
sh build_secrets.sh
mv secrets.yml  ~/.element-onpremise-config/
```

You will need to uncomment and set your `postgres_password` field to the proper password for your database.

Do not forget to also set the values for `ems_image_store_username` and `ems_image_store_token`, which will both be provided by Element.

If you have a paid docker hub account, you can specify your username and password to avoid being throttled in the `dockerhub_username` and `dockerhub_token` fields. This is optional.

Extra Configuration Items

It is possible to configure anything in Synapse's homeserver.yaml or Element’s config.json.

To do so, you need to create json or yaml files in the appropriate directory under the config directory. These files will be merged to the target configuration file.

Samples are available in `config-sample` under the installer directory.

To configure synapse:

- Create a directory `synapse` at the root of the config directory : `mkdir ~/.element-onpremise-config/synapse`
- Copy the configurations extensions you want to setup from `config-sample/synapse` to `~/.element-onpremise-config/synapse`.
- Edit the values in the file accordingly to your configuration

To configure element:

- Create a directory `element` at the root of the config directory : `mkdir ~/.element-onpremise-config/element`
- Copy the configurations extensions you want to setup from `config-sample/element` to `~/.element-onpremise-config/element`.
- Edit the values in the file accordingly to your configuration

For specifics on configuring permalinks for Element, please see Setting up Permalinks.

For specifics on configuring well known delegation, please see Setting Up Well Known Delegation

For specifics on setting up Delegated Authentication, please see Setting up Delegated Authentication With the Installer

For specifics on setting up Group Sync, please see Setting up Group Sync

For specifics on setting up the Integration Manager, please see Setting Up the Integration Manager With the Installer

For specifics on setting up GitLab, GitHub, and JIRA integrations, please see Setting up GitLab, GitHub, and JIRA Integrations With the Installer

For specifics on setting up Adminbot and Auditbot, please see: Setting up Adminbot and Auditbot

For specifics on setting up the Enterprise Admin Dashboard, please see: Configuring the Enterprise Admin Dashboard

For specifics on setting up Hydrogen, please see: Setting Up Hydrogen

For specifics on pointing your installation at an existing Jitsi instance, please see Setting Up Jitsi and TURN With the Installer

For specifics on configuring the Teams Bridge, please see Setting Up the Teams Bridge

For specifics on configuring the Telegram Bridge, please see Setting Up the Telegram Bridge

For specifics on configuring the IRC Bridge, please see Setting Up the IRC Bridge

For specifics on configuring the XMPP Bridge, please see Setting Up the XMPP Bridge

Installation

Let’s review! Have you considered:

- k8s Environments
- Postgresql Database
- TURN Server
- SSL Certificates
- Extra configuration items

Once you have the above sections taken care of and your `parameters.yml` and `secrets.yml` files are in order, you are ready to begin the actual installation.

From the installer directory, run:

```
bash install.sh  ~/.element-onpremise-config
```

The first run should go for a little while and then exit, instructing you to log out and back in.

Please log out and back in and re-run the installer from the installer directory again:

```
bash install.sh  ~/.element-onpremise-config
```

End-User Documentation

After completing the installation you can share our User Guide to help orient and onboard your users to Element

Upgrading between installer versions

Installer 2022-08.02

Added two **mandatory** variables related to telemetry data: `max_mau_users` and `strict_mau_users_limit`

They need to be set (ie: `max_mau_users = 1000` and `strict_mau_users_limit = true`).

Single Node Installations

How to Use the Installer

https://www.youtube-nocookie.com/embed/1dXoajt6RCk

Overview

Our Element Enterprise Single Node Installer can handle the installation of environments in which only one server is available. Our single node environment consists of a single server with microk8s running that we deploy our Element Enterprise Operator to, resulting in a fully functioning Synapse server with Element Web.

To get started with a single node installation, there are several things that need to be considered and this guide will work through them:

- Operating System
- Postgresql Database
- TURN Server
- SSL Certificates
- Extra configuration items

Once these areas have been covered, you'll be able to install a single node environment!

Operating System

To get started, we have tested on Ubuntu 20.04 and Red Hat Enterprise Linux 8.5 and suggest that you start there as well. For x86_64, you can grab an Ubuntu iso here:

https://releases.ubuntu.com/20.04.3/ubuntu-20.04.3-live-server-amd64.iso

or you can get Red Hat Enterprise Linux 8 with a Developer Subscription

https://developers.redhat.com/content-gateway/file/rhel-8.6-x86_64-dvd.iso

Note that future references in this document to `EL` reference Enterprise Linux.

Ubuntu Specific Directions

Make sure to select docker as a package option. Do set up ssh.

Once you log in, please run:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install python3-signedjson pwgen -y
```

The installer requires that you run it as a non-root user who has sudo permissions. Please make sure that you have a user who can use `sudo`. If you wanted to make a user called `element-demo` that can use `sudo`, the following commands (run as root) would achieve that:

```
useradd element-demo
gpasswd -a element-demo sudo
```

EL Specific directions

Make sure to select "Container Management" in the "Additional Software" section.

Once you log in, please run:

```
sudo yum update -y
sudo yum install podman-docker python-pip -y
sudo yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm -y
sudo update-alternatives --config python3
```

Add the following lines to `/etc/security/limits.conf`:

```
*              soft    nofile  100000
*              hard    nofile  100000
```

Then, run:

```
sudo yum install make gcc python3-devel pwgen -y
pip3 install signedjson --user
sudo update-alternatives --config python3
```

The installer requires that you run it as a non-root user who has sudo permissions. Please make sure that you have a user who can use `sudo`. If you wanted to make a user called `element-demo` that can use `sudo`, the following commands (run as root) would achieve that:

```
useradd element-demo
gpasswd -a element-demo wheel
```

Setting up the Configuration Directory

You should have the installer unpacked in a directory on your server. We will refer to this as the installer directory. You will also need to create a configuration directory that we will call the config directory. Both the `parameters.yml` and `secrets.yml` file live in the config directory.

To create the configuration directory, run the following:

```
mkdir ~/.element-onpremise-config
```

Network Specifics

Element Enterprise On-Premise needs to bind and serve content over:

- Port 80 TCP
- Port 443 TCP

microk8s needs to bind and serve content over:

- Port 16443 TCP
- Port 10250 TCP
- Port 10255 TCP
- Port 25000 TCP
- Port 12379 TCP
- Port 10257 TCP
- Port 10259 TCP
- Port 19001 TCP

For more information, see https://microk8s.io/docs/ports.

In a default Ubuntu installation, these ports are allowed through the firewall. You will need to ensure that these ports are passed through your firewall.

For EL, you need to explicitly open the above ports and enabling masquerading:

```
sudo firewall-cmd --add-service={http,https} --permanent
sudo firewall-cmd --add-port=16443/tcp --add-port=10250/tcp --add-port=10255/tcp --add-port=25000/tcp --add-port=12379/tcp --add-port=10257/tcp --add-port=10259/tcp --add-port=19001/tcp --permanent
sudo firewall-cmd --add-masquerade --permanent
sudo firewall-cmd --reload
```

Further, you need to make sure that your host is able to access the following hosts on the internet:

- api.snapcraft.io
- *.snapcraftcontent.com
- gitlab.matrix.org
- gitlab-registry.matrix.org
- pypi.org
- docker.io
- *.docker.com
- get.helm.sh

Further, you will also need to make sure that your host can access your distributions' package repositories. As these hostnames can vary, it is beyond the scope of this documentation to enumerate them.

Network Proxies

We also cover the case where you need to use a proxy to access the internet. Please see this article for more information: Configuring a microk8s Single Node Instance to Use a Network Proxy

Unpacking the Installer

Please make sure that you unpack `element-enterprise-installer` onto your single node system. The directory that it unpacks into will be referenced in this document as the installer directory.

Postgresql Database

The installation requires that you have a postgresql database with a locale of C and UTF8 encoding set up. See https://github.com/matrix-org/synapse/blob/develop/docs/postgres.md#set-up-database for further details.

If you have this already, please make note of the database name, user, and password as you will need these to begin the installation.

If you do not already have a database, then the single node installer will set up PostgreSQL on your behalf.

TURN Server

For installations in which you desire to use video conferencing functionality, you will need to have a TURN server installed and available for Element to use.

If you do not have an existing TURN server, our installer can configure one for you by following the extra steps in Setting Up Jitsi and TURN With the Installer

If you have an existing TURN server, please create a file called `synapse/turn.yml` in your config directory and put the following in it:

```
turn_uris: [ "turn:turn.matrix.org?transport=udp", "turn:turn.matrix.org?transport=tcp" ]
turn_shared_secret: "n0t4ctuAllymatr1Xd0TorgSshar3d5ecret4obvIousreAsons"
turn_user_lifetime: 86400000
turn_allow_guests: True
```

based on your TURN server specifics. This will allow the installer to configure synapse to use your TURN server.

A few notes on TURN servers:

- The TURN server has to be directly accessible by end-users. Normally this means a public IP, however if all the end-users are going to be on a VPN/private network then they just need to be able to access the private IP of the TURN server.
- The only reason to have TURN on a private network is if the private network disallows user <-> user traffic and only allows user <-> TURN server traffic. If user <-> user is allowed within the private network then a TURN server isn't needed.

SSL Certificates

For SSL Certificates, you have three options:

- Signed PEM encoded certificates from an internet recognized authority.
- Signed PEM encoded certificates from an internal to your company authority.
- LetsEncrypt
- Self-signed certificates

In the case of Signed certificates or LetsEncrypt, your hostnames must be accessible on the internet.

In the case of self-signed certificates, these are acceptable for a PoC (proof of concept) environment, but will not be supported in a production environment as the security risk would be too high. Configuring mobile clients and federation will not be possible with self-signed certificates.

You will need to configure certificates for the following names:

- fqdn.tld
- element.fqdn.tld
- synapse.fqdn.tld
- dimension.fqdn.tld
- hookshot.fqdn.tld

Using our example hosts, this would mean that we need certificates for:

- local
- element.local
- synapse.local
- dimension.local
- hookshot.local

Certificates without LetsEncrypt

If you have certificates for all of the aforementioned host names, then you can simply place the PEM encoded `.crt` and `.key` files in the certs directory under the configuration directory. Certificates in the certs directory must take the form of `fqdn.crt` and `fqdn.key`.

Self-signed certificates with mkcert

For information on using self-signed certificates with mkcert, please see this article: Using Self-Signed Certificates with mkcert

Certificates with LetsEncrypt

Our installer also supports using LetsEncrypt to build certificates for your host names and automatically install them into your environment. If your hosts are internet accessible, this is the easiest method and only requires an admin email address to provide to LetsEncrypt.

parameters.yml

Now it is time to set `parameters.yml`. A sample has been provided and to get started, it is easiest to do:

```
cp config-sample/parameters.yml.sample ~/.element-onpremise-config/parameters.yml
```

Using the example hostnames of `element.local` and `synapse.local`, we would set the following parameters first in `parameters.yml`:

```
domain_name: local
element_fqdn: element.local
synapse_fqdn: synapse.local
```

Next, we need to set the variables related to Postgres. If you do not have an existing Postgres server, do not make any changes. If you have an existing Postgres server, set the following:

```
postgres_create_in_cluster: false
postgres_fqdn: `Postgres Server`
postgres_user: `Postgres User`
postgres_db: `Postgres Database for Element`
```

The next line states:

```
media_size: "50Gi"
```

You wll want to adjust that to match the size of storage you've allocated for your media. It must be at least 50Gb.

