**Web Sockets**

* Protocol that provides full-duplex communication channels over a single, long-lived connection.
* Unlike traditional HTTP connections, which require a request-response cycle, Web Sockets enable two-way communication between a client and a server without the need for repeated requests.

**Key Concepts**:

1. **Full-Duplex**: Both client and server can send messages independently, without waiting for a response.
2. **Persistent Connection**: The connection remains open until either the client or server closes it.
3. **Low Latency**: Ideal for situations where latency needs to be minimized.

**WebSocket vs Secure WebSocket**

* **ws://** → WebSocket over plain TCP (no encryption).
* **wss://** → WebSocket over **TLS/SSL** (same security layer as HTTPS).

So when you use wss:// in production, you’re essentially running WebSockets **inside an encrypted TLS tunnel**.

**How wss:// protects data in transit**

* **Encryption**

1. All messages exchanged between client and server are **encrypted with TLS**.
2. Prevents attackers from eavesdropping on sensitive data (passwords, tokens, messages, etc.).

* **Integrity**

1. TLS ensures that data cannot be **modified in transit** without detection.
2. Protects against **man-in-the-middle (MITM) attacks**.

* **Authentication**

1. TLS certificates verify the server’s identity (just like HTTPS).
2. The client can trust it’s talking to the real server, not an imposter.

**Why this matters in production**

* Without wss://, data sent over ws:// is **plain text** and can be intercepted (e.g., on public Wi-Fi, corporate proxies, ISP monitoring).
* Modern browsers **block mixed content** (e.g., a secure HTTPS site cannot connect to an insecure ws:// WebSocket).
* Security compliance (GDPR, HIPAA, PCI, etc.) **requires encryption in transit**.

**Best Practices for Secure WebSockets**

* Always use **wss://** in production.
* Use valid TLS certificates (from Let’s Encrypt, DigiCert, etc.).
* Terminate TLS either:
  + At the **application server** (Node.js, etc.), or
  + At a **reverse proxy/load balancer** (e.g., Nginx, HAProxy, AWS ALB, Azure App Gateway).
* Authenticate WebSocket clients (e.g., via JWT token in the connection request).
* Use **secure headers** (Sec-WebSocket-Protocol for subprotocols, CSRF tokens, etc.) if needed.