TP N°7 : Docker & Git

This TP will focus on implementing a web application to have basic visualizations of dataframes
<u>IMPORTANT NOTE</u> : Resources and Documentations comes after the lists of tasks

Tasks

0 - Experiment with Docker:
- Through your docker utility docker pull an image
- Docker run it through a specific port
- Access the web app of it

1 - Creating the Application :
- Create a streamlit application where you can upload a dataframe and visualize its content.

3 - Git
- Create a Github account (https://github.com/)
- Create a Repository for this TP.
- Link the remote repository you just created with your local project folder.
- Push your code to the remote

4 - Adding new features
- Create a new branch on your Github repository called “dev”.
- From that branch create a new branch called “feature/plot-df”.
- On your local folder fetch the changes and checkout on the new feature branch.
- Add a plot of the dataframe on your streamlit application.
- Push your changes and create a pull request from “feature/plot-df” to “dev”.
- Add yourself as a reviewer and accept-it, do the same from “dev” to “main”.
- Create a README.md file explaining how to run your code.

OPTIONAL :
- Create the Dockerfile for this application
- Build the image and deploy it locally
- Add docstrings to your code.
- Add an argument in your Dockerfile which defines which plot will be displayed on your application

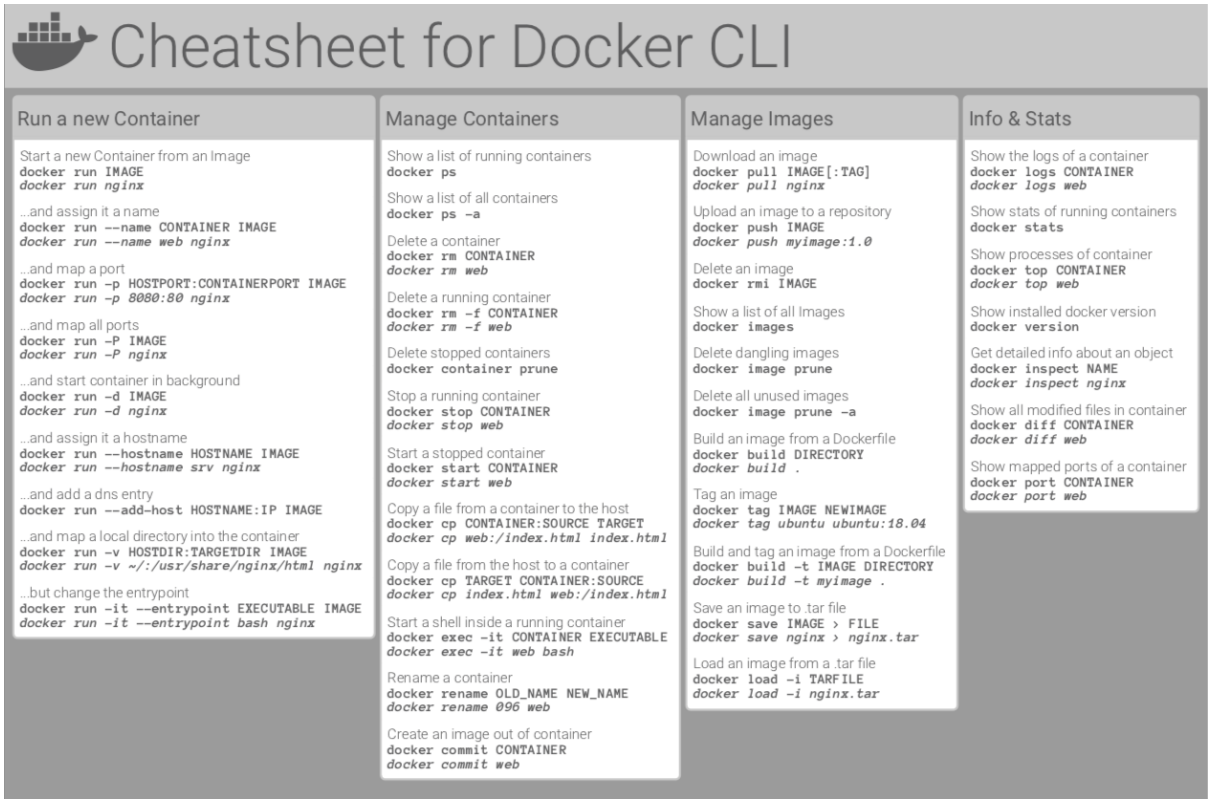
Resources

Docker

How to install :
Windows : https://docs.docker.com/desktop/install/windows-install/
Ubuntu : https://docs.docker.com/engine/install/ubuntu/

If you encounter a problem during installation use the docker playground environment
https://www.docker.com/play-with-docker/

Commands :



Streamlit

Create rapidly an app : https://docs.streamlit.io/library/get-started/create-an-app
Create a file uploader : https://docs.streamlit.io/library/api-reference/widgets/st.file_uploader
Display a table : https://docs.streamlit.io/library/api-reference/data/st.dataframe
Do a bar chart : https://docs.streamlit.io/library/api-reference/charts/st.bar_chart
Documentation : https://docs.streamlit.io/

Git

Git Cheat Sheet

Remember!
`git <COMMAND> --help`

Global configuration is stored in ~/.gitconfig.
`git config --help`

master is the default development branch.
