1 Build Documentation

In order to run the project:

1. the Appwrite docker swarm must be started
2. a project must be created in the Appwrite Console, with platform and API key
3. gather project-related environment variables
4. deploy the Smart Contract machinary
5. initialize Backend data like the Admin team and data base collections
6. the frontend must be built and started

On some Operating Systems like Ubuntu, you might need to write `sudo` before all the `docker` and "install"-related commands.

1.1 - Starting Appwrite v0.11

Generally you can use the Appwrite documentation here.

Please make sure to use the proper Appwrite version. At this point of writing, Appwrite has not reached a stable version and each version update is subject of significant changes.

We built our code on Appwrite v0.11. Many Appwrite functionalities in terms of API calls will not work if a different Appwrite version is used!!

If you don't have installed Appwrite already in the project's root directory (or another directory of your preference), you should do so first. You can use also the single CLI command as described in `/frontend/README.md`.

Otherwise, when the `appwrite/` folder is available together with `appwrite/.env` and `appwrite/docker-compose.yml`, you can start Appwrite within that folder via one command within the `appwrite/` folder:

```
docker-compose up -d
```

Where do I need to create the appwrite folder? It actually doesn't matter. You can have it in the `frontend` folder but the `backend` folder also works. But using a consistent locations seems to be important. Starting the Appwrite project from another location could cause Appwrite to behave like a fresh installment. Using a consistent location for your `docker-compose` call therefore is important.

It doesn't work?? Maybe you have used the command together with the `-f` option which allows you to use a YAML configuration from a different directory. If you do this it will use the environment variables of the directory where you executed the commmand and as there are maybe no `.env` variables in the current directory, it will not work.

It will not only start the Appwrite docker containers but also update running containers when `docker-compose.yml` or `.env` was changed.

The Appwrite project can be setup automatically as described below.

1.2 - Project Creation

This step can be skipped if this was done one time in the past. Just make sure to use the same location for `docker-compose` everytime when starting your Appwrite docker swarm.

You can check an online tutorial for how to setup a project:

Create an Appwrite Project And Dashboard – Walkthrough

1.3 - Environment variables

NFT-the-World requires some environment variables. You can find a list of those required for operation in `/frontend/.env`. If missing, the frontend defines own default values set in `/frontend/utils/config.js`.

There are additional environment variables needed for automatic Appwrite project initialization and configuration. You can find those in `/backend/setup_appwrite/.env`.

So before any scripts or the application can properly work, these environment variables must be set with values that you can obtain from the Appwrite console that was shown in step 2. Typical environment variables specifically used for our project start with `APPWRITE_...` or `REACT_APP_...`. Environment variables can be exported like so (replacing the `<project-ID>` with an actual string):

```
export APPWRITE_PROJECT=<project-ID>
```

1.3.1 Appwrite Environment Variables (Before continuing)

There are some other environment variables in the `appwrite/.env` file which are loaded when running Appwrite. Before you can add Admins to your team, you need to set the SMTP related environment variables there.

Example:

```
_APP_SMTP_HOST=mail.gmx.com
_APP_SMTP_PORT=587
_APP_SMTP_SECURE=TSL
_APP_SMTP_USERNAME=<your-email-address-which-permits-SMTP>
_APP_SMTP_PASSWORD=<your-plaintext-email-address-password;DONT-
SHARE!>
```

When you update these while running Appwrite then don't forget to execute

```
sudo docker-compose up
```

as well to let the changes take effect.

1.4 - Smart Contract Deployment

For deploying the contract, please refer to this README. (Make sure to follow the link on top of the README document.) We have been using Remix and Truffle for testing.

This step is unrelated to the following data initialization step which remembers blockchain ABIs in the database.

1.5 - Initialize and configure backend

We have a Github Workflow which can set up the backend application data by running `/backend/configuration/conigure_appwrite.py`.
It will add all important database collections ("relations") and the initial users as listed in `/backend/configuration/template.xslx`.

For manual server deployment you need to run it once yourself. You can consult `/backend/configuration/README.md` for an overview of required environment variables. This will be explained in-depth in the subsection.

Afterwards, you need to run the blockchain-crawler (see below).

1.5.1 Admin Teams

In order to use the privileged features in the frontend, an initial team of Admins must be added to the project. Automation of this process is part of the Python script `configuration/configure_appwrite.py`.

