Windows, WSL 2 & Bind Mounts

When working with Windows and WSL2, you can encounter issues when it comes to working with "Bind Mounts".

> *Please note, that these instructions **only** apply if you are using Windows with WSL2 and Docker. NOT if you're not using WSL2!*

Specifically, file change events might not be propagated to the container - i.e. processes running inside of the container (like "nodemon") are not made aware of file changes.

There are three main workarounds & solutions:

1) Quick & Dirty

This solution is "quick and dirty" but it does the job.

For Nodemon, you can switch from the "notify me" mode to the "I look for changes myself mode" by changing the starting script in the `package.json` file like this:

```
{
  "scripts": "nodemon -L server.js"
}
```

For React apps (where we also want to listen to file changes), you can pull a similar trick. Simply add an environment variable to your `docker run` command:

```
docker run -e CHOKIDAR_USEPOLLING=true ...other-options <react-image-id>
```

2) Mount Your Windows Paths Into WSL2

This solution might require an update to the latest Windows Insiders version - to be precise, you need at least this version: https://blogs.windows.com/windows-insider/2020/09/10/announcing-windows-10-insider-preview-build-20211/

Once you got that, you can follow the steps outlined in this article to mount your Windows path / folder into WSL2: https://devblogs.microsoft.com/commandline/access-linux-filesystems-in-windows-and-wsl-2/

As a result, file changes will be forwarded into the container.

3) Embrace WSL2

This is theoretically the best solution, but it's also an advanced solution.

"Embrace WSL2" means, that you should use WSL2 as your default terminal / command prompt (instead of the regular Windows command prompt).

Do **all your work** from inside WSL2 (and therefore from inside Linux) and you'll not face any issues.

What does that mean?

Launch your Linux distribution (and therefore WSL2) via the Windows starting menu.

This opens up a Linux shell which allows you to work with the Linux distribution which was installed on your Windows system. You also got full access to its internal file system there.

Navigate around via the `ls` (list content of the current directory) and `cd` (change directory) commands.

You can create a new directory via the `mkdir` command.

For example, you could create a `docker-practice` directory like this:

```
cd ~
mkdir docker-practice
cd docker-practice
```

Now when it comes to working with these folders (e.g. to create your Dockerfile and your project files in there), the following article can be a great next step: https://www.docker.com/blog/docker-desktop-wsl-2-best-practices/.

4) Deploy project directly in linux env