**LinkedInArticle:https://www.linkedin.com/posts/mahmoud-noaman9\_webdevelopment-rest-grpc-activity-7309225616220852225-LS\_R?utm\_source=share&utm\_medium=member\_desktop&rcm=ACoAAD7TFF8BY\_vB0T5SvzWLS4-l9vbzSeNT3MM**

**What’s the Difference Between gRPC and REST?**

gRPC and REST are two ways you can design an API. An API is a mechanism that enables two software components to communicate with each other using a set of definitions and protocols. In gRPC, one component (the client) calls or invokes specific functions in another software component (the server). In REST, instead of calling functions, the client requests or updates data on the server.

**REST vs gRPC: What’s the Difference?**

|  |  |  |
| --- | --- | --- |
| **Feature** | **REST** | **gRPC** |
| **Protocol** | Uses HTTP/HTTPS. | Uses HTTP/2. |
| **Message Format** | JSON (text-based). | Protocol Buffers (binary). |
| **Performance** | Slower due to JSON parsing. | Faster due to binary serialization. |
| **Communication** | Text-based, human-readable. | Binary, machine-efficient. |
| **Streaming Support** | Limited (uses WebSockets for real-time updates). | Native support for bi-directional streaming. |
| **Best For** | Web APIs, interoperability. | High-performance microservices, real-time applications. |
| **Tooling & Support** | Widely supported across all platforms. | Requires specific support but works well with modern frameworks. |

**When to Use REST vs gRPC?**

**Use REST when:**

You need a simple, widely-supported API for web and mobile apps.

Human readability of requests and responses is important.

You need to support multiple clients (JavaScript, Python, PHP, etc.).

**Use gRPC when:**

Performance and speed are critical (e.g., microservices communication).

You need real-time bi-directional streaming.

You work with strongly typed contracts using Protocol Buffers.