The python script requires three things of input. You may pass them as command line arguments. If one of this arguments is not provided, an environment variable is used by the Python script.

arguments

- Appwrite Endpoint URL, which is for example accessed by the frontend for API functionality. This typically is `<your-domain-IP:port>/v1` **including the protocol specification** (e.g. `http://`) and **without trailing slash!** otherwise it may output just "Not Found".
  - Environment variable: `APPWRITE_ENDPOINT`
  - Command line prefix: `--endpoint=...`

Deploying or testing locally? You can use `http://localhost:80` as domain + port.

- Appwrite Project ID. Copy the Project ID that is linked to the created project from step 2. You can copy the number from the URL while the project settings are opened. Example: `http://localhost/console/home?project=618eea46b90ef`

-> the Project ID is the hexadecimal number behind `?project=`, i.e. `618eea46b90ef`.

  - Environment variable: `APPWRITE_PROJECT`
  - command line prefix: `--projectid=...`

- API key. This also needs to be taken from the Appwrite console. Choose an API-Key with enough permissions for the features that you'd want to use. For simplicity, it's recommended to use a master key with all permissions enabled but you only need `team` and `user` permissions. You can add an API key via the Appwrite console if not available. The "secret" of the key is the required value. It can be displayed when clicking on "Show Secret" under the API key's name in the Appwrite console. The secret will be a long string of hexadecimal digits like `d783e2aa495a03575...`.

  - Environment variable: `APPWRITE_API_KEY`
  - command line prefix: `--apikey=...`

  My secret shows `false`?? Then you should delete the key and generate it again.

After you got the necessary project information, define the initial team of admins by editing `/backend/python_init_script/template.csv` (or the `.xlsx` excel file).

1.5.2 Database Collections

There are at least three database collections we use: **Announcements, ABIs and NFT-Drops**.

Database "collections" are a group of database documents with equal format (a "schema"). Live data is remembered and added to those collections via "database documents".

A fourth argument for the configuration script exists for this purpose.

- Environment Variable: `MAIN_CONTRACT`
- argument name: `--maincontract=`

(For more information, see `/backend/configuration/README.md`.

1.5.3 script execution

Finally start the script via following command from the root of the Git project.

```
python3 ./backend/configuration/configure_appwrite.py --endpoint=<...>\
 --projectid=<...> --apikey=<...>  --maincontract=<...>\
<path-to-CSV-or-XLS(X)-file>
```

or by passing shell variables to the script

```
APPWRITE_ENDPOINT="<...>" \
APPWRITE_PROJECT="<...>" \
APPWRITE_API_KEY="<...>" \
MAIN_CONTRACT="<...>" \
python3 ./backend/python_init_script/main.py <path-to-CSV-or-XLSX-file>
```

What to do when there is a module import error? Then you need to install some Python module dependencies first: `sh pip3 install -r ./backend/python_init_script/requirements.txt`

What file path to use? The path to your file could be `/backend/python_init_script/template.csv` (or `.xlsx`) if you edited it for this purpose.

Output says SMTP is disabled? It means you didn't set the SMTP environment variables `appwrite/.env`. Then run `docker-compose` as described at the end of this step.

It displays "missing scope"! If you see "missing scope" in the output it means, your API key didn't work – i.e. the API key doesn't exist, you're using a wrong value for `APPWRITE_API_KEY` or it misses the said permissions as written in the script's output.

1.5.4 update collection ID environment variables for frontend

After executing `configure_appwrite.py` you may update the `REACT_APP_` environment variables ending with `_ID` in `/frontend/.env`. In case of doubt, you can retrieve the ID from the Appwrite Web Console GUI.

1.5.5 Blockchain Crawler

There is a periodic background task which we run as a Cron Job with Docker. (For technical reasons; this causes less problems.)

It will update the Blockchain-ABIs and NFT-Drops periodically in the background to contain up-to-date live data. It's located in `/backend/cronjob-docker/` and it includes a simple README.md which lists the steps to run it in a per-minute-cycle.

You'll need to register for API keys of blockchain-related services (if you don't have an API key otherwise) to configure the arguments for the script.

1.6 - Building and running the frontend

After the environment variables are set in `/frontend/.env`, Admins teams and database collections were initialized, the frontend can be build in the last step. In the root of the git project, you can use following for building

```
docker build -t ${name} ./frontend/
```

and running

```
docker run --env-file .env -d -it -p ${port}:80 ${name}
```

the application. `${name}` could be for example `nftfrontend`. Choose any unique name that is distinguishable from your other docker containers. `${port}` could be `8181` which means that the application would be available at the URL `${APPWRITE_DOMAIN}:8181` which could be `localhost:8181` for example.

For more details, please have a look at `/frontend/README.md`.

2 Done

Now you can access *NFT-the-World* at your `APPWRITE_DOMAIN` with the `<port>` to which you bound the frontend's docker container.