

Host Section

Initial configuration options specific to the installer, including how ESS should be deployed.

Config:



The first section of the ESS installer GUI is the Host section, here you will configure essential details of how ESS will be installed including; deployment type; subscription credentials; PostgreSQL to use; and whether or not your setup is airgapped.

Settings configured via the UI in this section will mainly be saved to your `cluster.yml`. If performing a Kubernetes deployment, you will also be able to config Host Admin settings which will save configuration into both `internal.yml` and `deployment.yml`.

Depending on your environment you will need to select either `Standalone` or `Kubernetes Application`.

`Standalone` will install `microk8s` locally on your machine, and deploy to it so all pods are running locally on the host machine. `Kubernetes Application` will deploy to your Kubernetes infrastructure in a context you will need to have already setup via your kube config.

Deployment (Standalone)

Install



Standalone



Kubernetes Application

Config Example

```
spec:
  connectivity:
    dockerhub:
      password: example
      username: example
  install:
    emsImageStore:
      password: example
```

```
username: example
webhooks:
  caPassphrase: example
[]# Options unique to selecting Standalone
certManager:
  adminEmail: example@Dexample.com
microk8s:
  dnsResolvers:
    - 8.8.8.8
    - 8.8.4.4
postgresInCluster:
  hostPath: /data/postgres
  passwordsSeed: example
```

An example of the `cluster.yml` config generated when selecting Standalone, note that no specific flag is used within the config to specify selecting between Standalone or Kubernetes. If you choose to manually configure ESS bypassing the GUI, ensure only config options specific to how you wish to deploy are provided.

Select your deployment type here, if you've jumped ahead you should first read our [Introduction to Element Server Suite](#) and then see our [Requirements and Recommendations](#) which details the environment specifics needed for each deployment type.

Cert Manager

☒ Setup Cert Manager ☐ Skip Cert Manager

Admin Email *

example@example.com

Edited

The Admin email configured on the issuer

Config Example

```
spec:
  install:
    # certManager: {} # When 'Skip Cert Manager' selected
  certManager:
    adminEmail: example@example.com
```

You should keep this enabled if you will be using Let's Encrypt to verify your domain and generate your certificates, simply provide the username where due to expire certificate notices will be sent.

If you plan to upload your own certificates, or they will be Externally Managed, you should select Skip Cert Manager.

EMS Image Store


Username *

example

Edited

Token *

.....