The next section pertains to certmanager. If you are using your own certificates, please leave these items both blank, as such:

```
certmanager_issuer:
certmanager_admin_email:
```

If you have chosen to use letsencrypt, please specify “letsencrypt” for the certmanager_issue and an actual email address for who should manage the certificates for certmanager_admin_email:

```
certmanager_issuer: 'letsencrypt'
certmanager_admin_email: 'admin@mydomain.com'
```

Starting with installer 2022-08.02, we have added two mandatory variables related to telemetry data. These are `max_mau_users` and `strict_mau_users_limit`. You should set `max_mau_users` to the value defined in your contract with Element. If you set this number above your contractual limit, then the software will allow you to exceed your contractual limit and Element will bill you appropriately for the overage.

Setting `strict_mau_users_limit` to `true` forces synapse to cap the number of monthly active users to the value defined in `max_mau_users`. Say for example, you've paid Element for 1,000 monthly active users and don't want to exceed that, you would set:

```
max_mau_users: 1000
strict_mau_users_limit: true
```

Let's say that you paid Element for 1,000 monthly active users, but didn't mind going over provided that you didn't exceed 2,000 monthly active users. In this scenario, you would set:

```
max_mau_users: 2000
strict_mau_users_limit: true
```

For more information on the data that Element collects, please see: What Telemetry Data is Collected by Element?

You will also see two paths:

```
media_host_data_path: "/mnt/data/synapse-media"
# postgres_data_path: "/mnt/data/synapse-postgres"
```

For all installations, `media_host_data_path` should be uncommented. For installations in which you are letting the installer install postgresql for you, please uncomment the `postgres_data_path` line.

The next lines concern `images_dir` and `local_registry`. These are only needed in an air-gapped environment. If you are installing into an air-gapped environment, please see: Using the Single Node Installer in an Air-Gapped Environment

The next item in the configuration is the microk8s DNS resolvers. By default, the installer will use Google's publicly available DNS servers. If you have defined your hosts on a non-publicly available DNS server, then you should use your DNS servers instead of the publicly available Google DNS servers. Let's assume that your local dns servers are 192.168.122.253 and 192.168.122.252. To use those servers, you would need to add this line:

```
microk8s_dns_resolvers: "192.168.122.253,192.168.122.252"
```

You will also notice two lines towards the end regarding `synapse_registration` and `tls_managed_externally`. In most cases, you can leave these alone, but if you wish to close synapse registration or have your TLS managed externally, you may set them at this time.

Further, if you are not using DNS for hostname mapping, you will need to configure the `host_aliases` parameter in this file and that is documented in How to Setup Local Host Resolution Without DNS.

secrets.yml

Now we move on to configuring `secrets.yml`. You will need the following items here:

- A Macaroon key
- Your postgres password for the user specified in parameters.yml
- A Registration Shared Secret
- A signing Key
- An EMS Image Store username and token, which will have been provided to you by Element.

To build a `secrets.yml` with the macaroon key, the registration shared secret, the generic shared secret, and the signing key already filled in, please run (in the installer folder):

```
sh build_secrets.sh
mv secrets.yml  ~/.element-onpremise-config/
```

If you are using your own Postgres server, you will need to uncomment and fill in the `postgres_passwd`. If you are letting the installer install Postgres for you, then you will need to set a random password. You can generate a random password with:

```
pwgen 32 1
```

and then insert that value in the `postgres_passwd` field, making sure that you uncomment the line.

Do not forget to also set the values for `ems_image_store_username` and `ems_image_store_token`, which will both be provided by Element.

If you have a paid docker hub account, you can specify your username and password to avoid being throttled in the `dockerhub_username` and `dockerhub_token` fields. This is optional.

Extra Configuration Items

It is possible to configure anything in Synapse's homeserver.yaml or Element’s config.json.

To do so, you need to create json or yaml files in the appropriate directory under the config directory. These files will be merged to the target configuration file.

Samples are available in `config-sample` under the installer directory.

To configure synapse:

- Create a directory `synapse` at the root of the config directory : `mkdir ~/.element-onpremise-config/synapse`
- Copy the configurations extensions you want to setup from `config-sample/synapse` to `~/.element-onpremise-config/synapse`.
- Edit the values in the file accordingly to your configuration

To configure element:

- Create a directory `element` at the root of the installer directory : `mkdir ~/.element-onpremise-config/element`
- Copy the configurations extensions you want to setup from `config-sample/element` to `~/.element-onpremise-config/element`.
- Edit the values in the file accordingly to your configuration

For specifics on configuring permalinks for Element, please see Setting up Permalinks With the Installer

For specifics on configuring well known delegation, please see Setting Up Well Known Delegation

For specifics on setting up Delegated Authentication, please see Setting up Delegated Authentication With the Installer

For specifics on setting up Group Sync, please see Setting up Group Sync with the Installer

For specifics on setting up the Integration Manager, please see Setting Up the Integration Manager With the Installer

For specifics on setting up GitLab, GitHub, and JIRA integrations, please see Setting up GitLab, GitHub, and JIRA Integrations With the Installer

For specifics on setting up Adminbot and Auditbot, please see: Setting up Adminbot and Auditbot

For specifics on setting up the Enterprise Admin Dashboard, please see: Configuring the Enterprise Admin Dashboard

For specifics on setting up Hydrogen, please see: Setting Up Hydrogen

For specifics on pointing your installation at an existing Jitsi instance, please see Setting Up Jitsi and TURN With the Installer

For specifics on configuring the Teams Bridge, please see Setting Up the Teams Bridge

For specifics on configuring the Telegram Bridge, please see Setting Up the Telegram Bridge

For specifics on configuring the IRC Bridge, please see Setting Up the IRC Bridge

For specifics on configuring the XMPP Bridge, please see Setting Up the XMPP Bridge

Installation

Let’s review! Have you considered:

- Operating System
- Postgresql Database
- TURN Server
- SSL Certificates
- Extra configuration items

Once you have the above sections taken care of and your `parameters.yml` and `secrets.yml` files are in order, you are ready to begin the actual installation.

From the installer directory, run: (Note: You can replace `~/.element-onpremise-config` with whatever you have specified for your config directory.)

```
bash install.sh  ~/.element-onpremise-config
```

The first run should go for a little while and then exit, instructing you to log out and back in.

Please log out and back in and re-run the installer from the installer directory again:

```
bash install.sh  ~/.element-onpremise-config
```

Once this has finished, you can run:

```
kubectl get pods -n element-onprem
```

And you should get similar output to:

```
NAME                                          READY   STATUS    RESTARTS   AGE
app-element-web-c5bd87777-rqr6s               1/1     Running   1          29m
server-well-known-8c6bd8447-wddtm             1/1     Running   1          29m
postgres-0                                    1/1     Running   1          40m
instance-synapse-main-0                       1/1     Running   2          29m
instance-synapse-haproxy-5b4b55fc9c-hnlmp     1/1     Running   0          20m
```

At this time, you should also be able to browse to: `https://fqdn` and create a test account with Element on your new homeserver. Using our example values, I am able to go to `https://element.local/` and register an account, sign in and begin testing!

End-User Documentation

After completing the installation you can share our User Guide to help orient and onboard your users to Element

Upgrading between installer versions

Installer 2022-08.02

Added two **mandatory** variables related to telemetry data:
`max_mau_users` and `strict_mau_users_limit`

They need to be set (ie: `max_mau_users = 1000` and `strict_mau_users_limit = true`).

Using the Installer in an Air-Gapped Environment

Defining Air-Gapped Environments

An air-gapped environment is any environment in which the running hosts will not have access to the greater internet. This proposes a situation in which these hosts are unable to get access to various needed bits of software from Element and also are unable to share telemetry data back with Element.

For some of these environments, they can be connected to the internet from time to time and updated during those connection periods. In other environments, the hosts are never connected to the internet and everything must be moved over sneaker net.

This guide will cover running the microk8s installer when only sneaker net is available as that is the most restrictive of these environments.

Preparing the media to sneaker net into the air gapped environment

You will need our airgapped dependencies .tar.gz file which you can get from Element:

- element-enterprise-installer-airgapped-<version>.tar.gz

Running the installer in the air gapped environment

Extract the airgapped dependencies to the `airgapped` directory at the root of the installer folder. You obtain the following directories :

- `airgapped/pip`
- `airgapped/galaxy`
- `airgapped/snaps`
- `airgapped/containerd`
- `airgapped/images`

Your airgapped machine will still require access to airgapped linux repositories depending on your OS.

Add the following parameters in your parameters.yml :

- `local_registry: localhost:32000`
- `images_dir: <absolute path to the airgapped/images directory>`

The installer will upload the images automatically to your local registry, and use these references to start the workloads.

When running the install script, add the parameter `--airgapped` so that it installs its pip and galaxy dependencies from the airgapped folder.

If you are using the kubernetes installer (instead of the single node installer), please note that once the image upload has been done, you will need to copy the `airgapped/images/images_digests.yml` file to the same path on the machine which will be used to render or deploy element services. Doing this, the new image digests will be used correctly in the kubernetes manifests used for deployment.

Setting Up Jitsi and TURN With the Installer

Configure the Installer to install Jitsi and TURN

Prerequisites

Firewall

You will have to open the following ports to your microk8s host to enable coturn and jitsi :

For jitsi :

- `30301/tcp`
- `30300/udp`

For coturn, allow the following ports :

- `3478/tcp`
- `3478/udp`
- `5349/tcp`
- `5349/udp`

You will also have to allow the following port range, depending on the settings you define in `coturn.yml` (see below) :

- `<coturn min port>-<coturn max port>/udp`

DNS

The jitsi and coturn domain names must resolve to the VM access IP. You must not use `host_aliases` for these hosts to resolve to the private IP locally on your setup.

Coturn

- Copy sample file from `config-sample/coturn/coturn.yml` to the `coturn` sub-directory within your config folder
- Edit the file and add the following values :
  - `coturn_fqdn`: The access address to coturn. It should match something like `coturn.<fqdn.tld>`. **It must resolves to the public-facing IP of the VM**.
  - `shared_secret`: A random value, you can generate it with `pwgen 32`
  - `min_port`: The minimal UDP Port used by coturn for relaying UDP Packets, in range 32769-65535
  - `max_port`: The maximum UDP Port used by coturn for relaying UDP Packets, in range 32769-65535

Jitsi

- Copy sample file from `config-sample/jitsi/jitsi.yml` to the `jitsi` sub-directory within your config folder
- Edit the file and add the following values :
  - `jitsi_fqdn`: The access address to jitsi. It should match something like `jitsi.<fqdn.tld>`. **It must resolves to the public-facing IP of the VM**.
  - `jicofo_auth_password`: # a secret internal password for jicofo auth
  - `jicofo_component_secret`: # a secret internal password for jicofo component
  - `jvb_auth_password`: # a secret internal password for jvb
  - `helm_override_values`: {} # if needed, to override helm settings automatically set by the installer
  - `timezone`: Europe/Paris # The timezone in TZ format
  - `stun_servers`: Needed if you don't setup coturn using the installer. Should be a yaml list of server:port entries. Example:

```
stun_servers:
- ip:port
- ip:port
```

Element

- Copy sample file from `config-sample/element/jitsi.json` to the `element` sub-directory within your config folder
- Edit the file and replace `<jitsi_fqdn>` by the value of jitsi fqdn.

