User guide for the repository `IO-TEMPLATE-APP`

`IO-TEMPLATE-APP` is a template repository for creating Python applications. This document describes how to use this repository to create a new repository. In the following instructions, we assume that the new repository should be named `my-repo` and the application to be created with it should be named `myapp`.

I. Requirements

Regarding operating system, Ubuntu version 20.04 and above and Windows version 10 and above are supported. An existing Python 3 installation is required. Furthermore, the use of an IDE or a text editor that can replace texts across files is useful.

II. Repository creation

1. Create the new repository `my-repo`

As described here, the new repository my-repo must first be created. The creation of a very minimal basic version is sufficient, i.e. the only necessary parameter is the repository name.

2. Copy the repository `io-template-app`

- Open Git Bash

- Create a bare clone of the repository.

  git clone --bare https://github.com/io-aero/io-template-app

---

- Mirror-push to the new repository

  cd io-template-app.git git push --mirror https://github.com/io-aero/my-repo

---

- Remove the temporary local repository you created earlier

  cd .. rm -rf io-template-app.git

3. Create a local copy of the new repositoy `my-repo`

```
git clone https://github.com/io-aero/my-repo
```

4. Delete the two files with the User's Guide

```
`user_guide.md`
`user_guide.pdf`
```

5. Rename the following file directories and files

Old name	New name
`iotemplateapp`	`myapp`
`run_io_template_app.bat`	`run_my_repo.bat`
`run_io_template_app.sh`	`run_my_repo.sh`
`settings_io_template_app.toml`	`settings_my_repo.toml`

4. Replacing texts in the new repository `my-repo`

It is absolutely necessary to respect the capitalization!

Old text	New text
`IO-TEMPLATE-APP`	`MY-REPO`
`IO_TEMPLATE_APP`	`MY_REPO`
`io-template-app`	`my-repo`
`io_template_app`	`my_repo`
`iotemplateapp`	`myrepo`

5. Store your AWS access rights in file `~/.aws/credentials`

```
[default]
aws_access_key_id=...
aws_secret_access_key=...
```

6. Test the current state of the new application

6.1 If Miniconda is required

- Install Miniconda
- Run `make conda-dev`
- Run `make-final`

6.2 If Miniconda is not required

- Run `make pipenv-dev`
- Run `make-final`

7. Define GitHub Actions secrets

Under 'settings' -> 'Secrets and variables' -> 'Actions' -> Tab 'Secrets' define the following 'New repository secret's:

```
AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY
GLOBAL_USER_EMAIL
```

8. Define GitHub repository variables

Under 'settings' -> 'Secrets and variables' -> 'Actions' -> Tab 'Variables' define the following 'New repository variable's:

Name	Value	Reason
CONDA	true	To get Miniconda installed

9. Commit and push all changes to the repository as 'Base version'