# On-Premises Documentation | Version: 2021-05-01

## Helm & Nagel GmbH

Konfuzio by Helm & Nagel GmbH
Rosenweg 5
35614 Aßlar
info@helm-nagel.com
www.helm-nagel.com

Geschäftsführer:
Christopher Helm, Florian Zyprian
Steuernummer:
USt.-Id Nr.: DE 306902936
Amtsgericht Wetzlar, HRB 7706

# Required tools

Before deploying Konfuzio to your Kubernetes cluster, there are some tools you must have installed locally.

## kubectl

kubectl is the tool that talks to the Kubernetes API. kubectl 1.15 or higher is required and it needs to be compatible with your cluster (**+/- 1 minor release from your cluster**).

**> Install kubectl locally by following the Kubernetes documentation.**

The server version of kubectl cannot be obtained until we connect to a cluster. Proceed with setting up Helm.

## Helm

Helm is the package manager for Kubernetes. Konfuzio is tested and supported with Helm v3.

### Getting Helm

You can get Helm from the project's **releases page**, or follow other options under the official documentation of **installing Helm**.

Connect to a local Minikube cluster

For test purposes you can use `minikube` as your local cluster. If `kubectl cluster-info` is not showing `minikube` as the current cluster, use `kubectl config set-cluster minikube` to set the active cluster. For klusters in production please visit the **Kubernetes Documentation**.

### Initializing Helm

If Helm v3 is being used, there no longer is an `init` sub command and the command is ready to be used once it is installed. Otherwise please upgrade Helm.

## Next steps

Once kubectl and Helm are configured, you can continue to configuring your Kubernetes cluster.

# Deployment

Before running `helm install`, you need to make some decisions about how you will run Konfuzio. Options can be specified using Helm's `--set option.name=value` command line option. A complete list of command line options can be found **here**. This guide will cover required values and common options.

## Selecting configuration options

In each section collect the options that will be combined to use with `helm install`.

### Secrets

There are some secrets that need to be created (e.g. SSH keys). By default they will be generated automatically.

### Networking and DNS

By default, Konfuzio relies on Kubernetes `Service` objects of `type: LoadBalancer` to expose Konfuzio services using name-based virtual servers configured with `Ingress` objects. You'll need to specify a domain which will contain records to resolve the domain to the appropriate IP.

```
--set global.hosts.domain=example.com
```

IPs

If you plan to use an automatic DNS registration service, you won't need any additional configuration for Konfuzio, but you will need to deploy it to your cluster.

If you plan to manually configure your DNS records they should all point to a static IP. For example if you choose `example.com` and you have a static IP of `10.10.10.10`, then `konfuzio.example.com`, `registry.example.com` and `minio.example.com` (if using MinIO) should all resolve to `10.10.10.10`.

*Include these options in your Helm install command:*

```
--set global.hosts.externalIP=10.10.10.10
```

### Persistence

By default the setup will create Volume Claims with the expectation that a dynamic provisioner will create the underlying Persistent Volumes. If you would like to customize the storageClass or manually create and assign volumes, please review the **storage documentation**.

**Important**: After initial installation, making changes to your storage settings requires manually editing Kubernetes objects, so it's best to plan ahead before installing your production instance of Konfuzio to avoid extra storage migration work.

## TLS certificates

You should be running Konfuzio using https which requires TLS certificates. By default the setup will install and configure **cert-manager** to obtain free TLS certificates. If you have your own wildcard certificate, you already have cert-manager installed, or you have some other way of obtaining TLS certificates. For the default configuration, you must specify an email address to register your TLS certificates.

*Include these options in your Helm install command:*

```
--set certmanager-issuer.email=me@example.com
```

## PostgreSQL

By default this Konfuzio provides an in-cluster PostgreSQL database, for trial purposes only.

**NOTE: This configuration is not recommended for use in production.**

- A single, non-resilient Deployment is used

You can read more about setting up your production-ready database in the PostgreSQL documentation. As soon you have an external PostgreSQL database ready, Konfuzio can be configured to use it as shown below.

*Include these options in your Helm install command:*

```
--set postgresql.install=false
--set global.psql.host=production.postgress.hostname.local
--set global.psql.password.secret=kubernetes_secret_name
--set global.psql.password.key=key_that_contains_postgres_password
```

## Redis

All the Redis configuration settings are configured automatically.

## Persistent Volume

Konfuzio relies on object storage for highly-available persistent data in Kubernetes. By default Konfuzio uses a **persistent volume** within the cluster.

