How to Install a Production Environment

Our Element Enterprise Production Installer can handle the installation of Element Enterprise into your production k8s environment.

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

- Hostnames/DNS
- Resource Requirements
- 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!

Hostnames/DNS

You will need hostnames for the following pieces of infrastructure:

- Element Server
- Synapse Server
- Dimension Server
- Hookshot Server

These hostnames must resolve to the appropriate IP addresses. You must have a proper DNS server to serve these records in a production environment.

Resource Requirements

For running running in production, 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

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, we recommend installing `coturn` outside of your k8s environment. `coturn` must open a lot of ports to work and this can be problematic for k8s environments. 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 certificates from an internet recognized authority.
- LetsEncrypt
- Signed certificates from an internal to your company authority.

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

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 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. For your Postgres server, please set the following:

```
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 not using LetsEncrypt, 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'
```

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:

```
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 setting up SSO/SAML, please see Setting up SSO/SAML

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 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
- 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
```

Revision #11
Created 18 April 2022 14:32:04 by Karl Abbott
Updated 17 May 2022 19:08:34 by Karl Abbott