Server

The server is hosted in the Lys data center at EPFL. Due to legal limitation on EPFL's side, it is fed a direct connection to the ISP, and thus is not on the school's network. Accesses to the server room can be requested through the AGEP's IT manager which is responsible for the managment of the rack.

It can be accessed by ssh through the user `clic-admin`, below is a corresponding `.ssh/config` entry. Key management is done through the `init.yaml` playbook on the infra repo.

```
Host clic
  HostName clic.epfl.ch
  User clic-admin
```

Connectivity

The connection is provided by Saitis, along with an IPv4 (`62.220.153.185`) and an IPv6 subnet (`2001:788:f185::/48`) exposed onto `2001:788:153:a:185::2/80` (**important**: the IPv6 subnet is not yet setup on the server). It uses the port 35 of the Saitis switch. The current netplan config, `/etc/netplan/static.yaml`, is shown below and can be deployed using `netplan generate && netplan apply`.

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp7s4:
      addresses:
        - 62.220.153.185/24
      nameservers:
        addresses:
          - 8.8.8.8
      routes:
        - to: default
          via: 62.220.153.1
          metric: 200
```

Physical access

The server is stored in the Lys datacenter, rack 19, slots 9-11. It can be accessed at anytime, although the security staff sometimes locks the room after 9PM. Obviously, make sure that everything is in place when leaving the room: both front and back panel of the rack are closed, screens is put back on the right of the entrance (there are marks on the ground), etc.

The datacenter provides screens and keyboards, so no need to bring yours. Note that the server's USB ports are quite capricious, you should use the ones directly below the ethernet port to avoid most issues. The server can be power buttons (red: start, black: reset) are within the case and can be reached from the back using a pen.

Infrastructure

Infrastructure configuration is on the Clic Infra repository. Each service is hosted as a Docker stack. This makes it easy to add new services without worrying about dependencies, architecture, etc.

Updating the server when the configuration has been modified is done using the Ansible tool.

Add/Remove/Modify a service

Modifications to the infrastructure must first be made on the infra repository, then applied to the server using Ansible (see below).

Each service is configured via a `docker-compose.yaml` file (doc), in a role unique to the service. Services generally never communicate with each other, except via their exposed APIs (e.g. the website accesses Directus via its public url, not via a dedicated virtual network). If a service requires a database, it is instantiated in the same stack as the service and is exclusively dedicated to it.

> **Warning**: stacks must be modifiable with a simple `docker stack deploy`; don't use `config`, as these are immutable. The stack would have to be removed and then redeployed, which is a pain. Instead, configure services via their environment variables.

If the service needs to be accessible via the Internet, it must expose one or more ports. As of now, all ports of the machine are accessible, but this should be changed for security reasons.

Once everything is setup and pushed on the `clic-infra` repository, use `ansible-pull -K -U https://github.com/clicepfl/clic-infra.git -e @/var/secrets.yaml deploy.yaml -e SERVICES=<new-`

`service>,webhook -e WH_BUILD=false` on the server. The webhook needs to be reconfigured in order to allow redeployement requests for that service.

Secret management

Services may require private (or server-specific) information, such as an admin password, or a mailbox password. These values are stored in a `secrets.yaml` file (located at `/var/secrets.yaml`), and loaded by Ansible during deployment. They must be referenced in the service role argument list (`roles/<service>/meta/argument_specs.yaml`) passed as environment variables to the stack deployment task (`roles/<service>/tasks/main.yaml`). They can then be retrieved from each stack's configuration using the syntax `${MY_SECRET}`.

Deployment

Deployment is carried out using the playbook `deploy.yaml`, which deploys the stacks referenced in the `SERVICES` variable (using a comma separated list), or all if the variable is empty/undefined. Although ansible is designed to run playbooks on other machines, this one must be ran from the server itself to be able to properly access the secrets file. The `WH_BUILD` variable controls whether the webhook needs to be rebuild, if set to false the hook will only be reconfigured.

Example:

```
# Deploy nextcloud
ansible-pull -K -U https://github.com/clicepfl/clic-infra.git -e @/var/secrets.yaml
deploy.yaml -e SERVICES=nextcloud