Restart the install script once everyting is set.

Configure the installer to use an existing Jitsi instance

- Create a file called `jitsi.json` in the `~/.element-onpremise-config/element` directory.
- Edit the file :

```
{
      "jitsi": {
            "preferredDomain": "your.jitsi.example.org"
      }
}
```

replacing `your.jitsi.example.org` with the hostname of your Jitsi server.

- Restart the install script

Configure the installer to use an existing Coturn instance

Follow the instructions here: https://ems-docs.element.io/books/element-on-premise-documentation/page/single-node-installations#bkmrk-turn-server

Setting up Permalinks With the Installer

Element Extra Configurations

- Copy sample file from `config-sample/element/permalinks.json` in the installer directory to `~/.element-onpremise-config/element`
- Edit the file :

```
{
    "permalinkPrefix": "https://<element fqdn>"
}
```

- Restart the install script

Setting Up Well Known Delegation

Well Known Delegation Configuration

- Copy sample file from `config-sample/wellknowndelegation/wellknowndelegation.yml` in the installer directory to `CONFIG_DIRECTORY/wellknowndelegation`
- Edit the file :

```
---
extra_client: |
  {
    "im.vector.riot.jitsi": {
      "preferredDomain": "jitsi.dev.local"
    }
  }
extra_server: ""
```

Make whatever settings you need for the `extra_client` or `extra_server` parameters.

- Restart the install script

Setting up Delegated Authentication With the Installer

On Element Enterprise

- Depending on your provider, copy the sample file in the installer root directory from `config-sample/synapse/` to `~/.element-onpremise-config/synapse`

```
cp -r config-sample/synapse ~/.element-onpremise-config/synapse
```

- Edit the file for the provider you are setting up. You have at least 3 parameters to edit :
  - The IdP metadata url
  - The name and description of your synapse server, which your provider would display to inform the users to which app they are logging in
- Disable the local synapse user database and password workflows by creating a file `~/.element-onpremise-config/synapse/disable-local.yml` and putting the following in it:

```
password_config:
   localdb_enabled: false
   enabled: false
```

- Disable local user workflows in element by creating a file `~/.element-onpremise-config/element/delegatedauth.json` and putting the following in it:

```
{
  "setting_defaults": {
     "UIFeature.identityServer": false,
     "UIFeature.passwordReset": false,
     "UIFeature.registration": false,
     "UIFeature.deactivate": false,
     "UIFeature.thirdPartyId": false
   }
```

```
}
```

- Run the installer to configure SAML provisioning

On the provider

Here we cover Azure ADFS and Keycloak.

Other SAML providers can be configured for use with Element Enterprise. Please contact Element for further information in the event that you are not using one of the above providers.

Azure ADFS

- With an account with enough rights, go to : Enterprise Applications Portal
- Click on `New Application`
- Click on `Create your own application` on the top left corner
- Choose a name for it, and select `Integrate any other application you don't find in the gallery`
- Click on "Create"
- Select `Set up single sign on`
- Select `SAML`
- `Edit` on `Basic SAML Configuration`
- In `Identifier`, add the following URL : `https://<synapse fqdn>/_synapse/client/saml2/metadata.xml`
- Remove the default URL
- In `Reply URL`, add the following URL : `https://<synapse fqdn>/_synapse/client/saml2/authn_response`
- Click on `Save`
- `Edit` on `Attributes & Claims`
- Remove all defaults additional claims
- Click on `Add new claim` to add the following claims. The UID will be used as the MXID, the value here is mostly a suggestion :
  - Name: `uid`, Transformation : `ExtractMailPrefix`, Parameter 1 : `user.userprincipalname`
  - Name: `email`, Source attribute : `user.mail`
  - Name: `displayName`, Source attribute : `user.displayname`
- Click on `Save`
- In `Users and Groups`, add groups and users which may have access to element

Keycloak

- In `Configure` > `Clients`, add a new client. Enter `https://<synapse fqdn>/_synapse/client/saml2/metadata.xml` as its Client ID
- In `Mappers`, add the 3 following mappers :
  - Name: `uid` : User attribute : `username`
  - Name: `email`, User attribute : `email`
  - Name: `displayName`, Javascript mapper : `user.FirstName + " " + user.lastName`

Setting up Group Sync with the Installer

What is Group Sync?

Group Sync allows you to use the ACLs from your identity infrastructure in order to set up permissions on Spaces and Rooms in the Element Ecosystem. Please note that the initial version we are providing only supports a single node, non-federated configuration.

General settings

- Create ~/.element-onpremise-config/groupsync
- Copy sample file from `config-sample/groupsync/gsync.yml` in the installer directory to `~/.element-onpremise-config/groupsync`

```
mkdir ~/.element-onpremise-config/groupsync
cp config-sample/groupsync/gsync.yml ~/.element-onpremise-config/groupsync/
```

- Edit the file with the following values :
  - `provisioner.default_rooms` : Optional. A list of rooms that'll get automatically created in managed space. The ID is required to enable GPS to track whether they were already created or not. You can change it, but it'll cause new rooms to be generated.
  - `provisioner.whitelisted_users` : Optional. A list of userid patterns that will not get kicked from rooms even if they don't belong to them according to LDAP. This is useful for things like the auditbot. Patterns listed here will be wrapped in ^ and $ before matching.
  - `provisioner.invite_to_public_rooms`: Optional. Defaults to true. Set to false to disable auto-invite to public rooms into spaces.
  - `verify_tls` : Optional. If doing a POC with self-signed certs, set this to 0. The default value is 1.

Configuring the source

LDAP Servers

- You should create a ldap account with read access to the OUs containing the users
- This account should use password authentication
- To use LDAP source, copy the file `config-sample/groupsync/ldap.yml` in the installer directory to `~/.element-onpremise-config/groupsync`

```
cp config-sample/groupsync/ldap.yml ~/.element-onpremise-config/groupsync/
```

edit the following variables :

- `ldap_check_interval_seconds`: The interval check in seconds
- `ldap_uri`: The LDAP Uri to connect to the ldap server
- `ldap_base`: The LDAP base used to build the space hierarchy. This OU will become the root space. Every OU below this base will be a child-space.
- `ldap_bind_dn`: The user bind dn to use to read the space hierarchy.
- `ldap_bind_password`: The user password
- `ldap_attrs_uid`: The attribute to use to map to users mxids
- `ldap_attrs_name`: The attribute to use to map to spaces names
- Restart the install script

MS Graph (Azure AD)

- You need to create an `App registration`. You'll need the `Tenant ID` of the organization, the `Application (client ID)` and a secret generated from `Certificates & secrets` on the app.
- For the bridge to be able to operate correctly, navigate to API permissions and ensure it has access to Group.Read.All, GroupMember.Read.All and User.Read.All. Ensure that these are Application permissions (rather than Delegated).
- Remember to grant the admin consent for those.
- To use MSGraph source, copy the file `config-sample/groupsync/msgraph.yml` in the installer directory to `~/.element-onpremise-config/groupsync` and edit the following variables :
  - `msgraph_tenant_id`: This is the "Tenant ID" from your Azure Active Directory Overview
  - `msgraph_client_id`: Register your app in "App registrations". This will be its "Application (client) ID"
  - `msgraph_client_secret` : Go to "Certificates & secrets", and click on "New client secret". This will be the "Value" of the created secret (not the "Secret ID").
- Restart the install script

Space Mapping

The space mapping mechanism allows us to configure additional spaces that Group Sync will maintain, beyond the ones that it creates by default. It is optional – the configuration can be skipped if no additional spaces are to be created.

This is especially useful when used with bridges other than LDAP, which would normally not create any spaces other than the company-wide one. When used with the LDAP backend, the spaces created from LDAP OrgUnits will be added to the list of subspaces of the toplevel space.

Space mapping also replaces the group power level configuration and group filtering, being a superset of their functionality. It is recommended to use space mapping in their instead, as they might eventually be deprecated and removed.

Note: transitioning to space mapping

If you're using Group Sync already and want to transition to space mapping, make sure to match the root space ID with the one that Group Sync has already created by default -- otherwise it will create a brand new space and forget about the old one.

For LDAP, the default ID is the DN (distinguished name) of your main OrgUnit.

For MS Graph, the ID should be your tenant ID.

For SCIM, use `scim:<client-id>`, where the client-id is what you have defined in client.id in your configuration.

You can verify what the ID of the existing space is by running Group Sync in dry-run mode (`-n1` launch parameter).

Configuration

We define each space giving it a name (which will be displayed in Element), a unique ID (which allows Group Sync to track the Space even if it gets renamed), and a list of groups whose users will become the members of the Space. Users needs to be a member of *any* configured group, not all of them.

You can pick any ID you want (taking note of the section above), but if you change it later Group Sync will create a brand new space and abandon the old ones, likely confusing the users.

Each group may optionally include a powerLevel setting, allowing specific groups to have elevated permissions in the space.

A special group ID of `''` (an empty string) indicates that all users from the server, regardless of their group membership, should become the members of the Space.

An optional list of subspaces may also be configured, each using the same configuration format and behaviour (recursively).

The default Group Sync behaviour is equivalent to the following Space Mapping:

```
spaces:
  id: root
  name: 'Company'
  groups:
    - externalId: '' # include all users, not limited to any group
```

In order to limit space membership to a specific Group, we include its Group ID. This is equivalent to the `group_filter` configuration option.

```
spaces:
  id: root
  name: 'Company'
  groups:
    - externalId: 'element-users'
```

With `powerLevel` option allows us to give users extra permissions. This is equivalent to the `group_power_level` setting[^note].

[^note]: In the LDAP bridge `group_power_level` is the only way to assign permissions to spaces automatically generated from LDAP OrgUnits. If you define both space mapping and `group_power_level` in your configuration, `group_power_level` will **only** be used for the automatically generated spaces, it will not be taken into account for the spaces defined manually in your space mapping config.

```
spaces:
  id: root
  name: 'Company'
  groups:
    # regular users
```

```
    - externalId: 'element-users'
    # moderators
    - externalId: 'element-moderators'
      powerLevel: 50
```

In case of Power Level conflicts, the highest power level will be used. With the following configuration:

```
spaces:
  id: root
  name: 'Company'
  groups:
    - externalId: 'moderators'
      powerLevel: 50
    - externalId: 'admins'
      powerLevel: 100
```

A user who's a member of both `moderators` and `admins` will end up with Power Level of 100.

Subspaces can be configured analogically:

```
spaces:
  id: shared
  name: "Element Corp"
  groups:
  - externalId: 'matrix-mods'
    powerLevel: 50
  - externalId: ''
  subspaces:
  - id: london
    name: "London Office"
    groups:
    - externalId: 'london-matrix-mods'
      powerLevel: 50
    - externalId: 'london-employees'
```

Note for LDAP: CNs are also allowed here for backwards compatibility reasons, but **only for groups**. It is however advised to avoid using CNs and use DNs instead, since they are guaranteed to be unique across the LDAP tree. Group Sync's behaviour is undefined when mapping groups with duplicate names

```
spaces:
  id: root
  name: "Company"
  groups:
    - externalId: `dc=company` # or leave it empty with the same result
  subspaces:
    - id: engineering
      name: Engineering
      groups:
        - externalId: `ou=engineering,cn=company`
        - externalId: `cn=moderators,ou=engineering,cn=company`
          powerLevel: 50
```

Setting Up the Integration Manager With the Installer

Dimension is Deprecated

Starting with Element Enterprise Installer 2022-09.06, we are now shipping Integrator, our next generation integration manager. When installing 2022-09.06 or later, you should install Integrator. If you still have Dimension installed, please follow this step to remove it:

```
kubectl delete dimension/instance -n element-onprem
```

Now you will need to delete the two configuration files for dimension. Assuming your configuration directory is `~/.element-onpremise-config/`, the commands would be:

```
rm -rf ~/.element-onpremise-config/dimension
rm ~/.element-onpremise-config/element/dimension.json
```

Once you have finished these steps you may continue with the directions to configure integrator.

