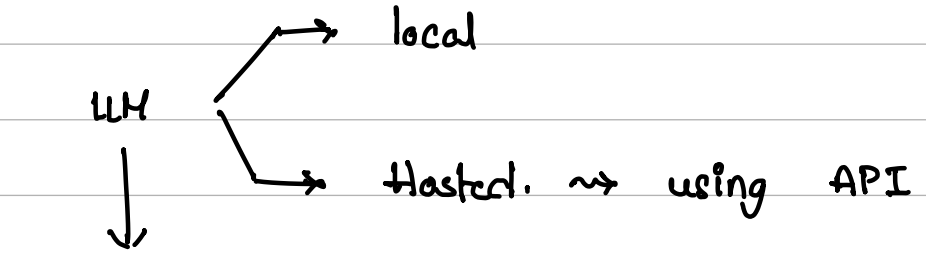
→ AI Project - Great Docker files using LLM's.



local	Hosted
- Security	- data is revelead
- privacy.	- Cost for API
- Hard to maintain infrastructure	

first Approach : local LLM's

Download { ollama ~~> used to Docker

has a cli to run
- it has ollama hub.

- First 'pull'
- Second 'run'

Steps :

1. Install ollama
2. Identify
3. pull
4. Setup the python env.
5. write the python script.

Virtual environment ~> isolation of packages

Ex: we can use different versions of the same package based on the project.

* python script:

```
import ollama

promt = """
        ==========
"""

# Api Calls

def generate_dockerfile(language):
    response=ollama.chat(
                model = "ollama.3.2:1b",
                message = [{'role': 'user', 'content': 'promt'}]
                )
    return response['message']['content']

if __name__ == '__main__':
    language = input()
    print(generate_dockerfile(language))
```

~> multi line script

Example of the prompt:

prompt = """

Only generate a dockerfile for {language} with best practices. Do not provide any description.

Include:

- Base image
- Installing dependencies
- setting working directory
- Adding source code
- Running the application

"""

Approach 2: Hosted LLM

API Calls ~> tokens as cost

free resource: Google AI Studio ~> Gemini pro 1.5

→ write the python script.

structure of the python file:

- import
- set up API key
- configure ~> (mean logic)
- prompt
- def function
- output

Interview tip:

Q: How do you control & avoid the high charges that will be done by a developer if he uses excessive tokens?

Ans: We can use the state limits and token count limitation to avoid high charges.