How to perform admin tasks

Backups

To backup your Oasis at least the file data and mongodb data needs to be saved. You determined the path to your file data (your uploads) during the installation. By default all data is stored in a directory called `.volumes` that is created in the current working directory of your installation/docker-compose. This directory can be backed up like any other file backup (e.g. rsync).

To backup the mongodb, please refer to the official mongodb documentation (https://docs.mongodb.com/manual/core/backups/). We suggest a simple mongodump export that is backed up alongside your files. The default configuration mounts `.volumes/mongo` into the mongodb container (as `/backup`) for this purpose. You can use this to export the NOMAD mongo database. Combine this with rsync on the `.volumes` directory and everything should be set. To create a new mongodump run:

```sh
docker exec nomad_oasis_mongo mongodump -d nomad_oasis_v1 -o /backup
```

The elasticsearch contents can be reproduced with the information in the files and the mongodb.

To create a new oasis with the backup data, create the `.volumes` directory from your backup. Start the new oasis. Use mongorestore:

```sh
docker exec nomad_oasis_mongo mongorestore /backup
```

Now you still have to recreate the elasticsearch index:

```sh
docker exec nomad_oasis_app python -m nomad.cli admin uploads index
```

```

Managing data with the CLI

The NOMAD command line interface (CLI) provides a few useful administrative functions. To use the NOMAD CLI, open a shell into the app container and run it from there:

```sh
docker exec -ti nomad_oasis_app bash
```

For example you can ls or remove uploads:

```sh
nomad admin uploads ls
nomad admin uploads rm --
```

You can also reset the processing (of "stuck") uploads and reprocess:

```sh
nomad admin uploads reset --
nomad admin uploads process --
```

You can also use the CLI to wipe the whole installation:

```sh
nomad admin reset --i-am-really-sure
```

Upload commands

The `nomad admin uploads` group of CLI commands allow you to inspect and modify all or some uploads in your installation. Sub-commands include `ls`, `rm`, `chown`, `process` (see below), `index` (see below).
```

The command group takes many different parameters to target specific subsets of uploads.

Here are a few examples:

- `--unpublished`
- `--published`
- `--outdated` Select published uploads with older NOMAD versions than the current
- `--processing-failure` Uploads with processing failures.

For a complete list refer to the CLI reference documentation (../../reference/cli.md#nomad-admin-uploads).

Alternatively, you can use a list of upload ids at the end of the command, e.g.:

```
nomad admin uploads ls --
```

If you have a list of ids (e.g. in a file), you could use `xargs`:

```
cat file_with_ids.txt | xargs nomad admin uploads ls --
```

Re-processing

Processing includes the conversion of raw files into NOMAD entries. Files are parsed, normalizers are called, the processing results are stored, and the search index is updated. In certain scenarios (failed processing, migration (migrate.md#migration-steps), changed plugins) might require that admins process certain uploads again.

```
nomad admin uploads process
```

Re-Indexing

Each NOMAD entry is represented in NOMAD's search index. Only if an entry is in this index, you can find it via the search interface. Some changes between NOMAD versions (see also our migration guide (migrate.md#migration-steps)), might require that you re-index all uploads.

```
nomad admin uploads index
```

Restricting access to your Oasis

An Oasis works exactly the same way the official NOMAD works. It is open and everybody can access published data. Everybody with an account can upload data. This might not be what you want.

Currently there are two ways to restrict access to your Oasis. First, you do not expose the Oasis to the public internet, e.g. you only make it available on an intra-net or through a VPN.

Second, we offer a simple white-list mechanism. As the Oasis administrator you provide a list of accounts as part of your Oasis configuration. To use the Oasis, all users have to be logged in and be on your white list of allowed users. To enable white-listing, you can provide a list of NOMAD account email addresses in your `nomad.yaml` like this:

```
oasis:
    allowed_users:
```

```
    - user1@gmail.com
    - user2@gmail.com
```

Configuring for performance

If you run the OASIS on a single computer, like described here (either with docker or bare
linux), you might run into problems with processing large uploads. If the NOMAD worker and app are run on the same computer, the app might become unresponsive, when the worker
consumes all system resources.

By default, the worker container might have as many worker processes as the system as CPU cores.

In addition, each worker process might spawn additional threads and consume
more than one CPU core.

There are multiple ways to restrict the worker's resource consumption:

- limit the number of worker processes and thereby lower the number of used cores
- disable or restrict multithreading
- limit available CPU utilization of the worker's docker container with docker

Limit the number of worker processes

The worker uses the Python package celery. Celery can be configured to use less than the
default number of worker processes (which equals the number of available cores). To use only
a single core only, you can alter the worker service command in the `docker-compose.yml` and
add a `--concurrency` argument:

```

command: python -m celery -A nomad.processing worker -l info --concurrency=1 -Q celery

```

See also the celery documentation (https://docs.celeryproject.org/en/stable/userguide/workers.html#id1).

Limiting the use of threads

You can also reduce the usable threads that Python packages based on OpenMP could use to

reduce the threads that might be spawned by a single worker process. Simply set the `OMP_NUM_THREADS`

environment variable in the worker container in your `docker-compose.yml`:

```

services:

  worker:

    ...

    environment:

      ...

      OMP_NUM_THREADS: 1

```

Limit CPU with docker

You can add a `deploy.resources.limits` section to the worker service in the `docker-compose.yml`:

```

services:

  worker:
```

```
    ...
    deploy:
      resources:
        limits:
          cpus: '0.50'
```

The number refers to the percentage use of a single CPU core.

See also the docker-compose documentation (https://docs.docker.com/compose/compose-file/compose-file-v3/#resources).