If you are on a release prior to 2022-09.06 and need documentation on dimension, please see:

Documentation Covering Installers From 2022.07.03 to 2022.09.05

On the hosting machine

- Create `integrator` directory in `~/.element-onpremise-config/`
- Copy sample file from `config-sample/integrator/integrator.yml` to the `integrator` sub-directory within your config folder
- Edit the file with the following values :
  - `integrator_fqdn`: The access address to integrator. It should match something like `integrator.<fqdn.tld>`
  - `postgres_user`: PostgreSQL username
  - `postgres_db`: PostgreSQL integrator database
  - `postgres_password`: PostgreSQL integrator password (If using postgres_create_in_cluster, set this to a random password, such as you would generate with `pwgen 32 1`)

- `postgres_fqdn`: If using postgres_create_in_cluster, set this to `postgres-integrator`, otherwise, set this to the fqdn of your postgres server.
- `postgres_create_in_cluster`: Whether or not to create the postgres as a k8s statefulset
- `postgres_data_path`: If using postgres_create_in_cluster, you should specify a path to store the postgres database.
- `jitsi_domain`: OPTIONAL. Allows you to specify the jitsi domain to be used by the jitsi widget.
- `verify_tls`: OPTIONAL. If doing a POC with self-signed certs, set this to `0`. The default is `1`.

- Restart the install script

Setting up GitLab, GitHub, JIRA and Webhooks Integrations With the Installer

In Element Enterprise On-Premise, our GitLab, GitHub, and JIRA integrations are provided by the hookshot package. This documentation explains how to configure the installer to install hookshot and then how to interact with hookshot once installed.

Configuring Hookshot with the Installer

- Copy sample file from `config-sample/hookshot/hookshot.yml` in the installer directory to `~/.element-onpremise-config/hookshot`
- Edit the file with the following values :
  - `logging_level` : The logging level
  - `hookshot_fqdn` : The adress of hookshot webhook fqdn. It should match something like `hookshot.<fqdn.tld>`
  - `passkey` : The name of the local key file. It can be generated using openssl - `openssl genrsa -out key.pem 4096`
  - `provisioning_secret` : The provisioning secret used with integration managers. Necessary for integration with integrator.
  - `bot_display_name` : The name of hookshot bot
  - `bot_avatar` : An `mxc://` uri to the hookshot bot avatar image.
  - `verify_tls` : Optional. If doing a POC with self-signed certificates, set this to 0. Defaults to 1.
  - `disallowed_ip_ranges` : Optional. A list of IP ranges to not allow connectivity to. For more information on this, please see: https://ems-docs.element.io/books/ems-knowledge-base/page/hookshot-fails-to-display-configuration-widget
- Restart the install script

Enabling GitHub Integration

On GitHub

- This bridge requires a GitHub App. You will need to create one.
- On the callback URL, set the following one : `https://<hookshot_fqdn>/oauth` and enable `Request user authorization (OAuth) during installation`
- On the webhook URL, set the following one : `https://<hookshot_fqdn>/` (don't forget the ending `/`)
- For the webhook secret, you can generate one using `pwgen 32 1` to generate one for example. Keep it somewhere safe, you'll need to to configure the bridge.
- Set the following permissions for the webhook :
  - Repository
    - Actions (read)
    - Contents (read)
    - Discussions (read & write)
    - Issues (read & write)
    - Metadata
    - Projects (read & write)
    - Pull requests (read & write)
  - Organisation
    - Team Discussions (read & write)
- Install the App

On the installation

- Copy sample file from `config-sample/hookshot/github.yml` in the installer directory to `~/.element-onpremise-config/hookshot`
- Edit the file with the following values :
  - `github_auth_id` : The AppID given in your github app page
  - `github_key_file` : The key file received via the `Generate a private key` button under `Private keys` section of the github app page.
  - `github_webhook_secret` : The webhook secret configured in the app.
  - `github_oauth_client_id` : The OAuth ClientID of the github app page.
  - `github_oauth_client_secret` : The OAuth Client Secret of the github app page.
  - `github_oauth_default_options` A mapping to enable special oauth options.
- Restart the install script

In Element's room

- Start a private conversation with the bot
- Type `github login`
- Follow the link to connect the bot to the configured app
- As an administrator of the room, invite the hookshot bot
- Promote the bot to a Moderator/Admin
- If you have setup Integrator, you can use the integration manager to add a bridge to github

Enabling GitLab integration

On GitLab

- Add a webhook under the group or the repository you are targeting
- On the webhook URL, set the following one : `https://<hookshot_fqdn>/`
- For the webhook secret, you can generate one using `pwgen 32 1` to generate one for example. Keep it somewhere safe, you'll need to to configure the bridge.
- You should add the events you wish to trigger on. Hookshot currently supports:
  - Push events
  - Tag events
  - Issues events
  - Merge request events
  - Releases events

On the installation

- Copy sample file from `config-sample/hookshot/gitlab.yml` in the installer directory to `~/.element-onpremise-config/hookshot`
- Edit the file with the following values :
  - `gitlab_instances`: A mapping of the GitLab servers
    - `git.example.org`: Replace with name of the GitLab server
      - `url`: Replace with URL of the GitLab server
  - `gitlab_webhook_secret`: The secret configured in the webhook.

In Element's room

- Start a private conversation with the bot
- Type `gitlab personaltoken instancename personaltoken` to connect to GitLab where instancename is one of the mappings of the GitLab servers (git.example.org) and personaltoken, a token generated from the GitLab admin UI

- As an administrator of the room, invite the hookshot bot
- Run the command `!hookshot gitlab project https://mydomain/my/project` to bridge a project to the room
- Type `!gl help` to list supported commands

Enabling JIRA integration

On JIRA

- This should be done for all JIRA organisations you wish to bridge. The steps may differ for SaaS and on-prem, but you need to go to the webhooks configuration page under Settings > System > Webhooks. It should point to `https://<hookshot_fqdn>/?secret=<jira_webhook_secret>`
- For the webhook secret, you can generate one using `pwgen 32 1` to generate one for example. Keep it somewhere safe, you'll need to to configure the bridge.

Enable OAuth

The JIRA service currently only supports atlassian.com (JIRA SaaS) when handling user authentication. Support for on-prem deployments is hoping to land soon.

- You'll first need to head to https://developer.atlassian.com/console/myapps/create-3lo-app/ to create a "OAuth 2.0 (3LO)" integration.
- Once named and created, you will need to:
- Enable the User REST, JIRA Platform REST and User Identity APIs under Permissions.
- Use rotating tokens under Authorisation.
- Set a callback url. This will be the public URL to hookshot with a path of /jira/oauth.
- Copy the client ID and Secret from Settings

On the installation

- Copy sample file from `config-sample/hookshot/jira.yml` in the installer directory to `~/.element-onpremise-config/hookshot`
- Edit the file with the following values :
  - `jira_webhook_secret`: The webhook secret configured
  - `jira_oauth_client_id`: If Oauth is enabled, it should point to the ClientID in Jira's App page. Else, you can keep it empty.
  - `jira_oauth_client_secret`: If Oauth is enabled, it should point to the Client secret in Jira's App page. Else, you can keep it empty.

In Element's room

- As an administrator of the room, invite the hookshot bot
- If you have setup Integrator, you can use the integration manager to add a bridge to JIRA. There is currently a limitation - it only works for public rooms.

Enabling generic webhooks integration

On the installation

- Copy sample file from `config-sample/hookshot/generic.yml` in the installer directory to `~/.element-onpremise-config/hookshot`
- Edit the file with the following values :
  - `generic_enabled`: `true` to enable it
  - `generic_allow_js_transformation_functions`: `true` if you want to enable javascript transformations
  - `generic_user_id_prefix`: Choose a prefix for the users generated by hookshot for webhooks you'll create

In Element's room

- As an administrator of the room, invite the hookshot bot
- Type `!hookshot webhook <name of the webhook>`
- The bot will answer with a URL that you can set up as a webhook.
- Please ensure that the `Content-Type` is set to the type matching what the webhook sends
- If you have setup Integrator, you can use the integration manager to add a bridge to a new webhook

Setting up Adminbot and Auditbot

Overview

Starting with Installer version 2022.07-03, we have enabled the configuration of our Adminbot and Auditbot products, which are available as add-ons to our Enterprise customers.

Adminbot allows for an Element Administrator to become admin in any existing room or space on a managed homeserver. This enables you to delete rooms for which the room administrator has left your company and other useful administration actions.

Auditbot allows you to have the ability to export any communications in any room that the auditbot is a member of, even if encryption is in use. This is important in enabling you to handle compliance requirements that require chat histories be obtainable.

This document details how to configure the Adminbot and Auditbot themselves, but you will also need to install and configure our Enterprise Admin Dashboard so that an Element Administrator can log in and then log in as the Adminbot or Auditbot and perform specific functions.

Configuring Admin Bot

Start by copying `config-sample/adminbot/adminbot.yml` into your configuration directory, by running these commands from your installer directory:

```
mkdir ~/.element-onpremise-config/adminbot
cp config-sample/adminbot/adminbot.yml ~/.element-onpremise-config/adminbot/
```

The above assumes that `~/.element-onpremise-config` is your configuration directory. Change it as necessary.

The config starts with these items:

```
bot_backup_phrase: # your secret storage backup phrase
bot_data_path: /mnt/data/adminbot
bot_data_size: 10M
```

```
enable_dm_admin: false
```

Let's discuss them:

- **bot_backup_phrase**: This is the security phrase that will guard access to your encryption keys. Do NOT share this phrase with anyone. This is required.
- **bot_data_path**: This is the directory where the bot's data will be stored. If you need to change the path, please do, but for most cases, you can leave this alone.
- **bot_data_size**: In most cases, you can leave this at 10M, but it does put a limit on the amount of data that can be written by the bot to the path.
- **enable_dm_admin**: This defaults to `false` and that behavior means that adminbot **will not** join DMs. If you want full control of DMs, simply set this to `true`.
- **verify_tls** : Optional. If doing a POC with self-signed certificates, set this to 0. Defaults to 1.

Once this configuration is in place, you can re-run the installer and watch adminbot come up and then start joining rooms on your server. You may also choose to continue configuring audit bot and then the Enterprise Admin Dashboard prior to re-running the installer.

Configuring Audit Bot

Start by copying `config-sample/auditbot/auditbot.yml` into your configuration directory, by running these commands from your installer directory:

```
mkdir ~/.element-onpremise-config/auditbot
cp config-sample/auditbot/auditbot.yml ~/.element-onpremise-config/auditbot/
```

The above assumes that `~/.element-onpremise-config` is your configuration directory. Change it as necessary.

