HTTP methods

GET

The most commonly used HTTP method is GET.
The purpose of the GET method is to simply retrieve data from the server. The GET method is used to request any of the following resources:

POST

The POST HTTP request method sends data to the server for processing.
The data sent to the server is typically in the following form:

DELETE

The HTTP DELETE method is self-explanatory. After execution, the resource a DELETE operation points to is removed from the server.
As with PUT operations, the HTTP DELETE method is idempotent and unsafe.

POST

The POST HTTP request method sends data to the server for processing.
The data sent to the server is typically in the following form:

HEAD

The HTTP HEAD method simply returns metadata about a resource on the server. This HTTP request method returns all of the headers associated with a resource at a given URL, but does not actually return the resource.
The HTTP HEAD method is commonly used to check the following conditions:
The size of a resource on the server.
If a resource exists on the server or not.
The last-modified date of a resource.
Validity of a cached resource on the server.
The following example shows sample data returned from a HEAD request:

PUT

The HTTP PUT method is used to completely replace a resource identified with a given URL.
The HTTP PUT request method includes two rules:
A PUT operation always includes a payload that describes a completely new resource definition to be saved by the server.
The PUT operation uses the exact URL of the target resource.
If a resource exists at the URL provided by a PUT operation, the resource's representation is completely replaced. If a resource does not exist at that URL, a new resource is created.

TRACE

The TRACE HTTP method is used for diagnostics, debugging and troubleshooting. It simply returns a diagnostic trace that logs data from the request-response cycle.
The content of a trace is often just an echo back from the server of the various request headers that the client sent

PATCH

Sometimes object representations get very large. The requirement for a PUT operation to always send a complete resource representation to the server is wasteful if only a small change is needed to a large resource.
The PATH HTTP method, added to the Hypertext Transfer Protocol independently as part of RFC 5789, allows for updates of existing resources. It is significantly more efficient, for example, to send a small payload rather than a complete resource representation to the server.

HTTP VERSION

1-HTTP/0.9 – The one-line protocol
The initial version of HTTP had no version number; it was later called 0.9 to differentiate it from later versions. HTTP/0.9 was extremely simple: requests consisted of a single line and started with the only possible method GET followed by the path to the resource. The full URL wasn't included as the protocol, server, and port weren't necessary once connected to the server

2-HTTP/1.0 – Building extensibility
HTTP/0.9 was very limited, but browsers and servers quickly made it more versatile:

Versioning information was sent within each request (HTTP/1.0 was appended to the GET line).
A status code line was also sent at the beginning of a response. This allowed the browser itself to recognize the success or failure of a request and adapt its behavior accordingly. For example, updating or using its local cache in a specific way.
The concept of HTTP headers was introduced for both requests and responses. Metadata could be transmitted and the protocol became extremely flexible and extensible.
Documents other than plain HTML files could be transmitted thanks to the Content-Type header.

3-HTTP/1.1 – The standardized protocol
In the meantime, proper standardization was in progress. This happened in parallel to the diverse implementations of HTTP/1.0. The first standardized version of HTTP, HTTP/1.1, was published in early 1997, only a few months after HTTP/1.0.

HTTP/1.1 clarified ambiguities and introduced numerous improvements:

LOG FILES

In your Apache configuration files, you can specify the variables you want to see in the access log.
A detailed treatment of these configuration options is beyond the scope of this post, but we'll give you a quick overview of the options we've just used above, along with other common options:
%h – Remote host (client IP address)
%l – User identity, or dash, if none (often not used)
%u – Username, via HTTP authentication, or dash if not used
%t – Timestamp of when Apache received the HTTP request
"%r – The actual request itself from the client
%>s – The status code Apache returns in response to the request
%b – The size of the request in bytes.
"%{Referer}i" – Referrer header, or dash if not used (In other words, did they click a URL on another site to come to your site)
"%{User-agent}i – User agent (contains information about the requester's browser/OS/etc)
As you can see, this is powerful stuff. You can get an awful lot of information about requests to your site and the people making them.