### Outgoing email

By default outgoing email is disabled. To enable it, provide details for your SMTP server using the `global.smtp` and `global.email` settings. You can find details for these settings in the command line options.

```
--set global.smtp.address=smtp.example.com:587
--set global.smtp.AuthUser=username-here
--set global.smtp.AuthPass=password-here
```

### CPU, GPU and RAM Resource Requirements

The resource requests, and number of replicas for the Konfuzio components in this setup are set by default to be adequate for a small production deployment. This is intended to fit in a cluster with at least 16 vCPU with AVX2 support enabled,  32 GB of RAM and one Nividia GPU which supports at least CUDA 10.1 and CUDNN 7.0. If you are trying to deploy a non-production instance, you can reduce the defaults in order to fit into a smaller cluster. Konfuzio can work without GPU, however runtime for extraction and training of medium to large dataset will be significantly slower.

## Deploy using Helm

Once you have all of your configuration options collected, we can get any dependencies and run Helm. In this example, we've named our Helm release `konfuzio`.

```
helm repo add konfuzio
helm repo update
helm upgrade --install konfuzio konfuzio/server \
  --timeout 600s \
  --set global.hosts.domain=example.com \
  --set global.hosts.externalIP=10.10.10.10 \
  --set certmanager-issuer.email=me@example.com
```

You can also use `--version <installation version>` option if you would like to install a specific version of Konfuzio. This will output the list of resources installed once the deployment finishes which may take 5-10 minutes.

## Monitoring the Deployment

The status of the deployment can be checked by running `helm status konfuzio` which can also be done while the deployment is taking place if you run the command in another terminal.

## Initial login

You can access the Konfuzio instance by visiting the domain specified during installation. If you manually created the secret for initial root password, you can use that

to sign in as `root` user. If not, Konfuzio would've automatically created a random password for `root` user. This can be extracted by the following command (replace `<name>` by name of the release - which is `konfuzio` if you used the command above).

```
kubectl get secret <name>-konfuzio-initial-root-password
-ojsonpath='{.data.password}' | base64 --decode ; echo
```

# Upgrade

Before upgrading your Konfuzio installation, you need to check the **changelog** corresponding to the specific release you want to upgrade to and look for any that might pertain to the new version.

We also recommend that you take a backup first.

## Steps

Upgrade Konfuzio following our standard procedure, with the following additions of:

1. Check the change log for the specific version you would like to upgrade to
2. Ensure that you have created a **PostgreSQL backup** in the previous step. Without a backup, Konfuzio data might be lost if the upgrade fails.
3. Go through deployment section step by step
4. Extract your previous `--set` arguments with (see Action 1)
5. Decide on all the values you need to set
6. Perform the upgrade, with all `--set` arguments extracted (see Action 2)
7. We will perform the migrations for the Database for PostgreSQL automatically.

*Action 1*

```
helm get values konfuzio > konfuzio.yaml
```

*Action 2*

```
helm upgrade kofuzio \
 --version <new version> \
 -f konfuzio.yaml \
 --set konfuzio.migrations.enabled=true \
 --set ...
```

# Alternative deployment options

## Single VM setup

Konfuzio can be configured to run on a single virtual machine, without relying on Kubernetes. In this scenario, all necessary containers are started manually or with a container orchestration tool of your choice.

We recommend a virtual machine with a minimum of 12 vCPU (incl. AVX2 support) and 64GB of RAM and an installed Docker runtime. A Nvidia GPU is recommended but not required. In this setup Konfuzio is running in the context of the Docker executor, therefore there are no strict requirements for the VMs operating systems. However, we recommend a Linux VM with Debian, Ubuntu, CentOS, or Redhat Linux.

The Konfuzio docker image can be downloaded via “docker pull”. We will provide you with the credentials. This action requires an internet connection.

```
> docker login registry.gitlab.com
> docker pull registry.gitlab.com/konfuzio/text-annotation/master:latest
```

The internet connection can be turned off once the setup is complete. In case there is no internet connection available during setup, the container must be transferred with an alternative method as a file to the virtual machine.

Once you have configured your deployment configuration file, Konfuzio can be started by passing these settings as environment variables.

```
> docker run --env-file <local_path_to your_deployment_config>
registry.gitlab.com/konfuzio/text-annotation/master:latest
```

We will support you in defining reasonable deployment settings for your scenario.

## Custom AI model training via CI pipelines

Konfuzio uses CI pipelines to allow users to run custom AI model code securely. In case the Kubernetes deployment option is not used, we recommend a dedicated virtual machine to run these pipelines. The selected CI application needs to support Docker and webhooks. The CI application needs network access to the Konfuzio installation.