How to Setup ESS Community on Hetzner Cloud

This guide walks you through setting up Element Server Suite (ESS) Community on Hetzner Cloud. By the end of this guide, you will be able to use Element to its full extent on the platform of your choice.

What is Element?

Element is an **open-source**, **decentralized**, **end-to-end encrypted**, **cross-platform messaging** application built on the Matrix protocol. It is available natively for Windows, macOS, Linux, Android, and iOS, and also offers a web client for any browser.

Element combines many of the features we love from other messaging apps: the privacy and anonymity of Signal; servers (*called spaces*), chat and voice channels, including video and screen sharing, similar to Discord; and polls and live-location sharing like WhatsApp.

Target Audience

This guide is aimed at beginners and non-tech-savvy users who want a robust and **ready-to-use** solution for a small community quickly. If needed, adjusting the configuration later is straightforward and non-destructive.

Investment

You will need approximately **$15 per month**, an account at Hetzner, and a domain purchased from a registrar of your choice (*e.g., Cloudflare*). For beginners, completing this guide is expected to take between 1–3 hours.

Disclaimer

This guide was written in early 2026. Since I do not regularly set up Element, parts of it may become outdated over time. It is tailored to macOS, so you may need to slightly adjust command-line interface (CLI) operations if you are using Linux, and more significantly if you are using Windows.

The official, up-to-date documentation can be found here: ess-helm, hetzner-k3s.

Need help?

Don't hesitate to open an issue on GitHub, and I will gladly try to help you.

Want to help?

Open an issue on GitHub and let me know where you struggled, or fork this repository, make improvements directly, and submit a pull request.

Prerequisites

Before following the installation instructions, you will need to complete the following:

1. Have a macOS device with brew installed.
2. Create an account on Hetzner and create a new project.
3. Generate and save a *Read & Write* API token by following this guide.
4. Buy a domain at a domain name registrar of your choice (*this guide uses Cloudflare*).
5. Follow this guide to create an SSH key if you don't already have one.

Installation

First, you are going to create the configuration files needed, then you'll be setting up Kubernetes, and finally, you are going to install the Element Server Suite.

1. Configuration

Chose a **permanent** folder where you will store every configuration. For the entire guide, you will always work in this folder.

1. Download cluster.yaml. Replace `<your token>` with the Hetzner API token you've generated before.
   If you wish to further configure k3s settings (e.g., adjust server location), follow this guide for a complete example with all options.
2. Download hostnames.yaml. Replace each `your.tld` with your domain name (*e.g., example.com*).
3. Download tls.yaml.
4. Download synapse.yaml.
   Find all available configuration options here, if you wish to customize Synapse.

2. Set up K3S

For this step, you'll have to open your terminal at the same directory where you have stored your configurations. Use this terminal session for the entirety of the remaining guide.

1. Install `hetzner-k3s`, `kubectl` and `helm`:

   ```
   brew install vitobotta/tap/hetzner_k3s
   brew install kubectl
   brew install helm
   ```

2. Create your Kubernetes cluster on Hetzner. This will take about 2–3 minutes.

   ```
   hetzner-k3s create --config cluster.yaml
   ```

   If you did not adjust `masters_pool` or `worker_node_pools` in `cluster.yaml`, this action will result in created services which are billed hourely when used and sum up to about 15$ per month.

3. Once finished, set up an environment variable:

```
export KUBECONFIG=./kubeconfig
```

Be advised that this environment variable will have to be set every time you open a new terminal session.

4. Verify that your Kubernetes cluster has successfully been created and is running:

```
kubectl get nodes
```

Make sure it says `Ready` under the *status* value.

3. Set up DNS

Without closing your terminal, open your browser of choice.

1. Go to Hetzner Console, then open your project, and finally chose *Servers* in the left-hand bar. Copy the value under *Public IP* (*e.g., 42.161.13.12*).
2. Create six DNS records of type *A*. Follow this guide if you have bought your domain at Cloudflare. No matter of your domain name registrar, the DNS record table should ultimately look akin to this:

Type	Name	IP	Proxy status
A	`<domain>`	`<Public IP>`	DNS only
A	account. `<domain>`	`<Public IP>`	DNS only
A	admin. `<domain>`	`<Public IP>`	DNS only
A	chat. `<domain>`	`<Public IP>`	DNS only
A	matrix. `<domain>`	`<Public IP>`	DNS only
A	mrtc. `<domain>`	`<Public IP>`	DNS only

Make sure to replace `<Public IP>` *with the value from Hetzner Console and* `<domain>` *with your domain name (e.g., example.com). Also make sure that you set the proxy status for each record to* `DNS only` *to avoid using a reverse proxy.*

4. Configure Certificates

Return to your terminal. Periodically run the following command until it returns the public IP address of your Kubernetes cluster (*replace* `<domain>` *with your domain name*):

```
dig +short A chat.<domain>
```

It may take some time for the DNS to propagate your records. We hereby make sure that your Kubernetes cluster can be reached via your domain before continuing.

1. Install cert-manager using *Helm*:

```
helm install \
cert-manager oci://quay.io/jetstack/charts/cert-manager \
--version v1.19.3 \
--namespace cert-manager \
--create-namespace \
--set crds.enabled=true
```

Find the latest version of cert-manager here.

2. Verify that cert-manager is running:

```
kubectl get pods -n cert-manager
```

Make sure that all pods are in the `Running` state.

3. Create the Let's Encrypt issuer:

```
kubectl apply -f - <<EOF
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: letsencrypt-prod
spec:
  acme:
    server: https://acme-v02.api.letsencrypt.org/directory
    privateKeySecretRef:
      name: letsencrypt-prod-private-key
    solvers:
      - http01:
          ingress:
            class: traefik
EOF
```

4. Verify that the issuer has been created:

```
kubectl get clusterissuer letsencrypt-prod
```

Make sure that the issuer is in the `Ready` state.

5. Set up Element Server Suite (ESS)

1. Create the `ess` namespace:

```
kubectl create namespace ess
```

2. Install ESS using *Helm*:

```
helm upgrade --install \
--namespace "ess" \
ess \
oci://ghcr.io/element-hq/ess-helm/matrix-stack \
-f hostnames.yaml \
-f tls.yaml \
-f synapse.yaml \
--wait
```

Add optional additional configuration files by appending `-f <file>` to the command. The same command is also used to update ESS after changes to configuration files.

3. Create your first user:

```
kubectl exec -n ess -it \
deploy/ess-matrix-authentication-service \
-- mas-cli manage register-user
```

Make sure to select `Create user` after entering your desired username and password, else the user will not be created. It might also be beneficial to make this user an admin.

4. Either download the desktop or mobile app or navigate to `https://app.<domain>`. Enjoy Element!