How to Install a POC Environment

Overview

Our Element Enterprise PoC Installer can handle the installation of Element Proof of Concept (POC) environments. Our standard POC environment is a single node server with microk8s running that we deploy our Element Enterprise Operator to, resulting in a fully functioning Synapse server with Element Web that can be used to conduct a POC. On-premise production deployments use the same installer and operator, but are intended to be deployed into a full kubernetes environment.

POC installations are not intended to be run for production purposes. You should plan on having a different installation for your production environment. The settings that you use with the installer will carry over for your production install, but your rooms and spaces will not.

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

- Hostnames/DNS
- Machine Size
- Operating System
- Users
- Network Specifics
- Postgresql Database
- TURN Server
- SSL Certificates
- Extra configuration items

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

Hostnames/DNS

You will need hostnames for the following pieces of infrastructure:

- Element Server (Required)

- Synapse Server (Required)
- Dimension Server (Required if you plan to use hookshot)
- Hookshot Server (Required if you need jira, gitlab, or github integrations)

These hostnames must resolve to the appropriate IP addresses. If you have a proper DNS server with records for these hostnames in place, then you will be good to go.

`/etc/hosts` may be used as an alternative to proper DNS in a POC scenario only. In this case, you will need entries similar to:

```
192.168.122.39 element.local element
192.168.122.39 synapse.local synapse
192.168.122.39 dimension.local dimension
192.168.122.39 hookshot.local hookshot
192.168.122.39 local
```

In the absence of proper DNS, for this to work in microk8s, you will also need to add the following to your parameters.yml: *(This was added in installer version 2022-05.03. If you have an installer prior to this and need this functionality, please update.)*

```
host_aliases:
  - ip: "192.168.122.39"
    hostnames:
      - "element.local"
      - "synapse.local"
      - "hookshot.local"
      - "dimension.local"
```

Machine Size

For running a proof of concept with our installer, we support only the x86_64 architecture and recommend the following minimums:

- No federation: 4 vCPUs/CPUS and 16GB RAM
- Federation: 8 vCPUs/CPUS and 32GB RAM

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-8.5-aarch64-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
```

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 python39-pip -y
sudo yum install
https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm -y
sudo alternatives --set python3 /usr/bin/python3.9
```

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

```
*               soft    nofile  4096
*               hard    nofile  16384
```

Further Pre-requisites

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
```

Please run the following commands to create the `/mnt/data` directory and install the `python3-signedjson` and `pwgen` packages which will be used during the configuration of the installer.

The `/mnt/data` directory should have at least 50 GB of space.

```
sudo mkdir /mnt/data
sudo mkdir /mnt/data/synapse-media
```

If you will be letting the installer install the Postgres database for you, also do:

```
sudo mkdir /mnt/data/synapse-postgres
```

Ubuntu:

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

EL:

```
sudo yum install make gcc python39-devel pwgen -y
```

EL: (as a normal user)

```
pip3 install signedjson --user
```

Network Specifics

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

- Port 80 TCP
- Port 443 TCP

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 (by enabling the `http` and `https` services) and enable masquerading:

```
sudo firewall-cmd --add-service={http,https} --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 make sure that the following host variables are set:

Ubuntu Specific Directions

If your company's proxy is `http://corporate.proxy:3128`, you would edit `/etc/environment` and add the following lines:

```
HTTPS_PROXY=http://corporate.proxy:3128
HTTP_PROXY=http://corporate.proxy:3128
https_proxy=http://corporate.proxy:3128
http_proxy=http://corporate.proxy:3128
NO_PROXY=10.1.0.0/16,10.152.183.0/24,127.0.0.1
no_proxy=10.1.0.0/16,10.152.183.0/24,127.0.0.1
```

The IP Ranges specified to `NO_PROXY` and `no_proxy` are specific to the microk8s cluster and prevent microk8s traffic from going over the proxy.

EL Specific Directions

Using the same example of having a company proxy at `http://corporate.proxy:3128`, you would edit `/etc/profile.d/http_proxy.sh` and add the following lines:

```
export HTTP_PROXY=http://corporate.proxy:3128
export HTTPS_PROXY=http://corporate.proxy:3128
export http_proxy=http://corporate.proxy:3128
export https_proxy=http://corporate.proxy:3128
export NO_PROXY=10.1.0.0/16,10.152.183.0/24,127.0.0.1
export no_proxy=10.1.0.0/16,10.152.183.0/24,127.0.0.1
```

The IP Ranges specified to `NO_PROXY` and `no_proxy` are specific to the microk8s cluster and prevent microk8s traffic from going over the proxy.

