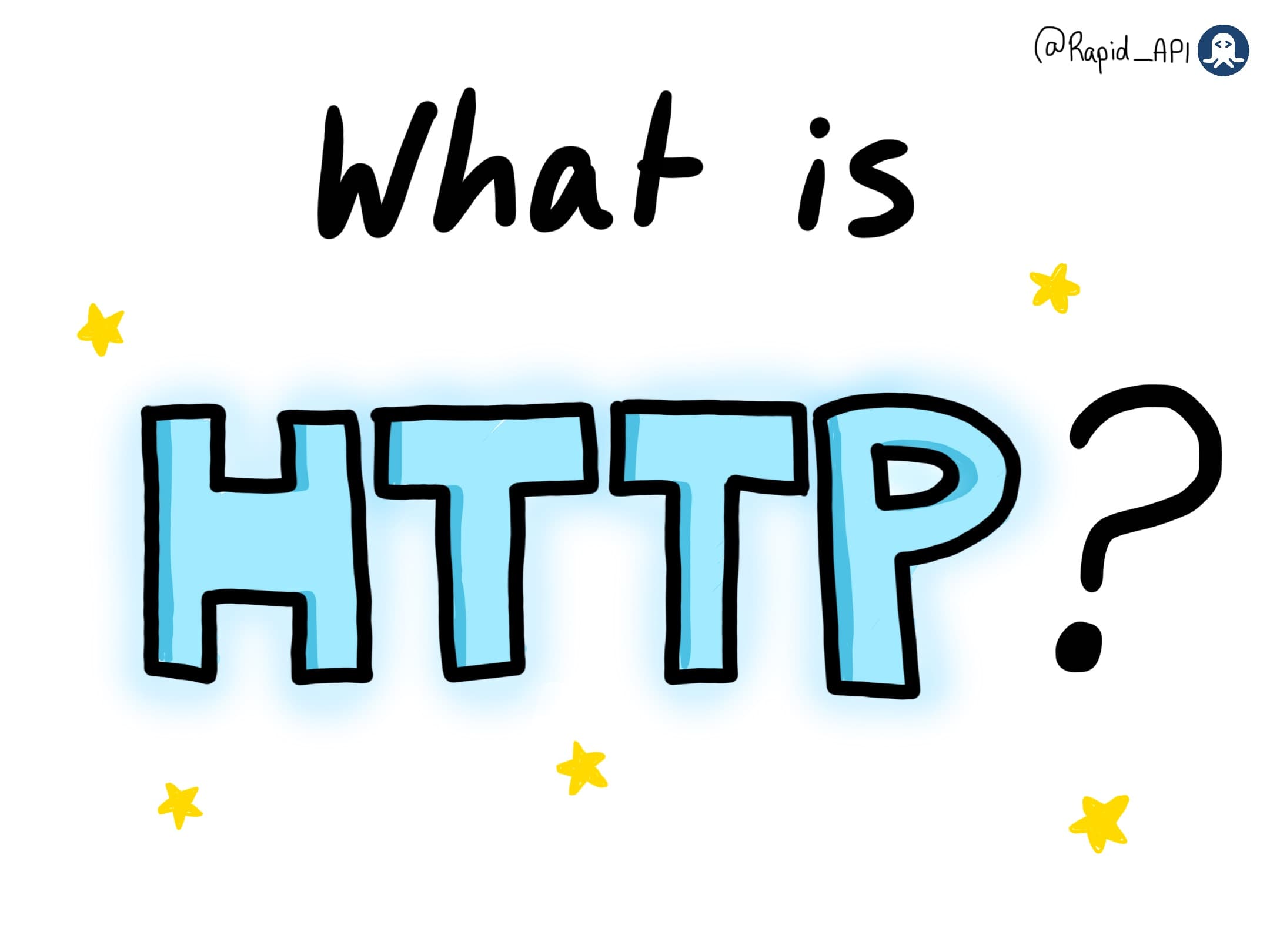
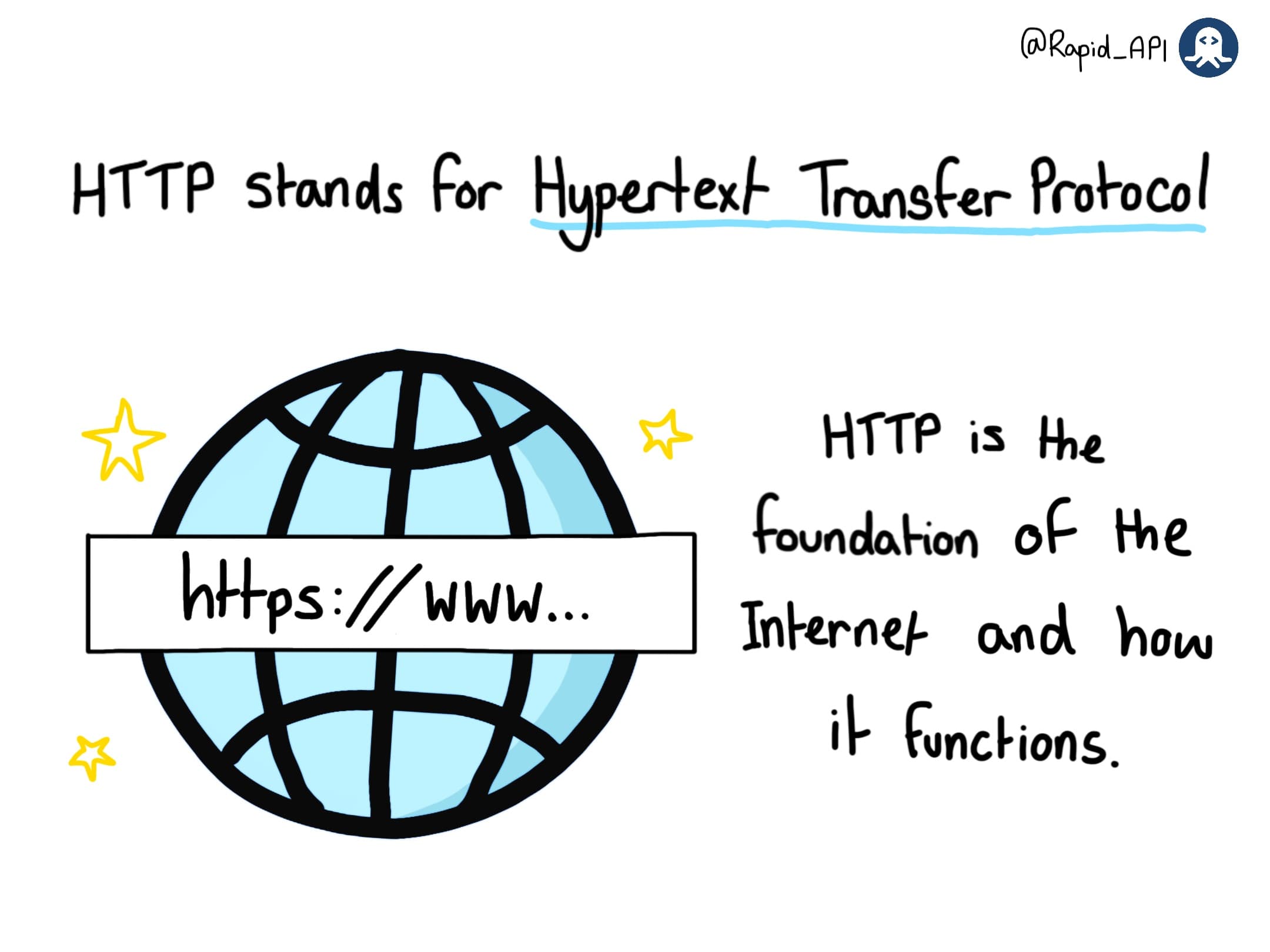
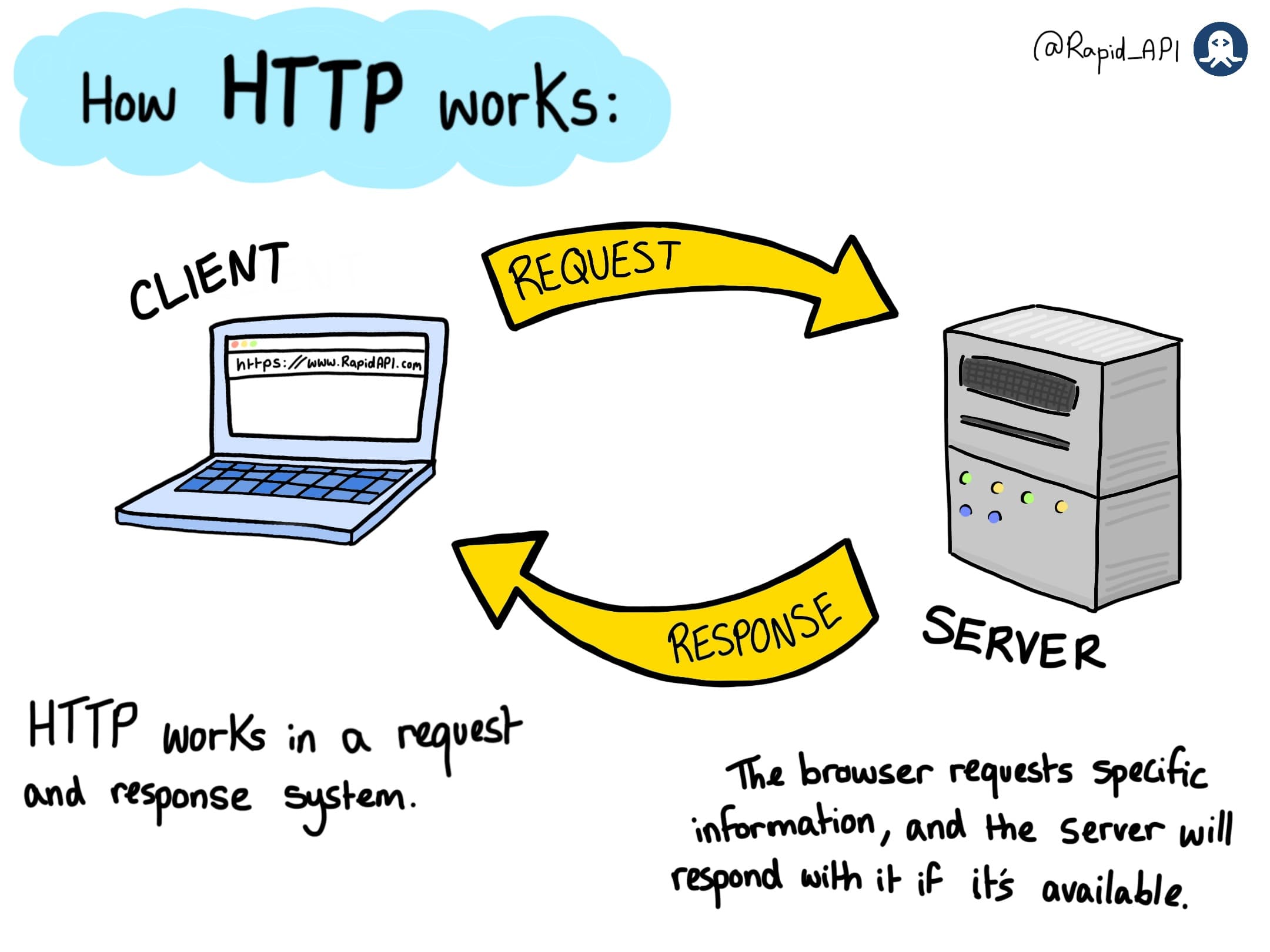
**What is HTTP?**

[](https://devrelstack.kinsta.