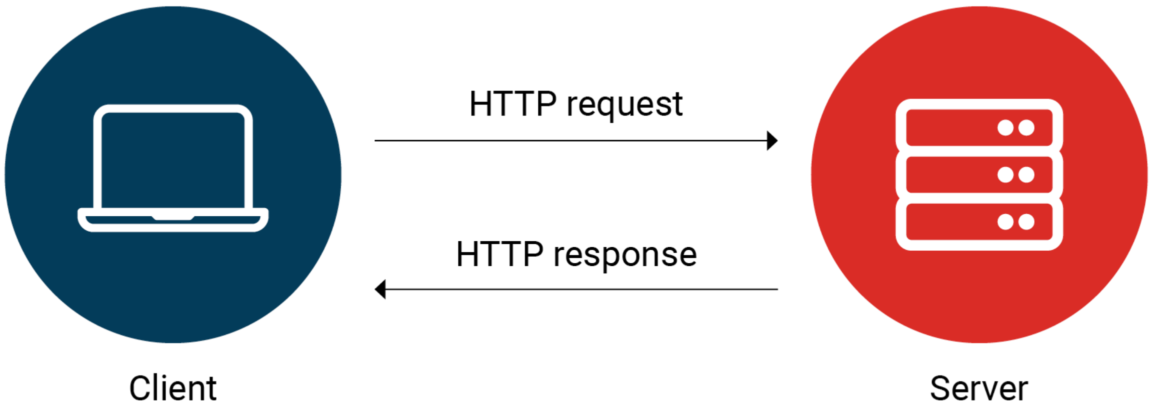
3.3.1 HTTP

When you use APIs, all of the data you need is generally sitting on someone else's computer. Essentially, that computer is now the server you want to connect to.

But how does that happen?

HTTP or HyperText Transfer Protocol is a protocol used to send and receive content, including images, HTML, and video, over the internet. It's the underlying communication framework at work when you surf the web; it is how your web browser, say Safari or Chrome, communicates with a web server. A web server retrieves content and brings it to the user or client. You, the client, would make a request to the server, and the server returns and displays a response.

This diagram represents a simplified view of HTTP communications: HTTP facilitates the sending and receiving of data.



COS60016: HTTP request (n.d.) adapted from BytesofGigabytes.

APIs depend on HTTP communications to make them functional. This week, you'll learn about different HTTP methods or verbs, which issue commands for different actions. The most common of these are as follows:

- `GET`: get a resource.
- `POST`: create a new resource.
- `HEAD`: receive the status of the resource.

- `PUT`: update an existing resource.
- `DELETE`: delete a resource.
- `PATCH`: replace an existing resource with new information.

You'll learn how to use these later this week. Before we get to that, though, it's helpful to have some context about how HTTP works and how it fits into the larger infrastructure of the internet.

HTTP's interactions are slightly more complicated than depicted in the image above. To learn more, watch this video:

COS60016: HTTP requests (2021) created by Swinburne Online.

Transcript (https://swinburneonline.instructure.com/courses/5055/pages/transcript-http-requests)

Select a term below to see a definition of three significant HTTP basics. Note that this is not key to your understanding of HTTPs and APIs but is useful contextual information to have.

Client or user-agent

The user-agent is any tool that works for the user. Generally, this tool is a web browser. The web browser sends a request to a server to fetch an HTML document, or web page, parses the document—executing scripts, confirming layout, and identifying subresources while doing so—and then displays the page to you, the user.

Servers

The server is the entity that returns, or serves, the requested HTML document. It sends a **response** to the **request**. An HTTP server knows how to read web addresses, or URLs, and, obviously, it understands HTTP. The server is accessed through the website domain names it contains, and, when requested, delivers the content to the client.

Proxies

Proxies are the many machines and computers that sit between the browser and the server. Proxies primarily relay any HTTP messages sent, but they can also cache and filter data, log history, authenticate access, and balance loads across multiple servers.

How does the internet work?

Understanding HTTP goes some way to understanding how the internet works. And how the internet works is a question many don't have a clear answer for! Do you have an idea of what happens when you search for a link on your browser and hit Enter? If not, please read **What happens when you type a URL in the browser and press enter? (https://medium.com/@maneesha.wijesinghe1/what-happens-when-you-type-an-url-in-the-browser-and-press-enter-bb0aa2449c1a)** to know more (Wijesinghe 2017). There are several other concepts like Domain Name System (DNS), load balancers, caching, databases, web application servers, and so on, where it is useful to have a high-level idea about. These concepts are well covered in **Web Architecture 101 (https://medium.com/storyblocks-engineering/web-architecture-101-a3224e126947)** (Fulton 2017).

HTTPS

HTTPS, or HyperText Transfer Protocol Secure, is, as the name suggests, the secure version of HTTP. In HTTPS, data is encrypted to ensure security of data transfer—particularly necessary when dealing with sensitive data, such as personal information. In a regular HTTP connection, when your browser is connected to a server and data is being sent back and forth, the entire process is relatively open—any eavesdropper can listen in.

HTTPS uses protocols to encrypt data. The protocol uses a 'handshake' process to authenticate the client and server and exchange keys to encrypt and decrypt the data. Without the correct key, it is nearly impossible to break the encryption and read the data.

So what does security mean for APIs? If you're considering consuming an API or integrating it into your system, you should ensure that it uses HTTPS. Accessing an API using HTTP means risking your data—you'll see examples later this week of the consequences of inadequate security in APIs. Moreover, you should make sure that any APIs you create necessarily use HTTPS rather than HTTP.

Additional resources

Cybersecurity is an extensive, well-documented topic. Read **10 Most Common Web Security Vulnerabilities** **(https://www.toptal.com/security/10-most-common-web-security-vulnerabilities)** (Williams 2022) to learn about some of the issues that particularly affect HTTP.

You can learn more about HTTPS encryption protocols by reading **SSL and SSL Certificates Explained For Beginners** **(http://www.steves-internet-guide.com/ssl-certificates-explained/)** (Cope 2021).

In your career as a programmer, you may encounter HTTP APIs that you become accountable for. Don't worry, they're not beyond repair. **Migrating APIs to HTTPS** **(https://https.cio.gov/apis/)** (CIO n.d.) outlines how you can migrate APIs to HTTPS.