In Conclusion

You will need to log out and back in for the environment variables to be re-read after setting them. If you already have microk8s running, you will need to issue:

```
microk8s.stop
microk8s.start
```

to have it reload the new environment variables.

If you need to use an authenticated proxy, then the URL schema for both EL and Ubuntu is as follows:

`protocol://user:password@host:port`

So if your proxy is `corporate.proxy` and listens on port 3128 without SSL and requires a username of `bob` and a password of `inmye1em3nt` then your url would be formatted:

`http://bob:inmye1em3nt@corporate.proxy:3128`

For further help with proxies, we suggest that you contact your proxy administrator or operating system vendor.

Users

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:

On Ubuntu:

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

On EL:

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

Unpacking the Installer

Please make sure that you unpack `element-enterprise-installer` onto your POC 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 PoC 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, we recommend installing `coturn`. Instructions on how to do that are available here: https://github.com/matrix-org/synapse/blob/master/docs/turn-howto.md (Note: On EL, you can do `yum install coturn -y`.)

Under "Synapse Setup" in the above instructions, you'll see what to change on the config. With the installer, you can 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 how you installed the TURN server. This will allow the installer to configure synapse to use your TURN server.

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 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 `.crt` and `.key` files in the certs directory under the installer directory. Certificates in the certs directory must take the form of `fqdn.cert` and `fqdn.key`.

Self-signed certificates with mkcert

The following instructions will enable you to use a tool called mkcert to generate self-signed certificates. Element nor Canonical ship this tool and so these directions are provided as one example of how to get self-signed certificates.

Ubuntu:

```
sudo apt-get install wget libnss3-tools
```

EL:

```
sudo yum install wget nss-tools -y
```

Both EL and Ubuntu:

```
wget
https://github.com/FiloSottile/mkcert/releases/download/v1.4.3/mkcert-v1.4.3-linux-amd64
sudo mv mkcert-v1.4.3-linux-amd64 /usr/bin/mkcert
sudo chmod +x /usr/bin/mkcert
```

Once you have mkcert executable, you can run:

```
mkcert -install
The local CA is now installed in the system trust store! ⚡
```

Now, you can verify the CA Root by doing:

```
mkcert -CAROOT
/home/element-demo/.local/share/mkcert
```

Your output may not be exactly the same, but it should be similar. Once we've done this, we need to generate self-signed certificates for our hostnames. The following is an example of how to do it for `element.local`. You will need to do this for all of the aforementioned hostnames, including the `fqdn.tld`.

The run for the element fqdn looks like this:

```
mkcert element.local element 192.168.122.39 127.0.0.1

Created a new certificate valid for the following names
- "element.local"
- "element"
- "192.168.122.39"
- "127.0.0.1"

The certificate is at "./element.local+3.pem" and the key at
"./element.local+3-key.pem"

It will expire on 1 May 2024
```

Once you have self-signed certificates, you need to copy them into the certs directory under the config directory. Certificates in the certs directory must take the form of `fqdn.crt` and `fqdn.key`.

Using our above example, these are the commands we would need to run from the installer directory: (We ran `mkcert` in that directory as well.)

```
mkdir ~/.element-onpremise-config/certs
cp element.local+3.pem  ~/.element-onpremise-config/certs/element.local.crt
cp element.local+3-key.pem  ~/.element-onpremise-config/certs/element.local.key
cp synapse.local+3.pem  ~/.element-onpremise-config/certs/synapse.local.crt
cp synapse.local+3-key.pem  ~/.element-onpremise-config/certs/synapse.local.key
cp dimension.local+3.pem  ~/.element-onpremise-config/certs/dimension.local.crt
cp dimension.local+3-key.pem  ~/.element-onpremise-config/certs/dimension.local.key
cp hookshot.local+3.pem  ~/.element-onpremise-config/certs/hookshot.local.crt
cp hookshot.local+3-key.pem  ~/.element-onpremise-config/certs/hookshot.local.key
cp local+2.pem  ~/.element-onpremise-config/certs/local.crt
cp local+2-key.pem  ~/.element-onpremise-config/certs/local.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` (not resolvable on the internet), 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 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"
```

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'
```

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:

```
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_password`. 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_password` 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 setting up SSO/SAML, please see Setting up SSO/SAML 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 pointing your installation at an existing Jitsi instance, please see Setting up Jitsi With the Installer

Installation

Let’s review! Have you considered:

- Hostnames/DNS
- Operating System
- Users
- Network Specifics
- 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 the proof of concept!

Revision #21
Created 18 April 2022 14:32:03 by Karl Abbott
Updated 17 May 2022 20:05:18 by Karl Abbott