**What is JWT?**

JWT, or JSON Web Token, is used for authorization, not authentication. While authentication verifies user credentials (like username and password), authorization checks if the user has permission to access a specific resource. JWT is a method of stateless authorization that replaces traditional session management.

**Traditional Session-Based Authorization:**

In traditional systems:

1. The user logs in by sending their credentials (username and password) to the server.
2. The server verifies the credentials and, if correct, stores the user data in a session on the server, generating a unique session ID.
3. The session ID is sent back to the client in a cookie. Every subsequent request includes this session ID, allowing the server to identify the user and authorize access.

**How JWT Works:**

1. Instead of storing session data on the server, JWT stores user data in a token on the client side.
2. After successful authentication, the server creates a JWT containing user information, signs it with a secret key, and sends it to the client.
3. The client stores the JWT (e.g., in local storage or a cookie) and sends it with each request.
4. The server verifies the token by checking its signature to ensure it hasn't been tampered with. If valid, it extracts the user information directly from the token.

**Advantages of JWT:**

* **Stateless**: JWT does not require server storage, making it ideal for scalable applications.
* **Cross-Server Authorization**: JWT can be used across multiple servers or services without the need for shared sessions, which is useful in distributed systems, microservices, or when a user needs to be authenticated across different applications.
* **Security**: JWT is signed by the server, preventing tampering. The expiration (exp) field can limit the token's lifespan, enhancing security.

**Use Cases:**

1. **Multi-Server Environments**: Allows a user to stay logged in across different services or servers (e.g., bank and retirement servers).
2. **Load Balancing**: No need to transfer session data between servers; JWT enables seamless switching between servers.
3. **Microservices**: JWT allows different services (e.g., API, web server) to use the same authentication token, simplifying architecture.

**Conclusion:**

JWT is useful for authorization in stateless applications, especially those using microservices or requiring cross-server authentication. It simplifies user management by storing user information on the client and verifying it through signed tokens.