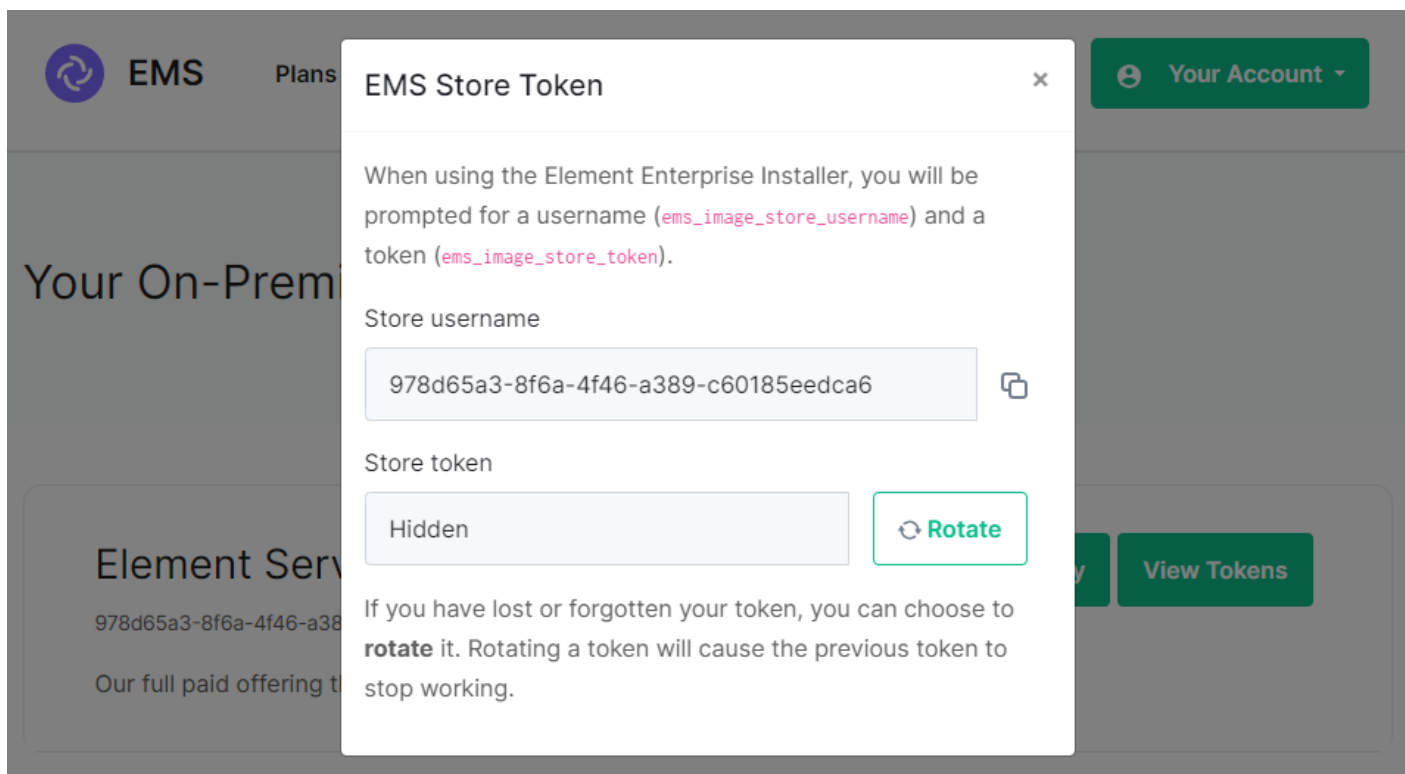


Config Example

```
spec:
  install:
    emsImageStore:
      password: token
      username: test
```

Here you will need to provide your EMS Image Store Username and Token associated with your subscription, which you can find at <https://ems.element.io/on-premise/subscriptions>.

If you forget your token and hit 'Refresh' in the EMS Control Panel, you will need to ensure you redeploy your instance with the new token - otherwise subsequent deployments will fail.



MicroK8s

Persistent Volumes Path

/data/element-deployment

Default

The host path where to store the persistent volumes. They will be hosted as subfolders of this path.

Registry Size *

20Gi

Default

The size of the registry in Gi

Config Example

```
spec:
  install:
    microk8s:
      persistentVolumesPath: /data/element-deployment
      registrySize: 25Gi
```

It is unlikely you should need to adjust these values and it is highly recommended to leave this as their defaults.

If you encounter a requirement to clean up your images cache, see the [Cleaning up images cache](#) section from the [Post-Installation Essentials](#) page.

DNS Resolvers

A DNS Server IP

8.8.8.8

Default

A DNS Server IP

8.8.4.4

Default

Add more DNS Resolvers

Config Example

```
spec:
  install:
    microk8s:
      dnsResolvers:
        - 8.8.8.8
        - 8.8.4.4
```

Defaulting to 8.8.8.8 and 8.8.4.4 , the DNS server IPs set here will be used by all deployed pods. Click [Add more DNS Resolvers](#) to add additional entries as required.

Nginx Extra Configuration

Name

Add to Nginx Extra Configuration

Config Example

```
spec:
  install:
    microk8s:
```

```
# Not present when disabled
nginxExtraConfiguration:
  custom-http-errors: "404"
  server-snippet: >-
    error_page 404 /404.html; location = /404.html { internal; return 200
    "<p>Hello World!</p>"; }
```

As linked via the ESS installer GUI, see the [Ingress-Nginx Controller ConfigMaps](#) documentation for the options that can be configured.

Example

The below example is for demonstration purposes only, you should follow the linked guidance before adding extra configuration.


For example, if you wanted to replace the standard 404 error page, you could do this using both `custom-http-errors` and `server-snippet`. To configure via the installer, simply add the specify `custom-http-errors` as the Name and click `Add to Nginx Extra Configuration`, then provide the required value in the newly created field:

 Custom HTTP Errors

"404"

Edited

Repeat for `server-snippet`:

 Server Snippet

error_page 404 /404.html; location = /404.html { inte

Edited

The above example is used to explain how to configure the Nginx Extra Configuration, and so is for demonstration purposes only, it is not recommended to use this example config. Ideally your web server should manage traffic that would otherwise hit a 404 being served by ESS.

PostgreSQL in Cluster

☒ PostgreSQL in Cluster ☐ External PostgreSQL Server

Host Path *

/data/postgres

Default

The host path where to store the postgres dbs. They will be hosted as subfolders of this path.

