

# InkVisitor installation on the server

This chapter is intended for your IT support. It describes how to deploy the InkVisitor application on a server so that you can start using it.

- [How to deploy your own instance of InkVisitor](#)

# How to deploy your own instance of InkVisitor

## Deploy with Docker

You can use docker to deploy the InkVisitor application

1. Install [docker](#).
2. Install [docker-compose tool](#).
3. Clone | Fork | Download the Inkvisitor [repository](#).
4. Prepare `.env` files for servers listed under `env_file` sections. Check the server's [README.md](#) and [example.env](#) files for more information.
5. To prepare the necessary configuration files for the client application, you should identify the appropriate environment variables (ENV) under the `build -> args` section and then use them to create the `.env` files. You can see the server's [README.md](#) and [example.env](#) files to ensure you have included all the necessary configuration information.
6. Run the database - either as a service or containerized using `docker-compose up -d database`
7. Build app image (will also be done in next step if not available) `docker-compose build inkvisitor` (or `inkvisitor-<env>`).
8. Run the containerized application with the command `docker-compose up inkvisitor` (or `inkvisitor-<env>`).

## Deploy by packages

The InkVisitor codebase consists of three interconnected packages (parts) - the client application, the server, and the database. You can deploy those packages individually if you do not want to use Docker. In each step, make sure to have the appropriate `.env.<env>` file accessible - see the Readme.md file in the package for more information.

### 1. Client application

The client application runs on static files - html/css/js + additional assets. These files need to be moved to your HTTP server by:

1. Build the frontend app by `npm run build-<env>` to create/update the `dist` folder
2. Copy contents of `dist` folder to the directory used by your HTTP server.

### 2. Server

The server is also built in Javascript, using mainly the Node + Express libraries. You need to first build the application, move the build to your server and run it from there.

1. Run `yarn run build` to transpile the code.
2. Move the `dist` folder to your server that supports the Node.js environment.
3. Do `ENV_FILE=<env> yarn run start` to run the built application with a loaded `.env.<env>` file.

### 3. Database

Follow tutorials on the [official page](#) to install RethinkDB on your machine. Then, use the import script to create the database structure and (optional) import some testing data by running ``npm run import`` and following the information in the prompt.

## Firewall

Make sure the ports required by each application are not blocked. Required ports are listed in [docker-compose.yml](#). Examples:

1. [ufw](#): `ufw allow <port>`
2. [firewalld](#): `firewall-cmd --zone=public --permanent --add-port=<port>/tcp`

Setup for additional system specific features (reverse proxies etc) are beyond the scope of this readme.