origin is the default upstream repository.

Create

From existing data
```
cd ~/my_project_directory
git init
git add .
```

From existing repository
```
git clone ~/existing_repo ~/new/repo
git clone git://host.org/project.git
git clone ssh://user@host.org/project.git
```

Show

Files changed in working directory
`git status`

Changes made to tracked files
`git diff`

What changed between ID1 and ID2
`git diff <ID1> <ID2>`

History of changes
`git log`

History of changes for file with diffs
`git log -p <FILE> <DIRECTORY>`

Who changed what and when in a file
`git blame <FILE>`

A commit identified by ID
`git show <ID>`

A specific file from a specific ID
`git show <ID>:<FILE>`

All local branches
`git branch`
star (*) marks the current branch

Revert

Return to the last commited state
`git reset --hard`
This cannot be undone!

Revert the last commit
`git revert HEAD`
Creates a new commit

Revert specific commit
`git revert <ID>`
Creates a new commit

Fix the last commit
`git commit -a --amend`
(after editing the broken files)

Checkout the ID version of a file
`git checkout <ID> <FILE>`

Update

Fetch latest changes from origin
`git fetch`
(this does not merge them)

Pull latest changes from origin
`git pull`
(does a fetch followed by a merge)

Apply a patch that someone sent you
`git am -3 patch.mbox`
In case of conflict, resolve the conflict and
`git am --resolved`

Publish

Commit all your local changes
`git commit -a`

Prepare a patch for other developers
`git format-patch origin`

Push changes to origin
`git push`

Make a version or milestone
`git tag v1.0`

Branch

Switch to a branch
`git checkout <BRANCH>`

Merge BRANCH1 into BRANCH2
```
git checkout <BRANCH2>
git merge <BRANCH1>
```

Create branch BRANCH based on HEAD
`git branch <BRANCH>`

Create branch BRANCH based on OTHER and switch to it
`git checkout -b <BRANCH> <OTHER>`

Delete branch BRANCH
`git branche -d <BRANCH>`

Resolve merge conflicts

View merge conflicts
`git diff`

View merge conflicts against base file
`git diff --base <FILE>`

View merge conflicts against your changes
`git diff --ours <FILE>`

View merge conflicts against other changes
`git diff --theirs <FILE>`

Discard a conflicting patch
```
git reset --hard
git rebase --skip
```

After resolving conflicts, merge with
```
git add <CONFLICTING_FILE>
git rebase --continue
```

Workflow