Passwords Seed *

.....



The passwords seeds to use.

Config Example

```
spec:
  install:
    microk8s:
      # postgresInCluster: {} # If 'External PostgreSQL Server' selected
    postgresInCluster:
      hostPath: /data/postgres
      passwordsSeed: example
```

Only available in Standalone deployments you can have the installer deploy PostgreSQL for you, this will remove the requirement to configure PostgreSQL connection and authentication credentials in later parts of the installer. It is highly recommended to keep the default settings if you opt for this approach.

If you already have an external PostgreSQL server you wish to use, make sure you have followed the [PostgreSQL Standalone Environment Prerequisites](#) detailed on the [Requirements and Recommendations](#) page. Selecting this option will present an additional `Database` section in the installer process.

Internal Webhooks

Settings for webhooks sent within the Kubernetes cluster

CA Passphrase

YpiNQMMzBjalfVPQqxcxO4e211YFR5

The passphrase used by the internal CA to self-sign certificates

Config Example

```
spec:
  install:
    webhooks:
      caPassphrase: YpiNQMMzBjalfVPQqxcxO4e211YFR5
```

You should not need to change this, a unique CA passphrase will be generated on first run of the installer and is used by the internal CA to self-sign certificates.

Deployment (Kubernetes Application)

Install

☐ Standalone ☒ Kubernetes Application

Config Example

```
spec:
  connectivity:
    dockerhub:
      password: example
      username: example
  install:
    emsImageStore:
      password: example
      username: example
  webhooks:
    caPassphrase: example
  # Options unique to selecting Standalone
  clusterDeployment: true
  kubeContextName: example
  namespaces: {}
  skipElementCrdsSetup: false
  skipOperatorSetup: false
  skipUpdaterSetup: false
```


An example of the `cluster.yml` config generated when selecting Kubernetes, note that no specific flag is used within the config to specify selecting between Standalone or Kubernetes. If you choose to manually configure ESS bypassing the GUI, ensure only config options specific to how you wish to deploy are provided.

Select your deployment type here, if you've jumped ahead you should first read our [Introduction to Element Server Suite](#) and then see our [Requirements and Recommendations](#) which details the environment specifics needed for each deployment type.

Cluster Deployment



Cluster Deployment

Deploy the operator & the updater using Cluster Roles

Config Example

```
spec:
  install:
    clusterDeployment: true
```

Deploy the operator & the updater using Cluster Roles.

Kube Context Name

Kube Context Name *

example

Name of a Kubernetes context already setup in your kube config to install into

Config Example

```
spec:
  install:
    kubeContextName: example
```

The name of the Kubernetes context you have already setup that ESS should be deployed into.

Skip Setup Options

☐ Skip Element CRDs setup

Default

☐ Skip Operator Setup

Default

☐ Skip Updater Setup

Default

Config Example

```
spec:
  install:
    skipElementCrdsSetup: false
    skipOperatorSetup: false
    skipUpdaterSetup: false
```

Selecting these will allow you to skip the setup of the Element CRDs, Operator and Updater as required.

Namespaces

☒ Create Namespaces

Create the namespaces or use an existing one

Default

Element Deployment

element-onprem

Default

The namespace where to deploy the element applications

Operator

operator-onprem

Default

The namespace where to deploy the operator controller manager

Updater

updater-onprem

Default

The namespace where to deploy the updater controller manager

Config Example

```
spec:
  install:
    # namespaces: {} # When left as default namespaces
    # namespaces: # When `Create Namespaces` is disabled
    # createNamespaces: false
    namespaces: # When custom namespaces are provided
    elementDeployment: element-example # Omit any that should remain as default
    operator: operator-example
    updater: updater-example
```

Allows you to specify the namespaces you wish to deploy into, with the additional option to create them if they don't exist.

Namespace-scoped Deployments

Namespace-scoped deployments in Kubernetes offer a way to organize and manage resources within specific namespaces rather than globally across the entire cluster.

Preparing the Cluster

Installing the Helm Chart Repositories

The first step is to start on a machine with helm v3 installed and configured with your kubernetes cluster and pull down the two charts that you will need.

First, let's add the element-updater repository to helm:

```
helm repo add element-updater https://registry.element.io/helm/element-updater --username
ems_image_store_username --password 'ems_image_store_token'
```

Replace `ems_image_store_username` and `ems_image_store_token` with the values provided to you by Element.