The config starts with these items:

```
bot_backup_phrase: # your secret storage backup phrase
bot_data_path: /mnt/data/auditbot
bot_data_size: 10M

enable_dm_audit: false

### optional : the S3 bucket where to store the audit logs
#s3_bucket:
#s3_access_key_id:
```

```
#s3_secret_access_key:
#s3_key_prefix:
#s3_region:
#s3_endpoint:

### optional :  the local logfile settings.  Used if s3 bucket is not enabled.
logfile_size:  1M
logfile_keep:  3
```

Let's discuss them:

- **bot_backup_phrase**: This is the security phrase that will guard access to your encryption keys. Do NOT share this phrase with anyone. This is required.
- **bot_data_path**: This is the directory where the bot's data will be stored. If you need to change the path, please do, but for most cases, you can leave this alone.
- **bot_data_size**: In most cases, you can leave this at 10M, but it does put a limit on the amount of data that can be written by the bot to the path.
- **enable_dm_admin**: This defaults to `false` and that behavior means that adminbot **will not** join DMs. If you want full control of DMs, simply set this to `true`.
- **verify_tls** : Optional. If doing a POC with self-signed certificates, set this to 0. Defaults to 1.

Once this configuration is in place, you can re-run the installer and watch auditbot come up and then start joining rooms on your server. You may also choose to continue configuring the Enterprise Admin Dashboard prior to re-running the installer.

Adminbot Federation

On the central admin bot server

- Copy sample file from `config-sample/adminbot/central.yml` to the `adminbot` sub-directory within your config folder
- `adminbot_fqdn` : The FQDN which will be targeted by remote federated servers as the central audit server
- `remote_federated_homeservers`: A list containing every remote audited server. It contains the following variables :
  - `matrix_server`: URL of the synapse server
  - `domain_name`: Domain name from parameters.yaml (the server name part of the users mxid)
  - If the server is managed by the installer :
    - `generic_shared_secret`: The generic shared secret to get from secrets.yaml

    - `adminuser_token`: The token from the admin user, to get via `kubectl get synapseusers/adminuser-donotdelete -n element-onprem -o yaml`. It's the value of the field `status.accessToken`.
  - If the server is not managed by the installer :
  - `as_token` : The as token configured on the remote appservice file on the remote server.
  - `hs_token` : The as token configured on the remote appservice file on the remote server.
  - `adminuser_token` : An access token to an user which is server admin.

On the remote admin bot server

- Copy sample file from `config-sample/adminbot_access/access.yml` :
- Configure the following variables :
  - `central_adminbot_fqdn` : The value of `adminbot_fqdn` on the central audit bot server

Auditbot Federation

On the central auditbot server

- Copy sample file from `config-sample/auditbot/central.yml` to the `auditbot` sub-directory within your config folder
- `auditbot_fqdn` : The FQDN which will be targeted by remote federated servers as the central audit server
- `remote_federated_homeservers`: A list containing every remote audited server. It contains the following variables :
  - `matrix_server`: URL of the synapse server
  - `domain_name`: Domain name from parameters.yaml (the server name part of the users mxid)
  - If the server is managed by the installer :
    - `generic_shared_secret`: The generic shared secret to get from secrets.yaml
    - `adminuser_token`: The token from the admin user, to get via `kubectl get synapseusers/adminuser-donotdelete -n element-onprem -o yaml`. It's the value of the field `status.accessToken`.
  - If the server is not managed by the installer :
  - `as_token` : The as token configured on the remote appservice file on the remote server.
  - `hs_token` : The as token configured on the remote appservice file on the remote server.
  - `adminuser_token` : An access token to an user which is server admin.

On the remote audit bot server

- Copy sample file from `config-sample/auditbot_access/access.yml` :
- Configure the following variables :
  - `central_auditbot_fqdn` : The value of `auditbot_fqdn` on the central audit bot server

Enterprise Admin Dashboard

Please see this document on Configuring the Enterprise Admin Dashboard

Setting Up the Enterprise Admin Dashboard

Overview

Our Enterprise Admin Dashboard gives you the ability to manage users, rooms, the Adminbot, and the Auditbot. In the future, we will be expanding the functionality of this dashboard.

Configuring the Admin Dashboard

Start by copying `config-sample/synapseadminui/synapseadminui.yml` into your configuration directory, by running these commands from your installer directory:

```
mkdir ~/.element-onpremise-config/synapseadminui
cp config-sample/synapseadminui/synapseadminui.yml ~/.element-onpremise-config/synapseadminui/
```

The above assumes that `~/.element-onpremise-config` is your configuration directory. Change it as necessary.

The config has these items:

```
synapseadmin_fqdn: <admin fqdn>
admin_elementweb_fqdn: <special elementwen web admin fqdn>
```

Let's discuss them:

- **synapseadmin_fqdn**: This is an fqdn with PEM encoded SSL certificates that the installer can use to host the Enterprise Admin Dashboard.
- **admin_elementweb_fqdn**: This is an fqdn with PEM encoded SSL certificates that the installer can use to host a special Element Web Application that is used only by the Adminbot and Auditbot for the purpose of logging in these users.
- **verify_tls** : Optional. If doing a POC with self-signed certificates, set this to 0. Defaults to 1.

For each of these FQDNs, you will need to make sure that a PEM encoded `.crt` and `.key` pair are in the `certs` directory of the configuration directory.

If you are not using delegated authentication, you will also need to set one more variable in your `secrets.yml` in the configuration directory and that is:

```
adminuser_password: <password>
```

Replacing `<password>` with the actual password that you want to use to be able to login to the Admin Dashboard with the

`onprem-admin-donotdelete` user.

If you are using delegated authentication, then you will need to give synapse admin privileges to one of your users. Let's say that your user who needs to have admin is named bob@server.name. To give this user

```
kubectl exec -n element-onprem -it pods/postgres-0 -- /usr/bin/psql -d synapse -U
synapse_user -c "update users set admin = 1 where name = '@bob:server.name';"
```

Once you have done this, re-run the installer and after the pods have come up, you will be able to access the Enterprise Admin Dashboard at the provided FQDN.

Setting Up Hydrogen

Configuring Hydrogen

- Copy sample file from `config-sample/hydrogen/hydrogen.yml` in the installer directory to `CONFIG_DIRECTORY/hydrogen`
- Edit the file :

```
---
hydrogen_fqdn: "hydrogen.dev.local"


# Additional JSON that will end up in Hydrogen's config.json
extra_config: ""
```

  - `hydrogen_fqdn` is the FQDN that will be used for accessing hydrogen. It must have a PEM formatted SSL certificate as mentioned in the introduction. The crt/key pair must be in the `CONFIG_DIRECTORY/certs` directory.
  - `extra_config` is extra json config that should be injected into the hydrogen client configuration.
- Restart the install script

Setting up On-Premise Metrics

Setting up prometheus and grafana (Starting from installer 2022-08.02)

- Copy sample file from `config-sample/prometheus/prom.yml` to the `prometheus` sub-directory within your config folder
- If you want to write prometheus data to a remote prometheus instance, please define these 4 variables :
  - `remote_write_url`: The URL of the endpoint to which to push remote writes
  - `remote_write_external_labels`: The labels to add to your data, to identify the writes from this cluster
  - `remote_write_username`: The username to use to push the writes
  - `remote_write_password`: The password to use to push the writes
- You can configure which prometheus components you want to deploy :
- `deploy_prometheus`: `true` to deploy prometheus
- `deploy_node_exporter`: requires prometheus deployment. Set to `true` to gather data about the k8s nodes.
- `deploy_kube_control_plane_monitoring`: requires prometheus deployment. Set to `true` to gather data about the kube controle plane.
- `deploy_kube_state_metrics`: requires prometheus deployment. Set to `true` to gather data about kube metrics.
- `deploy_element_service_monitors`: Set to `true` to create `ServiceMonitor` resources into the K8S cluster. Set it to `true` if you want to monitor your element services stack using prometheus.
- You can choose to deploy grafana on the cluster :
  - `deploy_grafana`: `true`
  - `grafana_fqdn`: The FQDN of the grafana application
  - `grafana_data_path`: `/mnt/data/grafana`
  - `grafana_data_size`: 1G
- If doing a POC with self-signed certificates:

- `verify_tls` : Optional. If doing a POC with self-signed certificates, set this to 0. Defaults to 1.

After running the installer, open the FQDN of Grafana. The initial login user is `admin` and password is `admin`. You'll be required to set a new password, please define one secured and keep it in a safe place.

Setting Up the Telegram Bridge

Configuring Telegram bridge

On Telegram platform

- Login to my.telegram.org to get a telegram app ID and hash (get from). You should use a phone number associated to your company.

Basic config

- Copy sample file from `config-sample/telegram/telegram.yml` to the `telegram` sub-directory within your config folder
- Edit the file with the following values :
  - `postgres_create_in_cluster`: `true` to create the postgres db into the k8s cluster. On a standalone deployment, it is necessary to define the `postgres_data_path`.
  - `postgres_fqdn`: The fqdn of the postgres server. If using `postgres_create_in_cluster`, you can choose the name of the workload.
  - `postgres_data_path`: "/mnt/data/telegram-postgres"
  - `postgres_port`: 5432
  - `postgres_user`: The user to connect to the db.
  - `postgres_db`: The name of the db.
  - `postgres_password`: A password to connect to the db.
  - `telegram_fqdn`: The FQDN of the bridge for communicating with Telegram and using public login user interface.
  - `max_users`: Max number of users enabled on the bridge.
  - `bot_username`: The username of the bot for users to manage their bridge connectivity.
  - `bot_display_name`: The display name of the bot.
  - `bot_avatar`: An mx content URL to the bot avatar.
  - `admins`: The list of admins of the bridge.

- `enable_encryption`: true to allow e2e encryption in bridge.
- `enable_encryption_by_default`: true to enable by default e2e encryption on every chat created by the bridge.
- `enable_public_portal`: true to give the possibility to users to login using the bridge portal UI.
- `telegram_api_id`: The telegram API ID you got from telegram platform.
- `telegram_api_hash`: The telegram api hash you got from telegram platform.

Usage

- Talk to the telegram bot to login to the bridge. See Telegram Bridge starting at "Bridge Telegram to your Element account". Instead of addressing the bot as that document explains, use "@bot_username:domain" as per your setup.

Setting Up the Teams Bridge

Configuring Teams Bridge

Register with Microsoft Azure

You will first need to generate an "Application" to serve connect your Teams bridge with Microsoft.

- Connect to Azure on https://portal.azure.com/#blade/Microsoft_AAD_IAM/ActiveDirectoryMenuBlade/Overview to go to the Active Directory.
- Go to "Register an application screen" and register an application.
- Supported account types can be what fits your needs, but do not select "Personal Microsoft accounts"
- **Redirect URI** must be `https://<teams_fqdn>/authenticate`. You must use the type `Desktop and Mobile apps`. You don't need to check any of suggested redirection URIs.
- You should be taken to a general configuration page. Click Certificates & secrets
- Generate a **Client Secret** and copy the resulting value. The value will be your `teams_client_secret`.

Permissions

You will need to set some API permissions.

For each of the list below click Add permission > Microsoft Graph > and then set the **Delegated permissions**.