# Deploy the website and directus
ansible-pull -K -U https://github.com/clicepfl/clic-infra.git -e @/var/secrets.yaml
deploy.yaml -e SERVICES=website,directus

# Build and configure the webhook
ansible-pull -K -U https://github.com/clicepfl/clic-infra.git -e @/var/secrets.yaml
deploy.yaml -e SERVICES=webhook

# Configure the webhook without rebuilding it
ansible-pull -K -U https://github.com/clicepfl/clic-infra.git -e @/var/secrets.yaml
deploy.yaml -e SERVICES=webhook -e WH_BUILD=false
```

The `-K` arguments forces the command to ask for the `BECOME` password, i.e. the password for `sudo`. This is required as some services like Caddy or the webhook requires root access to be redeployed.

Backups

Backups are managed by BorgBackup, enabling encrypted, incremental backups. The Borgmatic tool, configured by Playbook `deploy.yaml`, periodically generates and uploads backups. Its configuration (also in the Playbook) must be updated whenever a service is added/removed/modified.

To manually create a backup, you can run `sudo borgmatic` (ideally in a detached screen `screen sudo borgmatic`, since it takes several hours to complete). You can inspect the backup repository using the `borg` CLI tool.

Installation

In the event of a complete server reinstallation, here are the steps to follow:

1. Use the init playbook: create an inventory on your computer (see the docs), then `ansible-playbook -i <inventory> <path/to/init.yaml>`.
2. SSH into the server.
3. Set up the `/var/secrets.yaml` file (see the example in the infra repo). Move the database dumps to the directories set in this file. If you do not have access to the Directus tokens, use a stub.
4. Generate an SSH key for the default user an one for `root` and save both in the deployment keys of the infra repository.
5. Launch the stacks with Playbook `deploy.yaml`: `ansible-pull -U https://github.com/clicepfl/clic-infra.git -e @/var/secrets.yaml deploy.yaml`.
6. Check the databases content, using `docker exec -it <container-name> bash` then either `psql -U <user> <database>` or `mysql --user <user> -p`. The dump is present in `/docker-entrypoint-initdb.d/<smth>.sql` and should already be applied; if its not the case it can be piped into the database connection command shown previously: `psql -U <user> <database> < /docker-entrypoint-initdb.d/init.sql`.
7. Stop all the services (`docker stack rm <stack list>`, using `docker stack ls` to list the stacks).
8. Copy/Move the file content in the volumes (stored under `/var/lib/docker/volumes/<volume-name>/_data`), after having remove the content placed by the first run of each service. Skip database volumes (already initialized).
9. If needed, regenerate token for Directus and set them in the secrets file.
10. Restart all the services using the playbook `deploy.yaml`.

Everything should be set now !

Webhook

The webhook is a systemd service running on the server baremetal (i.e. not in a container). It allows GitHub to redeploy some services, for example when a new version is available. It is exposed on the port 4000.

For more detail, see the dedicated README in the infra repository.

Configuring the webhook for a Github repository

1. On GitHub, open the settings of the repository, then go to "Webhooks" and "Add webhook".
2. Set the following values:
   - Url: `https://clic.epfl.ch:4000/website`
   - Content type: `application/json`
   - Secret: Found in `/var/secrets`, under `services.webhook.github.secret`
   - SSL Verification: Enabled
   - Events: Depends on when you want the redeployment to be done, but usually "Packages" if there is a docker build in the CI (in that case, don't forget to uncheck "Pushes").
3. Save the webhook

From now on, every time a new version is available, the server will automatically redeploy the service.

Help I lost access to the server :'(

In the unfortunate case where the server is unreachable, it can be physically accessed at the Lys.

See the section above.

Help everything broke down.

There is a sheet in the admin shelf in the office with the information to access the backup repository, including the URL, encryption passphrase and credentials for the repository's web interface. Obviously **this sheet should always be present as it is the only way to recover the server's data in case of a critical hardware failure**.

Revision #26
Created 12 November 2023 14:44:43 by Ludovic Mermod
Updated 27 March 2026 13:19:25 by IT CLIC