Secondly, let's add the element-operator repository to helm:

```
helm repo add element-operator https://registry.element.io/helm/element-operator --username
ems_image_store_username --password 'ems_image_store_token'
```

Replace `ems_image_store_username` and `ems_image_store_token` with the values provided to you by Element.

Now that we have the repositories configured, we can verify this by:

```
helm repo list
```

and should see the following in that output:

NAME	URL
element-operator	https://registry.element.io/helm/element-operator
element-updater	https://registry.element.io/helm/element-updater

Deploy the CRDs

Write the following `values.yaml` file:

```
clusterDeployment: true
deployCrds: true
deployCrdRoles: true
deployManager: false
```

To install the CRDs with the helm charts, simply run:

```
helm install element-updater element-updater/element-updater -f values.yaml
helm install element-operator element-operator/element-operator -f values.yaml
```

Now at this point, you should have the following two CRDs available:

```
[user@helm ~]$ kubectl get crds | grep element.io
elementwebs.matrix.element.io          2023-10-11T13:23:14Z
wellknowndelegations.matrix.element.io 2023-10-11T13:23:14Z
elementcalls.matrix.element.io          2023-10-11T13:23:14Z
hydrogens.matrix.element.io             2023-10-11T13:23:14Z
mautrixtelegrams.matrix.element.io      2023-10-11T13:23:14Z
sydents.matrix.element.io               2023-10-11T13:23:14Z
synapseusers.matrix.element.io           2023-10-11T13:23:14Z
bifrosts.matrix.element.io              2023-10-11T13:23:14Z
lowbandwidths.matrix.element.io          2023-10-11T13:23:14Z
synapsemoduleconfigs.matrix.element.io   2023-10-11T13:23:14Z
matrixauthenticationservices.matrix.element.io 2023-10-11T13:23:14Z
ircbridges.matrix.element.io             2023-10-11T13:23:14Z
slidingsyncs.matrix.element.io           2023-10-11T13:23:14Z
```

securebordergateways.matrix.element.io	2023-10-11T13:23:14Z
hookshots.matrix.element.io	2023-10-11T13:23:14Z
matrixcontentscanners.matrix.element.io	2023-10-11T13:23:14Z
sygnals.matrix.element.io	2023-10-11T13:23:14Z
sipbridges.matrix.element.io	2023-10-11T13:23:14Z
livekits.matrix.element.io	2023-10-11T13:23:14Z
integrators.matrix.element.io	2023-10-11T13:23:14Z
jitsis.matrix.element.io	2023-10-11T13:23:14Z
mautrixwhatsapps.matrix.element.io	2023-11-15T09:03:48Z
synapseadminuis.matrix.element.io	2023-10-11T13:23:14Z
synapses.matrix.element.io	2023-10-11T13:23:14Z
groupsyncs.matrix.element.io	2023-10-11T13:23:14Z
pipes.matrix.element.io	2023-10-11T13:23:14Z
elementdeployments.matrix.element.io	2023-10-11T13:34:25Z
chatterboxes.matrix.element.io	2023-11-21T15:55:59Z

Namespace-scoped role

In the namespace where the ESS deployment will happen, to give a user permissions to deploy ESS, please create the following role and roles bindings:

- User role:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: ess-additional
rules:
- apiGroups:
  - apiextensions.k8s.io
  resources:
  - customresourcedefinitions
  verbs:
  - list
  - watch
  - get
- apiGroups:
  - project.openshift.io
  resources:
  - projects
```

verbs:

- get
- list
- watch

- User roles bindings:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: ess-additional
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: ess-additional
subjects:
# role subjects which maps to the user or its groups
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: ess
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: edit
subjects:
# role subjects which maps to the user or its groups
```

Once your cluster is prepared, you can setup your namespace-scoped deployment by configuring these settings:

- **Skip Operator Setup.**
Unchecked
- **Skip Updater Setup.**
Unchecked
- **Skip Element CRDs Setup.**
Checked
- **Cluster Deployment.**
Unchecked
- **Kube Context Name.**
Set to
- **Namespaces.**

- **Create Namespaces.**
Unchecked
- **Operator.**
Set to `namespace_to_deploy_ess`
- **Updater.**
Set to same as Operator, `namespace_to_deploy_ess`
- **Element Deployment.**
Set to same as Operator, `namespace_to_deploy_ess`

