Getting started with Link-Live

Hardware requirements

Hardware	Minimum	Recommended	Detail
Disk space	150GB	300GB	150GB should allow for thousands of uploaded results, files, generated PDFs, etc. Monitor and adjust accordingly.
Memory	12GB	24GB	Insufficient available memory will result in adverse performance effects, and slow or failed PDF report generation. Monitor and adjust accordingly
Processor	4 Cores @ 1.4GHz	8 Cores @ 2.0GHz or faster	Insufficient processing power will result in adverse performance effects, and slow or failed PDF report generation. Monitor and adjust accordingly

Software requirements

We officially support running Docker via Docker Compose on top of Linux/Ubuntu with the following versions:

- Ubuntu 20.04
- Docker, client version 20.10.2 / server version 20.10.2 Docker
- Compose 1.25.0

First steps

- The steps to deploy Link-Live Private through the Linux terminal are as follows. For this example, link_live_private.tar.gz will be used as the file provided. However, the name of the tar.gz file will vary for each customer.
    - Extract the tar.gz file provided.
    - In the terminal, run the command tar -xvzf link_live_private.tar.gz
    - Locate the start script
    - After completing the first step there will be a new directory with the same name as the original provided file name
    - Move into this directory by running the command cd link_live_private
    - Run the start command
    - Run the command sudo ./start.sh
    - From here the user will step through a series of or prompts that detail out how the user is to answer each prompt
- Ensure the docker and docker-compose are installed
- There should be the following files and directories in extracted Link-Live archive that includes this readme:

```
nginx/
docker-images/
version.txt
docker-compose.yml
readme.md
...
```

- The archive files in the `docker-images` directory are Docker images. All of these images need to be loaded via `docker load -i`
    - Reference: https://docs.docker.com/engine/reference/commandline/load/
    - If you're using Linux, you may be able to set the `docker-images` directory as your working directory and use `find . -maxdepth 1 -type f -exec sudo docker load -i {} \;` to find and load in all the images

Certificates

When the application is started without any additional configuration, it will be served on both port `80` and on port `443` , with the NGINX container managing certificates.

- Replace the certificate files in `nginx/certs` with your own SSL certificate files. Failing to replace these will result in port `443` using invalid, self-signed certificates
- If you would like to use a different certificate management strategy than the one provided, the NGINX configuration template can be found at `nginx/templates/default.conf.template` , although we do not provide support for custom NGINX configurations

Environment variables & startup

- Set the `APP_DOCKER_HOST` environment variable. This is the hostname or IP address that browsers and units will connect to. The server will generate URLs pointed at this hostname that are then sent to the units, so ensure that it is accessible and will not change.
- Set the `TAG` environment variable to the version of Link-Live that you are running. This value can be found in `version.txt`
- Set the `TOKEN_PUBLIC_KEY` and `TOKEN_PRIVATE_KEY` environment variables. These are base64 representations of the RS256 PEM files that will be used to sign authentication JWTs
  - Note: These are not the base64 contents of the PEM files, but the entire PEM file encoded in base64
- Set `FNETID_TEST_USER` to an email address and `FNETID_TEST_SECRET` to a password. These will be used on container startup to create a superuser account, which is necessary for the application to function correctly.
  - Note: We do not recommend changing the password on this account, using it to claim units, or for other routine Link-Live functionality. Changing the `FNETID_TEST_USER` value on subsequent startups will create a new account.
- To start Link-Live, run `docker-compose up --build` . Add `-d` to run detached
  - Reference: https://docs.docker.com/compose/reference/up/

Alternatively, if you are on a Linux-based system, you may run the `start.sh` for an interactive walk through of this process.

Upgrade process

When a new version of Link-Live is released, we recommend the following steps:

1. Back up your existing data/volumes (see `Backups & more information`)
2. Download and extract the new version of Link-Live
3. Load in the new Link-Live images from their archives in the newly extracted `docker-images` directory. Alternatively, if you are on a Linux-based system, run the new `load-images-only.sh` script
4. Replace the old `docker-compose.yml` file with the new `docker-compose.yml` file. If you have made any modifications to the in-use `docker-compose.yml` file, propagate them to the new one.
5. Replace the old `version.txt` file with the new `version.txt` file. This is only necessary if you are leveraging the `start.sh` script
6. Stop any running Link-Live containers and start them again in the existing Link-Live directory (not in the newly extracted directory). See the `Environment variables & startup` section. The `start.sh` script can be used to start the containers after the upgrade

Backups & more information

- Data is stored persistently in the named Docker volumes `angular` , `ll-mongo` , `redis` , `bedrock-redis` , and `minio` Use
- `docker volume ls` to retrieve the specific volume names for your instance.
- Reference: https://docs.docker.com/storage/volumes/#backup-restore-or-migrate-data-volumes

Stopping

- `docker stop $(docker ps -aq)` can be used to stop all the containers