1 — One approach is to consider rewriting the entire ML codebase in a language that suits the software engineering team. While this might seem like a logical solution, it can be incredibly time-consuming and resource-intensive, especially when dealing with complex models. Additionally, some languages, such as JavaScript, lack robust ML libraries, making this option less appealing.

2 — Another, more practical approach is to adopt an API-first strategy. Web APIs have revolutionized the way applications interact across languages. By exposing your ML models through a web API, you provide a seamless bridge between data science and software engineering. Frontend developers, for example, can access your ML model via a simple URL endpoint, enabling them to integrate machine learning capabilities into web applications effortlessly.

Before delving deeper into the API-first approach, let's take a moment to understand what an API really is and how it can serve as a powerful connector between different technology stacks.

What are APIs?

"In simple words, an API is a (hypothetical) contract between 2 software's saying if the user software provides input in a pre-defined format, the later with extend its functionality and provide the outcome to the user software."

Essentially, APIs function much like web applications, though with a distinct focus on data exchange rather than presenting information in a user-friendly HTML format. Instead, they typically return data in standardized formats like JSON or XML. Once a developer obtains the desired output from an API, they have the creative freedom to style and present it according to their specific preferences and requirements. This flexibility empowers developers to integrate data seamlessly into their applications while maintaining full