Internal Webhooks

Settings for webhooks sent within the Kubernetes cluster

CA Passphrase

YpiNQMMzBjalfVPQqxcxO4e211YFR5

The passphrase used by the internal CA to self-sign certificates

Config Example

```
spec:
  install:
    webhooks:
      caPassphrase: YpiNQMMzBjalfVPQqxcxO4e211YFR5
```

Connectivity

☒ Connected ☐ Airgapped

Config Example

```
spec:
  connectivity:
```

Connected

Dockerhub

Optionally reduce rate limiting by providing your dockerhub credentials

Username & Password



Config Example

```
spec:
  connectivity:
    # dockerhub: {} # When Username & Password is disabled per default
  dockerhub:
    password: password
    username: test
```

Connected means the installer will use the previously provided **EMS Image Store** credentials to pull the required pod images as part of deployment, optionally, you can specify DockerHub credentials to reduce potential rate limiting.

Airgapped

- ☒ Upload Images with the Installer
- ☐ Target an Existing Local Image Registry

The installer will automatically upload images to the provided registry.

Local Registry * Default

The local registry url to find airgapped images

Source Directory *

The path to the airgapped directory to upload the images

Upload Credentials

Upload credentials, if the registry is protected behind auth.

- ☒ Authenticate against the Registry to Upload
- ☐ Upload without Authentication

Password *

Upload password

Username *

Upload username

- ☐ Upload Images with the Installer
- ☒ Target an Existing Local Image Registry

If using this mode, images must already exist in the local image registry.

Local Registry * Default

The local registry url to find airgapped images

Source Directory *

The path to the airgapped directory to upload the images

Config Example

```
spec:
  connectivity:
    airgapped:
      localRegistry: localhost:32000
      sourceDirectory: /home/ubuntu/airgapped/
  # uploadCredentials not present if `Target an Existing Local Image Registry` selected
  # uploadCredentials: {} # If 'Upload without Authentication'
  uploadCredentials:
    password: example
    username: example
```

An airgapped 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.

Selecting Airgapped means the installer will rely on images stored in a registry local to your environment, by default the installer will host this registry uploading images found within the specified `Source Directory`, however you can alternatively specify one already present in your environment instead.

Getting setup within an Airgapped environment

Alongside each Installer binary available for download, for those customers with airgapped permissions, is an equivalent airgapped package `element-enterprise-installer-airgapped-<version>-gui.tar.gz`. Download and copy this archive to the machine running the installer, then use `tar -xzf element-enterprise-installer-airgapped-<version>-gui.tar.gz` to extract out its contents, you should see a folder `airgapped` with the following directories within:

- `pip`
- `galaxy`
- `snaps`
- `containerd`
- `images`

Copy the full path of the root `airgapped` folder, for instance, `/home/ubuntu/airgapped` and paste that into the `Source Directory` field. Should you ever update the ESS installer binary, you will need to ensure you delete and replace this `airgapped` folder, with its updated equivalent.

Your airgapped machine will still require access to airgapped linux repositories depending on your OS. If using Red Hat Enterprise Linux, you will also need access to the [EPEL repository](#) in your airgapped environment.

Host Admin

Host Admin Domain

Domain name to use for this UI when running directly on the host (optional)

Trusted HTTPS

The certificate used for accessing the host admin panel (this UI when running directly on the host)

- ☒ Self Signed ☐ Automatic Let's Encrypt ☐ Certificate File
- ☐ Existing TLS Certificates in the Cluster ☐ Externally Managed

Use an automatically generated self-signed certificate.

Config Example

- `internal.yml`

spec:

fqdn: admin.example.com

tls:

When selecting `Self Signed`

mode: self-signed

When selecting `Automatic Let's Encrypt`

mode: automatic

automatic:

adminEmail: example@example.com

When selecting `Certificate File`

mode: certfile

certificate:

certFile: "example" # Base64 encoded string from certificate

privateKey: "example" # Base64 encoded string from certificate key

When selecting `Existing TLS Certificates in the Cluster`

mode: existing

secretName: example

When selecting `Externally Managed`

mode: external

- deployment.yml

spec:

components:

synapseAdmin:

config:

hostOrigin: >-

https://admin.example.com,https://admin.example.com:8443

The Host Admin section allows you to configure the domain name and certificates to use when serving the ESS installer GUI, when running directly on the host - changes here will take affect the next time you run the installer.

Revision #113

Created 1 February 2024 12:28:40 by Kieran Mitchell Lane

Updated 28 May 2025 11:25:32 by Kieran Mitchell Lane