- ChannelMessage.Read.All - Delegated
- ChannelMessage.Send - Delegated
- ChatMessage.Read - Delegated
- ChatMessage.Send - Delegated
- ChatMember.Read - Delegated
- ChatMember.ReadWrite - Delegated
- Group.ReadWrite.All - Delegated
- offline_access - Delegated
- profile - Delegated
- Team.ReadBasic.All - Delegated

- User.Read - Delegated
- User.Read.All - Delegated

For each of the list below click Add permission > Microsoft Graph > and then set the **Application permissions**:

- ChannelMember.Read.All - Application
- ChannelMessage.Read.All - Application
- Chat.Create - Application
- Chat.Read.All - Application
- Chat.ReadBasic.All - Application
- Chat.ReadWrite.All - Application
- ChatMember.Read.All - Application
- ChatMember.ReadWrite.All - Application
- ChatMessage.Read.All - Application
- Group.Create - Application
- Group.Read.All - Application
- Group.ReadWrite.All - Application
- GroupMember.Read.All - Application
- GroupMember.ReadWrite.All - Application
- User.Read.All - Application

Once you are done, click **Grant admin consent**

- Go to Overview
- Copy the "Application (client) ID" as your `teams_client_id` in the config
- Copy the "Directory (tenant) ID" as the `teams_tenant_id` in the config.

Setting up the bot user

The bridge requires a Teams user to be registered as a "bot" to send messages on behalf of Matrix users. You just need to allocate one user from the Teams interface to do this.

- First, you must go to the Azure Active Directory page.
- Click users.
- Click New user.
- Ensure **Create user** is selected.
- Enter a User name ex. "matrixbridge".
- Enter a Name ex. "Matrix Bridge".
- Enter an Initial password.
- Create the user.
- Optionally, set more profile details like an avatar.
- You will now need to log in as this new bot user to set a permanent password (Teams requires you to reset the password on login).
- After logging in you should be prompted to set a new password.

- Enter the bot username and password into config under `teams_bot_username` and `teams_bot_password`

Getting the groupId

The groupId can be found by opening Teams, clicking ... on a team, and clicking "Get link to team". The groupId is included in the URL `12345678-abcd-efgh-ijkl-lmnopqrstuvw` in this example.

```
https://teams.microsoft.com/l/team/19%3XXX%40thread.tacv2/conversations?groupId=12345678-abcd-efgh-ijkl-lmnopqrstuvw&tenantId=87654321-dcba-hgfe-lkji-zyxwvutsrqpo
```

On the hosting machine

Generate teams registration keys

```
openssl genrsa -out teams.key 1024
openssl req -new -x509 -key teams.key -out teams.crt -days 365
```

Configure Teams Bridge

- Copy the sample file from `config-sample/teams/` to the `teams` sub-directory within your config folder
- Edit the files accordingly to the provider. Parameters are as follows:

```
teams_client_id: # teams app client id
teams_client_secret: # teams app secret
teams_tenant_id: # teams app tenant id
teams_bot_username: # teams bot username
teams_bot_password: # teams bot password
teams_cert_file: teams.crt
teams_cert_private: teams.key
teams_fqdn: <teams bridge fqdn>
teams_bridged_groups:
- group_id: 218b0bfe-05d3-4a63-8323-846d189f1dc1 #change me
  properties:
    autoCreateRooms:
```

```
      public: true
      powerLevelContent:
        users:
          "@alice:example.com": 100 # This will add <alice> account as admin
          "@teams-bot:example.com": 100 # the Teams bot mxid
<bot_sender_localpart>:<domain_name>
    autoCreateSpace: true
    limits:
      maxChannels: 25
      maxTeamsUsers: 25
# repeat "- group_id:" section above for each Team you want to bridge


bot_display_name: Teams Bridge Bot
bot_sender_localpart: teams-bot
enable_welcome_room: true
welcome_room_text: |
 Welcome, your Element host is configured to bridge to a Teams instance.

 This means that Microsoft Teams messages will appear on your Element
 account and you can send messages in Element rooms to have them appear
 on teams.

 To allow Element to access your Teams account, please say `login` and
 follow the steps to get connected. Once you are connected, you can open
 the   Explore Rooms dialog to find your Teams rooms.
# namespaces_prefix_user: OPTIONAL: default to _teams_
# namespaces_prefix_aliases: OPTIONAL: default to teams_
```

- For each Bridged Group, you will need to set a group_id and some properties found in the config sample.

Setting Up the IRC Bridge

Overview

The IRC bridge allows you to bridge IRC servers into your Element server.

To configure the irc bridge, beging by copying `config-sample/ircbridge/bridge.yml` to `CONFIG_DIR/ircbridge/bridge.yml`. Then edit the file and set the following settings:

- `key_file: passkey.pem` To generate the passkey.pem file, please run the following in the `CONFIG_DIR/ircbridge` directory: `openssl genpkey -out passkey.pem -outform PEM -algorithm RSA -pkeyopt rsa_keygen_bits:2048`
- `postgres_fqdn: ircbridge-postgres` Use `ircbridge-postgres` if using `postgres-create-in-cluster` otherwise point this at your external database.
- `postgres_user: ircbridge` Leave this if you are using `postgres-create-in-cluster`.
- `postgres_db: ircbridge` Leave this if you are using `postgres-create-in-cluster`.
- `postgres_password: postgres_password` Set this to either your password for the user connecting to an existing database, or if using `postgres_create_in_cluster`, set this to a new password with `pwgen 32 1`.
- `# postgres_create_in_cluster: true` # uncomment if you want the installer to install postgresql for you. Not supported with the multi-node installer, where you must use an external postgres.
- `postgres_port` Can be used to specify a non-standard port. 5432 is used if not specified. Optional
- `postgres_sslmode` Can be used to specify the sslmode for the Postgres connection. Options are 'disable', 'no-verify' or 'verify-full'. 'disable' is used if not specified. Optional
- Now specify a list of Matrix IDs that have admin access to the IRC bridge such as:

```
admins:
- "@adminuser:dev.local"
- "@adminuser2:dev2.local"
```

- `enable_presence: true` This determines if presence is presented to IRC or not.
- `drop_matrix_messages_after_seconds: 0`
- `bot_username: "ircbridgebot"` The name of the bot.
- `enable_ident: false` Whether or not to enable IRC ident.
- `ident_port_type: # HostPort or NodePort` Required if enabling ident.
- `ident_port_number: 10230` Required if enabling ident.
- `logging_level: info` Set the default logging level of the bridge.
- `enable_provisioning: true`

Next, we have the provisioning rules section, which will make sure that rooms are not bridged if a match is made on these rules. This is useful for preventing bad actors on Matrix from flooding IRC. This section looks like:

```
provisioning_rules:
 # The bridge checks the joined members of a propective room and checks to see
 # if any users matching these regex sets are in the room. `exempt` users never
 # match, and will be ignored. If any user matches `conflict`, the room will not
 # be allowed to be bridged until the user is removed. Both sets take a regular expression.
 userIds:
   exempt:
     # These users never conflict, even if matching
     - "@doubleagent:badguys.com"
   conflict:
     # These users will deny a room from being bridged.
     - "@.*:badguys.com"
provisioning_room_limit: 50
```

- `rmau_limit: 100` Set this to the maximum number of remote monthly active users that you would like to allow in a bridged IRC room.
- `users_prefix: "irc_"` Set a user prefix for irc users.
- `alias_prefix: "irc_"` Set an alias prefix for irc users.
- `address: irc.someserver.net` The adress of the irc server to bridge. Now for the above IRC server, we have a set of parameters that can be set:
- `name: "Server Name"` The server name to show on the bridge. Now below that, you'll see the botConfig with these parameters:
- `enabled: true` Leave this on.
- `nick: "MatrixBot"` Nick of the bridge bot
- `username: "matrixbot"` Username of the bridge bot.
- `password: "some_password"` Password of the bridge bot. Generate this with `pwgen 32 1`

For other settings that can also be applied to this config file, please see:https://github.com/matrix-org/matrix-appservice-irc/blob/develop/config.sample.yaml#L52

Connecting to the Bridge

1. From Element, send a DM to the bridge bot `/msg @ircbridgebot:element.local` where `element.ocal` is you servers domain name
2. Send the bridge `!whois` to see if you can get logged in to the IRC network

Setting Up the SIP Bridge

Configuring SIP bridge

Basic config

- Copy sample file from `config-sample/sipbridge/sipbridge.yml` to the `sipbridge` sub-directory within your config folder
- Edit the file with the following values :
  - `postgres_create_in_cluster`: `true` to create the postgres db into the k8s cluster. On a standalone deployment, it is necessary to define the `postgres_data_path`.
  - `postgres_fqdn`: The fqdn of the postgres server. If using `postgres_create_in_cluster`, you can choose the name of the workload.
  - `postgres_data_path`: "/mnt/data/sipbridge-postgres"
  - `postgres_port`: 5432
  - `postgres_user`: The user to connect to the db.
  - `postgres_db`: The name of the db.
  - `postgres_password`: A password to connect to the db.
  - `port_type`: `HostPort` or `NodePort` depending on which kind of deployment you want to use. On standalone deployment, we advise you to use `HostPort` mode.
  - `port`: The port on which to configure the SIP protocol. On `NodePort` mode, it should be in kubernetes range:
  - `enable_tcp`: `true` to enable TCP SIP.
  - `pstn_gateway`: The hostname of the PSTN Gateway.
  - `external_address`: The external address of the SIP Bridge
  - `proxy` : The address of the SIP Proxy
  - `user_agent`: A user agent for the sip bridge.
  - `user_avatar`: An MXC url to the sip bridge avatar. Don't define it if you have not uploaded any avatar.
  - `encryption_key`: A 32 character long secret used for encryption. Generate this with `pwgen 32 1`

Setting Up the XMPP Bridge

Configuring the XMPP Bridge

The XMPP bridge relies on the xmpp "component" feature. It is an equivalent of matrix application services. You need to configure an XMPP Component on an XMPP Server that the bridge will use to bridge matrix and xmpp user.

On the hosting machine

- Copy sample file from `config-sample/xmpp/xmpp.yml` to the `xmpp` sub-directory within your config folder
- Edit the file with the following values :
  - `xmpp_service`: XMPP Address of the service endpoint.
  - `xmpp_domain`: The XMPP FQDN witht eh External Component subdomain (i.e. element.xmpp.example.com)
  - `bot_username`: The xmpp bot username on matrix
  - `alias_prefix`: The prefix for bridged aliases
  - `user_prefix`: The prefix for bridged users
  - `enable_portals_gateway`: `true` to enable portals.
  - `xmpp_component_password`: Xmpp component password
  - `postgres_create_in_cluster`: `true` if you want the installer to automatically set up postgres. Requiers `postgres_data_path` if using this.
  - `postgres_fqdn`: PostgreSQL server fqdn or ip
  - `postgres_user`: PostgreSQL username
  - `postgres_db`: PostgreSQL database
  - `postgres_port`: PostgreSQL port, default to 5432
  - `postgres_password`: PostgreSQL password
  - `postgres_create_in_cluster`: Whether or not to create the postgres as a k8s statefulset
- Restart the installer

In all the examples below the follow are set

`domain_name` from `parameters.yml` : `element.local` XMPP Server FQDN: xmpp.example.com XMPP External Component/`xmpp_domain`: `element.xmpp.example.com`

Prosody Example

If you are configuring prosody, you need the following component configuration (for the sample xmpp server, `element.xmpp.example.com`):

```
    Component "element.xmpp.example.com"
        ssl = {
          certificate = "/etc/prosody/certs/tls.crt";
          key = "/etc/prosody/certs/tls.key";
        }
      component_secret = "eeb8choosaim3oothaeGh0aequiop4ji"
```

And then with that configured, you would pass the following into `xmpp.yml`:

```
xmpp_service:  xmpp://xmpp.example.com:5347
xmpp_domain:  "element.xmpp.example.com"  # external component subdomain
xmpp_component_password:  eeb8choosaim3oothaeGh0aequiop4ji # xmpp component password
```

Note: We've used `pwgen 32 1` to generate the `component_secret`.

