

TP N°7 : Docker & Git

This TP will focus on implementing a web application to have basic visualizations of dataframes

IMPORTANT NOTE : 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 :

 Cheatsheet for Docker CLI			
Run a new Container	Manage Containers	Manage Images	Info & Stats
<p>Start a new Container from an Image</p> <pre>docker run IMAGE docker run nginx</pre> <p>...and assign it a name</p> <pre>docker run --name CONTAINER IMAGE docker run --name web nginx</pre> <p>...and map a port</p> <pre>docker run -p HOSTPORT:CONTAINERPORT IMAGE docker run -p 8080:80 nginx</pre> <p>...and map all ports</p> <pre>docker run -P IMAGE docker run -P nginx</pre> <p>...and start container in background</p> <pre>docker run -d IMAGE docker run -d nginx</pre> <p>...and assign it a hostname</p> <pre>docker run --hostname HOSTNAME IMAGE docker run --hostname srv nginx</pre> <p>...and add a dns entry</p> <pre>docker run --add-host HOSTNAME:IP IMAGE</pre> <p>...and map a local directory into the container</p> <pre>docker run -v HOSTDIR:TARGETDIR IMAGE docker run -v ~/.usr/share/nginx/html nginx</pre> <p>...but change the endpoint</p> <pre>docker run -it --entrypoint EXECUTABLE IMAGE docker run -it --entrypoint bash nginx</pre>	<p>Show a list of running containers</p> <pre>docker ps</pre> <p>Show a list of all containers</p> <pre>docker ps -a</pre> <p>Delete a container</p> <pre>docker rm CONTAINER docker rm web</pre> <p>Delete a running container</p> <pre>docker rm -f CONTAINER docker rm -f web</pre> <p>Delete stopped containers</p> <pre>docker container prune</pre> <p>Stop a running container</p> <pre>docker stop CONTAINER docker stop web</pre> <p>Start a stopped container</p> <pre>docker start CONTAINER docker start web</pre> <p>Copy a file from a container to the host</p> <pre>docker cp CONTAINER:SOURCE TARGET docker cp web:/index.html index.html</pre> <p>Copy a file from the host to a container</p> <pre>docker cp TARGET CONTAINER:SOURCE docker cp index.html web:/index.html</pre> <p>Start a shell inside a running container</p> <pre>docker exec -it CONTAINER EXECUTABLE docker exec -it web bash</pre> <p>Rename a container</p> <pre>docker rename OLD_NAME NEW_NAME docker rename 096 web</pre> <p>Create an image out of container</p> <pre>docker commit CONTAINER docker commit web</pre>	<p>Download an image</p> <pre>docker pull IMAGE[:TAG] docker pull nginx</pre> <p>Upload an image to a repository</p> <pre>docker push IMAGE docker push myimage:1.0</pre> <p>Delete an image</p> <pre>docker rmi IMAGE</pre> <p>Show a list of all Images</p> <pre>docker images</pre> <p>Delete dangling images</p> <pre>docker image prune</pre> <p>Delete all unused images</p> <pre>docker image prune -a</pre> <p>Build an image from a Dockerfile</p> <pre>docker build DIRECTORY docker build .</pre> <p>Tag an image</p> <pre>docker tag IMAGE NEWIMAGE docker tag ubuntu ubuntu:18.04</pre> <p>Build and tag an image from a Dockerfile</p> <pre>docker build -t IMAGE DIRECTORY docker build -t myimage .</pre> <p>Save an image to tar file</p> <pre>docker save IMAGE > FILE docker save nginx > nginx.tar</pre> <p>Load an image from a tar file</p> <pre>docker load -i TARFILE docker load -i nginx.tar</pre>	<p>Show the logs of a container</p> <pre>docker logs CONTAINER docker logs web</pre> <p>Show stats of running containers</p> <pre>docker stats</pre> <p>Show processes of container</p> <pre>docker top CONTAINER docker top web</pre> <p>Show installed docker version</p> <pre>docker version</pre> <p>Get detailed info about an object</p> <pre>docker inspect NAME docker inspect nginx</pre> <p>Show all modified files in container</p> <pre>docker diff CONTAINER docker diff web</pre> <p>Show mapped ports of a container</p> <pre>docker port CONTAINER docker port web</pre>

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 committed 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 branch -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

