VM setup - Update docker container

Overview

Assuming you have cloned the repository containing the preview server code (as described here), you can use the `update-preview-server.sh` script on the VM host to update the preview server based on changes in the source code.

The script will pull the latest changes from the origin's master branch and examine if there have been any changes in files inside the preview server source code directory (see below). If there are, it will create a new image and verify that it works as expected. Finally, it will stop and remove the old docker container and image, create a new container based on the new image and start it.

The script assumes that the preview server source code is in the repository's `aio/aio-builds-setup/` directory and expects the following inputs:

- **$1**: `HOST_REPO_DIR`
- **$2**: `HOST_SECRETS_DIR`
- **$3**: `HOST_BUILDS_DIR`
- **$4**: `HOST_LOCALCERTS_DIR`
- **$5**: `HOST_LOGS_DIR`

See here for more info on what each input directory is used for.

Note 1: The script has to execute docker commands with `sudo`.

Note 2: The script has to execute `yarn` commands, so make sure `yarn` is on the `PATH` when invoking the script.

Note 3: Make sure the user that executes the script has access to update the repository.

Run the script manually

You may choose to manually run the script, when necessary. Example:

```
update-preview-server.sh \
    /path/to/repo \
    /path/to/secrets \
    /path/to/builds \
    /path/to/localcerts \
    /path/to/logs
```

Run the script automatically

You may choose to automatically trigger the script, e.g. using a cronjob. For example, the following cronjob entry would run the script every 30 minutes, update the preview server (if necessary) and log its output to `update-preview-server.log` (assuming the user has the necessary permissions):

```
# Periodically check for changes and update the preview server (if necessary)
*/30 *  * * *   /path/to/update-preview-server.sh /path/to/repo /path/to/secrets /path/to/bu
```

Note: Keep in mind that cron jobs run in non-interactive, non-login shells. This means that the execution context might be different compared to when running the same commands from an interactive, login shell. For example, `.bashrc` files are normally *not* sourced automatically in cron jobs. See here for more info.