Joining an XMPP Room

Once you have the XMPP bridge up, you need to map an XMPP room to a Matrix ID. To do this, if the room on XMPP is named:

```
#iwotevo@conference.xmpp.lab.element.com
```

(conference is the fqdn of the component's hosting rooms on our xmpp test instance)

then on Matrix, you would join:

```
#_xmpp_iwotevo_conference.xmpp.example.com:element.local
```

The command to do that from within the Element client would be: (assuming your homeserver domain is example.com)

```
/join #_xmpp_iwotevo_conference.xmpp.example.com:element.local
```

Joining a Matrix room from XMPP

If the Element/Matrix room is public you should be able to query the room list at the external component server address(Ex: element.xmpp.example.com)

The Matrix room at alias `#roomname:element.local` maps to `#roomname#element.local@element.xmpp.example.com` on the XMPP server xmpp.example.com if yout `xmpp_domain: element.xmpp.example.com`

Element		XMPP
#**roomname**:element.local (native Matrix room)	→	#**roomname** #element.local@element.xmpp.example.com (bridged into XMPP)
#_xmpp_**roomname** _conference.xmpp.example.com:element.local (bridged into Matrix/Element)	←	#**roomname** @conference.xmpp.example.com (native XMPP room)

Single Node Installs: Storage and Backup Guidelines

General storage recommentations for single-node instances

- `/mnt` (or a common root for all `<component>_data_path` variables) should be a distinct mount point
  - Ideally this would have an independent lifecycle from the server itself
  - Ideally this would be easily snapshot-able, either at a filesystem level or with the backing storage

Adminbot storage:

- Files stored with uid=10006/gid=10006, sample config uses `/mnt/data/adminbot` for single-node instances
  - The backing path for single node instances can be changed by setting `bot_data_path` in the `adminbot` config directory
- Storage space required is proportional to the number of user devices on the server. 1GB is sufficient for most servers
  - The size of the PVC can be changed by setting `bot_data_size` in the `adminbot` config directory

Auditbot storage:

- Files stored with uid=10006/gid=10006, sample config uses `/mnt/data/auditbot` for single-node instances
  - The backing path for single node instances can be changed by setting `bot_data_path` in the `auditbot` config directory
- Storage space required is proportional to the number of events tracked. 1GB is sufficient with the sample config `logfile_size` / `logfile_keep` values

    - The size of the PVC can be changed by setting `bot_data_size` in the `auditbot` config directory

Dimension storage :

- Main:
  - File stored with uid=10005/gid=1000, sample config uses `/mnt/dimension` for single-node instances
    - The backing path for single node instances can be changed by setting `bot_data_path` in the `dimension` config directory
  - Storage space is constant to store a single file. 10M is sufficient for every server
    - The size of the PVC can be changed by setting `bot_data_size` in the `dimension` config directory
- Postgres (in-cluster):
  - Files stored with uid=999/gid=999, sample config does not specify a default path for single-node instances
    - The backing path for single node instances can be changed by setting `postgres_data_path` in the `dimension` config directory
  - Storage space is proportional to the number of integration instances. 5GB is sufficient for most servers
    - The size of the PVC can be changed by setting `postgres_storage_size` in the `dimension` directory folder

Synapse storage:

- Media:
  - File stored with uid=10991/gid=10991, sample config uses `/mnt/data/synapse-media` for single-node instances
    - The backing path for single node instances can be changed by setting `media_host_data_path` in `parameters.yml`
  - Storage space required grows with the number and size of uploaded media. 50GB is used as a starting point for PoC but can easily be exceeded depending on your use-case
    - The size of the PVC can be changed by setting `media_size` in `parameters.yml`

Postgres (in-cluster) storage:

- Files stored with uid=999/gid=999, sample config uses `/mnt/data/synapse-postgres` for single-node instances
  - The backing path for single node instances can be changed by setting `postgres_data_path` in `parameters.yml`

- Storage space is proportional to the activity on the homeserver. 5GB is used as a starting point for PoC but can easily be exceeded depending on traffic
  - The size of the PVC can be changed by setting `postgres_storage_size` in `parameters.yml`

Backup Guidance:

- **Adminbot**:
  - Backups should be made by taking a snapshot of the PV (ideally) or rsyncing the backing directory to backup storage
- **Auditbot**:
  - Backups should be made by taking a snapshot of the PV (ideally) or rsyncing the backing directory to backup storage
- **Dimension**:
  - Backups should be made by taking a snapshot of the PV (ideally) or rsyncing the backing directory to backup storage
- **Synapse Media**:
  - Backups should be made by taking a snapshot of the PV (ideally) or rsyncing the backing directory to backup storage
- **Postgres (in-cluster)**:
  - Backups should be made by `kubectl -n element-onprem exec -it postgres-synapse-0 -- sh -c 'pg_dump -U $POSTGRES_USER $POSTGRES_DB' > synapse_postgres_backup_$(date +%Y%m%d-%H%M%S).sql`
- **Postgres (external)**:
  - Backup procedures as per your DBA
- **Configuration**:
  - Please ensure that your entire configuration directory (that contains at least `parameters.yml` & `secrets.yml` but may also include other sub-directories & configuration files) is regularly backed up
    - The suggested configuration path in Element's documentation is `~/.element-onpremise-config` but could be anything. It is whatever directory you used with the installer.

On-Premise Support Scope of Coverage

For Element Enterprise On-Premise, we support the following:

- Installation and Operation (Configuring the Installer, Debugging Issues)
- Synapse Usage/Configuration/Prioritised Bug Fixes
- Element Web Usage/Configuration/Prioritised Bug Fixes
- Integrations
  - Delegated Auth (e.g. SAML/LDAP) (Add-on)
  - Group Sync (LDAP, AD Graph API, SCIM supported) (Add-on)
  - Github / Gitlab
  - JIRA
  - Webhooks
  - Jitsi
  - Chatterbox (Add-on)
  - Adminbot (Add-on)
  - Auditbot (Add-on)

For Element On-Premise, we support the following:

- Installation and Operation (Configuring the Installer, Debugging Issues)
- Synapse Usage/Configuration/Prioritised Bug Fixes
- Element Web Usage/Configuration/Prioritised Bug Fixes
- Integrations
  - Github / Gitlab
  - JIRA
  - Webhooks
  - Jitsi

The following items are ***not*** included in support coverage:

- General Infrastructure Assistance
- K8s Assistance
- Operating System Support
- Postgresql Database Support

For single node setups, the following also applies:

- Element does not support deployment to a microk8s that was not installed by our installer.
- Element does not provide a backup solution.
- Element does not provide support for any underlying storage.

For kubernetes deployments, the following also applies:

- Element does not support deploying the installer created postgresql in a kubernetes environment.
- Element requires that you deploy postgresql separately in a kubernetes environment, external to your Element deployment.

Troubleshooting

Introduction to Troubleshooting

Troubleshooting the Element Installer comes down to knowing a little bit about kubernetes and how to check the status of the various resources. This guide will walk you through some of the initial steps that you'll want to take when things are going wrong.

install.sh problems

Sometimes there will be problems when running the ansible-playbook portion of the installer. When this happens, you can increase the verbosity of ansible logging by editing `.ansible.rc` in the installer directory and setting:

```
export ANSIBLE_DEBUG=true
export ANSIBLE_VERBOSITY=4
```

and re-running the installer. This will generate quite verbose output, but that typically will help pinpoint what the actual problem with the installer is.

Problems post-installation

Checking Pod Status and Getting Logs

- In general, a well-functioning Element stack has at it's minimum the following containers (or pods in kubernetes language) running:

```
[user@element2 ~]$ kubectl get pods -n element-onprem
NAME                                      READY   STATUS    RESTARTS      AGE
instance-synapse-main-0                   1/1     Running   4 (27h ago)   6d21h
postgres-0                                1/1     Running   2 (27h ago)   6d21h
app-element-web-688489b777-v7l2m          1/1     Running   6 (27h ago)   6d22h
server-well-known-55bdb6b66-m8px6         1/1     Running   2 (27h ago)   6d21h
```

```
instance-synapse-haproxy-554bd57975-z2ppv   1/1     Running   3 (27h ago)   6d21h
```

The above `kubectl get pods -n element-onprem` is the first place to start. You'll notice in the above, all of the pods are in the `Running` status and this indicates that all should be well. If the state is anything other than "Running" or "Creating", then you'll want to grab logs for those pods. To grab the logs for a pod, run:

```
kubectl logs -n element-onprem <pod name>
```

replacing `<pod name>` with the actual pod name. If we wanted to get the logs from synapse, the specific syntax would be:

```
kubectl logs -n element-onprem instance-synapse-main-0
```

and this would generate logs similar to:

```
 2022-05-03 17:46:33,333 - synapse.util.caches.lrucache - 154 - INFO -
LruCache._expire_old_entries-2887 - Dropped 0 items from caches
2022-05-03 17:46:33,375 - synapse.storage.databases.main.metrics - 471 - INFO -
generate_user_daily_visits-289 - Calling _generate_user_daily_visits
2022-05-03 17:46:58,424 - synapse.metrics._gc - 118 - INFO - sentinel - Collecting
gc 1
2022-05-03 17:47:03,334 - synapse.util.caches.lrucache - 154 - INFO -
LruCache._expire_old_entries-2888 - Dropped 0 items from caches
2022-05-03 17:47:33,333 - synapse.util.caches.lrucache - 154 - INFO -
LruCache._expire_old_entries-2889 - Dropped 0 items from caches
2022-05-03 17:48:03,333 - synapse.util.caches.lrucache - 154 - INFO -
LruCache._expire_old_entries-2890 - Dropped 0 items from caches
```

- Again, for every pod not in the `Running` or `Creating` status, you'll want to use the above procedure to get the logs for Element to look at.
- If you don't have any pods in the `element-onprem` namespace as indicated by running the above command, then you should run:

```
[user@element2 ~]$ kubectl get pods -A
NAMESPACE            NAME                                        READY   STATUS
RESTARTS       AGE
container-registry   registry-5f697bb7df-dbzpq                   1/1     Running
6 (27h ago)    6d22h
kube-system          dashboard-metrics-scraper-69d9497b54-hdrdq  1/1     Running
6 (27h ago)    6d22h
kube-system          hostpath-provisioner-7764447d7c-jckkc       1/1     Running
11 (17h ago)   6d22h
element-onprem       instance-synapse-main-0                     1/1     Running
```

```
4 (27h ago)    6d22h
element-onprem       postgres-0                                  1/1     Running
2 (27h ago)    6d22h
element-onprem       app-element-web-688489b777-v7l2m            1/1     Running
6 (27h ago)    6d22h
element-onprem       server-well-known-55bdb6b66-m8px6           1/1     Running
2 (27h ago)    6d21h
kube-system          calico-kube-controllers-6966456d6b-x4scn    1/1     Running
6 (27h ago)    6d22h
element-onprem       instance-synapse-haproxy-554bd57975-z2ppv   1/1     Running
3 (27h ago)    6d21h
kube-system          calico-node-l28tp                           1/1     Running
6 (27h ago)    6d22h
kube-system          coredns-64c6478b6c-h5jp4                    1/1     Running
6 (27h ago)    6d22h
ingress              nginx-ingress-microk8s-controller-n6wmk     1/1     Running
6 (27h ago)    6d22h
operator-onprem      osdk-controller-manager-5f9d86f765-t2kn9    2/2     Running
9 (17h ago)    6d22h
kube-system          metrics-server-679c5f986d-msfc5             1/1     Running
6 (27h ago)    6d22h
kube-system          kubernetes-dashboard-585bdb5648-vrn42       1/1     Running
10 (17h ago)   6d22h
```

