**Method**

**GET:**

* The GET method requests a representation of the specified resource.
* Requests using GET should only retrieve data.
* It is primarily used for safe and idempotent operations, meaning it should not change the server's state.

**POST:**

* The POST method submits an entity to the specified resource.
* It often causes a change in state or side effects on the server.
* It is used for operations that are not necessarily idempotent and may result in the creation of a new resource.

**HEAD:**

* The HEAD method asks for a response identical to a GET request but without the response body.
* It is used to retrieve metadata about the requested resource without the actual data.
* Useful for checking resource headers or determining if a resource has been modified without fetching the entire content.

**PUT:**

* The PUT method replaces all current representations of the target resource with the request payload.
* It is used to update or create a resource at a specific URI.
* The entire resource is replaced, and if it doesn't exist, a new resource is created.

**DELETE:**

* The DELETE method deletes the specified resource.
* It is used to request removal of a resource identified by a URI.
* It is considered idempotent, meaning multiple identical requests should have the same effect as a single request.

**HTTP Version**

**HTTP Version 0.9 (The-One-Line Protocol):**

* Basic one-line protocol.
* Very simple and lacked features.
* Introduced the GET method.

**HTTP Version 1.0:**

* Added new methods: POST and HEAD.
* Still relatively simple compared to later versions.
* Primarily a request-response model.

**HTTP Version 1.1:**

* Added six new methods: PUT, PATCH, DELETE, CONNECT, TRACE, and OPTIONS.
* Persistent connections to reuse the same connection for multiple requests.
* Improved host header support.
* Enhanced caching mechanisms.

**HTTP Version 2.0:**

* Introduced request multiplexing, allowing asynchronous sending of requests and receiving of responses over a single connection.
* Binary framing for more efficient data transfer.
* Header compression to reduce overhead.
* Server push to send multiple responses for a single request.

**HTTP Version 3.0 (Draft as of 2020):**

* Named HTTP/3.0.
* Transport protocol: QUIC (Quick UDP Internet Connections).
* Supports request multiplexing with improved mechanisms.
* Faster connection establishment and reduced latency.
* Enhanced security with built-in encryption.
* Stream prioritization for efficient handling of requests.
* More efficient header compression.
* Graceful fallback mechanism to HTTP/2.0 or HTTP/1.1 if needed.

**LogFormat**

* **%v**: Represents the server name, the canonical name of the server serving the request.
* **%p**: Represents the port number of the server.
* **%h**: Represents the remote hostname, the DNS name of the client making the request.
* **%l**: Represents the remote logname of the user. It's often not logged due to privacy concerns.
* **%u**: Represents the username of the user making the request. It's often - for privacy.
* **%t**: Represents the timestamp of the request in the format [day/month/year:hour:minute:second zone].
* **%r**: Represents the request line from the client, enclosed in double quotes. This includes the HTTP method (GET, POST, etc.) and the requested URI.
* **%>s**: Represents the status code of the response from the server.
* **%O**: Represents the number of bytes sent by the server to the client in response to the request.
* **"%{Referer}i"**: Represents the referer header sent by the client, indicating the URL of the page that referred the request.
* **"%{User-Agent}"**: Represents the user-agent header sent by the client, indicating the browser or user-agent making the request.