- This is the output from a healthy system, but if you have any of these pods not in the `Running` or `Creating` state, then please gather logs using the following syntax:

```
kubectl logs -n <namespace> <pod name>
```

- So to gather logs for the kubernetes ingress, you would run:

```
kubectl logs -n ingress nginx-ingress-microk8s-controller-n6wmk
```

  and you would see logs similar to:

```
I0502 14:15:08.467258       6 leaderelection.go:248] attempting to acquire leader
lease ingress/ingress-controller-leader...
I0502 14:15:08.467587       6 controller.go:155] "Configuration changes detected,
backend reload required"
I0502 14:15:08.481539       6 leaderelection.go:258] successfully acquired lease
ingress/ingress-controller-leader
I0502 14:15:08.481656       6 status.go:84] "New leader elected" identity="nginx-
```

```
ingress-microk8s-controller-n6wmk"
I0502 14:15:08.515623       6 controller.go:172] "Backend successfully reloaded"
I0502 14:15:08.515681       6 controller.go:183] "Initial sync, sleeping for 1
second"
I0502 14:15:08.515705       6 event.go:282] Event(v1.ObjectReference{Kind:"Pod",
Namespace:"ingress", Name:"nginx-ingress-microk8s-controller-n6wmk", UID:"548d9478-
094e-4a19-ba61-284b60152b85", APIVersion:"v1", ResourceVersion:"524688",
FieldPath:""}): type: 'Normal' reason: 'RELOAD' NGINX reload triggered due to a
change in configuration
```

Again, for all pods not in the `Running` or `Creating` state, please use the above method to get log data to send to Element.

Other Commands of Interest

Some other commands that may yield some interesting data while troubleshooting are:

- **Verify DNS names and IPs in certificates**
  In the `certs` directory under the configuration directory, run:

  ```
  for i in $(ls *crt); do echo $i && openssl x509 -in $i -noout -text | grep DNS; done
  ```

  This will give you output similar to:

  ```
  local.crt
                DNS:local, IP Address:192.168.122.118, IP Address:127.0.0.1
  synapse2.local.crt
                DNS:synapse2.local, IP Address:192.168.122.118, IP Address:127.0.0.1
  ```

  and this will allow you to verify that you have the right host names and IP addresses in your certificates.
- **Show hostname to IP mappings from within a pod**
  Run:

  ```
  kubectl exec -n element-onprem <pod_name> -- getent hosts
  ```

  and you will see output similar to:

  ```
  127.0.0.1       localhost
  127.0.0.1       localhost ip6-localhost ip6-loopback
  10.1.241.30     instance-hookshot-0.instance-hookshot.element-
  onprem.svc.cluster.local instance-hookshot-0
  192.168.122.5   ems.onprem element.ems.onprem hs.ems.onprem adminbot.ems.onprem
  auditbot.ems.onprem integrator.ems.onprem hookshot.ems.onprem admin.ems.onprem
  ```

```
eleweb.ems.onprem
```

This will help you troubleshoot host resolution.

- **Show all persistent volumes and persistent volume claims for the `element-onprem` namespace:**

```
kubectl get pv -n element-onprem
```

This will give you output similar to:

```
NAME                                       CAPACITY   ACCESS MODES   RECLAIM
POLICY   STATUS   CLAIM                                STORAGECLASS        REASON
AGE
pvc-9fc3bc29-2e5d-4b88-a9cd-a4c855352404   20Gi       RWX
Delete           Bound    container-registry/registry-claim   microk8s-
hostpath            55d
synapse-media                              50Gi       RWO
Delete           Bound    element-onprem/synapse-media        microk8s-
hostpath            7d
postgres                                   5Gi        RWO
Delete           Bound    element-onprem/postgres             microk8s-
hostpath            7d
```

- **Show the synapse configuration:**
For installers prior to 2022-05.06, use:

```
kubectl describe cm -n element-onprem instance-synapse-shared
```

and this will return output similar to:

```
send_federation: True
start_pushers: True
turn_allow_guests: true
turn_shared_secret: n0t4ctuAllymatr1Xd0TorgSshar3d5ecret4obvIousreAsons
turn_uris:
- turns:turn.matrix.org?transport=udp
- turns:turn.matrix.org?transport=tcp
turn_user_lifetime: 86400000
```

For the 2022-05.06 installer and later, use:

```
kubectl -n element-onprem get secret synapse-secrets -o yaml 2>&1 | grep shared.yaml
| awk -F 'shared.yaml: ' '{print $2}' - | base64 -d
```

and you will get output similar to the above.

- **Show the Element Web configuration:**

```
kubectl describe cm -n element-onprem app-element-web
```

and this will return output similar to:

```
config.json:
----
{
    "default_server_config": {
        "m.homeserver": {
            "base_url": "https://synapse2.local",
            "server_name": "local"
        }
  },
  "dummy_end": "placeholder",
  "integrations_jitsi_widget_url":
"https://dimension.element2.local/widgets/jitsi",
  "integrations_rest_url": "https://dimension.element2.local/api/v1/scalar",
  "integrations_ui_url": "https://dimension.element2.local/element",
  "integrations_widgets_urls": [
      "https://dimension.element2.local/widgets"
  ]
}
```

- **Show the nginx configuration for Element Web: (If using nginx as your ingress controller in production or using the PoC installer.)**

```
kubectl describe cm -n element-onprem app-element-web-nginx
```

and this will return output similar to:

```
  server {
      listen       8080;

      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 'self'";
      add_header X-Robots-Tag "noindex, nofollow, noarchive, noimageindex";

      location / {
```

```
        root   /usr/share/nginx/html;
        index  index.html index.htm;

        charset utf-8;
    }
}
```

- **Check list of active kubernetes events:**

```
kubectl get events -A
```

You will see a list of events or the message `No resources found`.

- Show the state of services in the `element-onprem` namespace:

```
kubectl get services -n element-onprem
```

This should return output similar to:

```
NAME                              TYPE        CLUSTER-IP       EXTERNAL-IP
PORT(S)                   AGE
postgres                          ClusterIP   10.152.183.47    <none>
5432/TCP                  6d23h
app-element-web                   ClusterIP   10.152.183.60    <none>
80/TCP                    6d23h
server-well-known                 ClusterIP   10.152.183.185   <none>
80/TCP                    6d23h
instance-synapse-main-headless    ClusterIP   None             <none>
80/TCP                    6d23h
instance-synapse-main-0           ClusterIP   10.152.183.105   <none>
80/TCP,9093/TCP,9001/TCP  6d23h
instance-synapse-haproxy          ClusterIP   10.152.183.78    <none>
80/TCP                    6d23h
```

- **Show the status of the stateful sets in the `element-onprem` namespace:**

```
kubectl get sts -n element-onprem
```

This should return output similar to:

```
NAME                   READY   AGE
postgres               1/1     6d23h
instance-synapse-main  1/1     6d23h
```

- **Show deployments in the `element-onprem` namespace:**

```
kubectl get deploy -n element-onprem
```

This will return output similar to:

```
NAME                        READY   UP-TO-DATE   AVAILABLE   AGE
app-element-web             1/1     1            1           6d23h
server-well-known           1/1     1            1           6d23h
instance-synapse-haproxy    1/1     1            1           6d23h
```

- **Show the status of all namespaces:**

```
kubectl get namespaces
```

which will return output similar to:

```
NAME                 STATUS   AGE
kube-system          Active   20d
kube-public          Active   20d
kube-node-lease      Active   20d
default              Active   20d
ingress              Active   6d23h
container-registry   Active   6d23h
operator-onprem      Active   6d23h
element-onprem       Active   6d23h
```

- **View the MAU Settings in Synapse:**

```
kubectl get  -n element-onprem secrets/synapse-secrets -o yaml | grep -i shared.yaml
-m 1| awk -F ': ' '{print $2}' - | base64 -d
```

which will return output similar to:

```
# Local custom settings
mau_stats_only: true

limit_usage_by_mau: False
max_mau_value: 1000
mau_trial_days: 2

mau_appservice_trial_days:
  chatterbox: 0

enable_registration_token_3pid_bypass: true
```

- **Redeploy the micro8ks setup**
  It is possible to redeploy microk8s by running the following command as root:

  ```
  snap remove microk8s
  ```

  This command does remove all microk8s pods and related microk8s storage volumes. Once this command has been run, you need to reboot your server. Add `--purge` flag to remove the data if disk usage is a concern.
  After the reboot, you can re-run the installer and have it re-deploy microk8s and Element Enterprise On-Premise for you.

Node-based pods failing name resoution

```
05:03:45:601 ERROR [Pipeline] Unable to verify identity configuration for bot-auditbot:
Unknown errcode Unknown error
05:03:45:601 ERROR [Pipeline] Unable to verify identity.  Stopping
matrix-pipe encountered an error and has stopped Error:  getaddrinfo EAI_AGAIN
synapse.prod.ourdomain
    at GetAddrInfoReqWrap.onlookup [as oncomplete] (node:dns:84:26) {
  errno:  -3001,
  code:  'EAI_AGAIN',
  syscall:  'getaddrinfo',
  hostname:  'synapse.prod.ourdomain'
}
```

To see what Hosts are set, try:

```
kubectl exec -it -n element-onprem <pod name> getent hosts
```

So to do this on the adminbot-pipe-0 pod, it would look like:

```
kubectl exec -it -n element-onprem adminbot-pipe-0 getent hosts
```

and return output similar to:

```
127.0.0.1       localhost
127.0.0.1       localhost ip6-localhost ip6-loopback
10.1.241.27     adminbot-pipe-0
192.168.122.5   ems.onprem element.ems.onprem hs.ems.onprem adminbot.ems.onprem
auditbot.ems.onprem integrator.ems.onprem hookshot.ems.onprem admin.ems.onprem
eleweb.ems.onprem
```

Node-based pods failing SSL

```
2023-02-06 15:42:04 ERROR: IrcBridge Failed to fetch roomlist from joined rooms: Error:
unable to verify the first certificate. Retrying
MatrixHttpClient (REQ-13) Error: unable to verify the first certificate
at TLSSocket.onConnectSecure (_tls_wrap.js:1515:34)
at TLSSocket.emit (events.js:400:28)
at TLSSocket.emit (domain.js:475:12)
at TLSSocket._finishInit (_tls_wrap.js:937:8),
at TLSWrap.ssl.onhandshakedone (_tls_wrap.js:709:12) {
code: 'UNABLE_TO_VERIFY_LEAF_SIGNATURE
```

Drop into a shell on the pod

```
kubectl exec -it -n element-onprem adminbot-pipe-0 -- /bin/sh
```

Check it's abililty to send a request to the Synapse server

```
node

require=("http")
request(https://synapse.server/)
```

Archived Documentation Repository

Documentation Covering Installers From 2022.07.03 to 2022.09.05

element-on-premise-documentation-0703-0905.pdf

Documentation covering v1 and installers prior to 2022-07.03

element-on-premise-documentation-july28-2022.pdf