cloud/wp-content/uploads/2022/07/image0-3-2.jpeg" \t "_blank)

[](https://devrelstack.kinsta.cloud/wp-content/uploads/2022/07/image1-1.jpeg" \t "_blank)

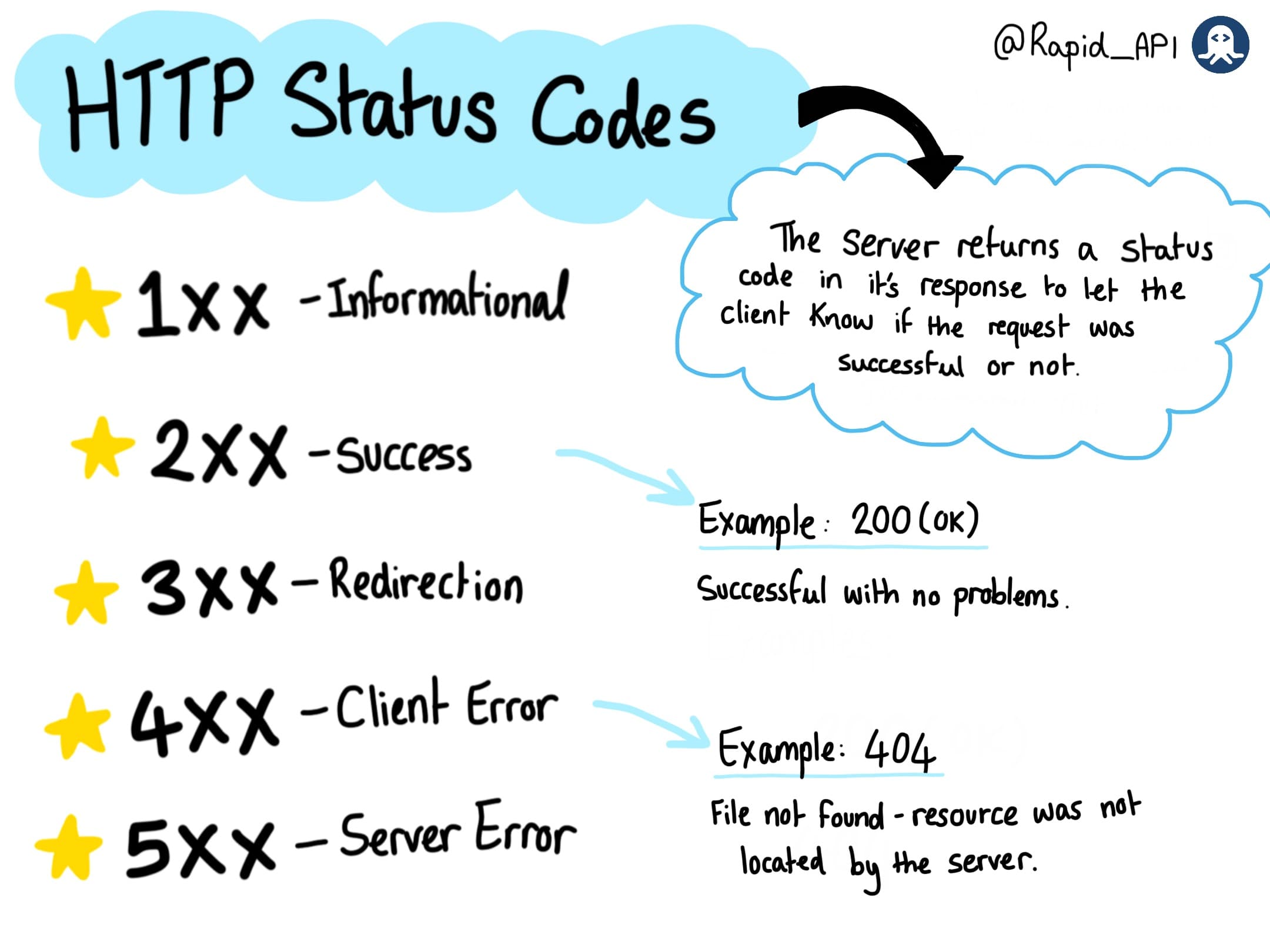
[What is HTTP?](https://devrelstack.kinsta.cloud/wp-content/uploads/2022/07/image1-1.jpeg" \t "_blank)

HTTP stands for Hypertext Transfer Protocol. It is the foundation of the Internet and how it functions and exchanges data.

[](https://devrelstack.kinsta.cloud/wp-content/uploads/2022/07/image2-2.jpeg" \t "_blank)

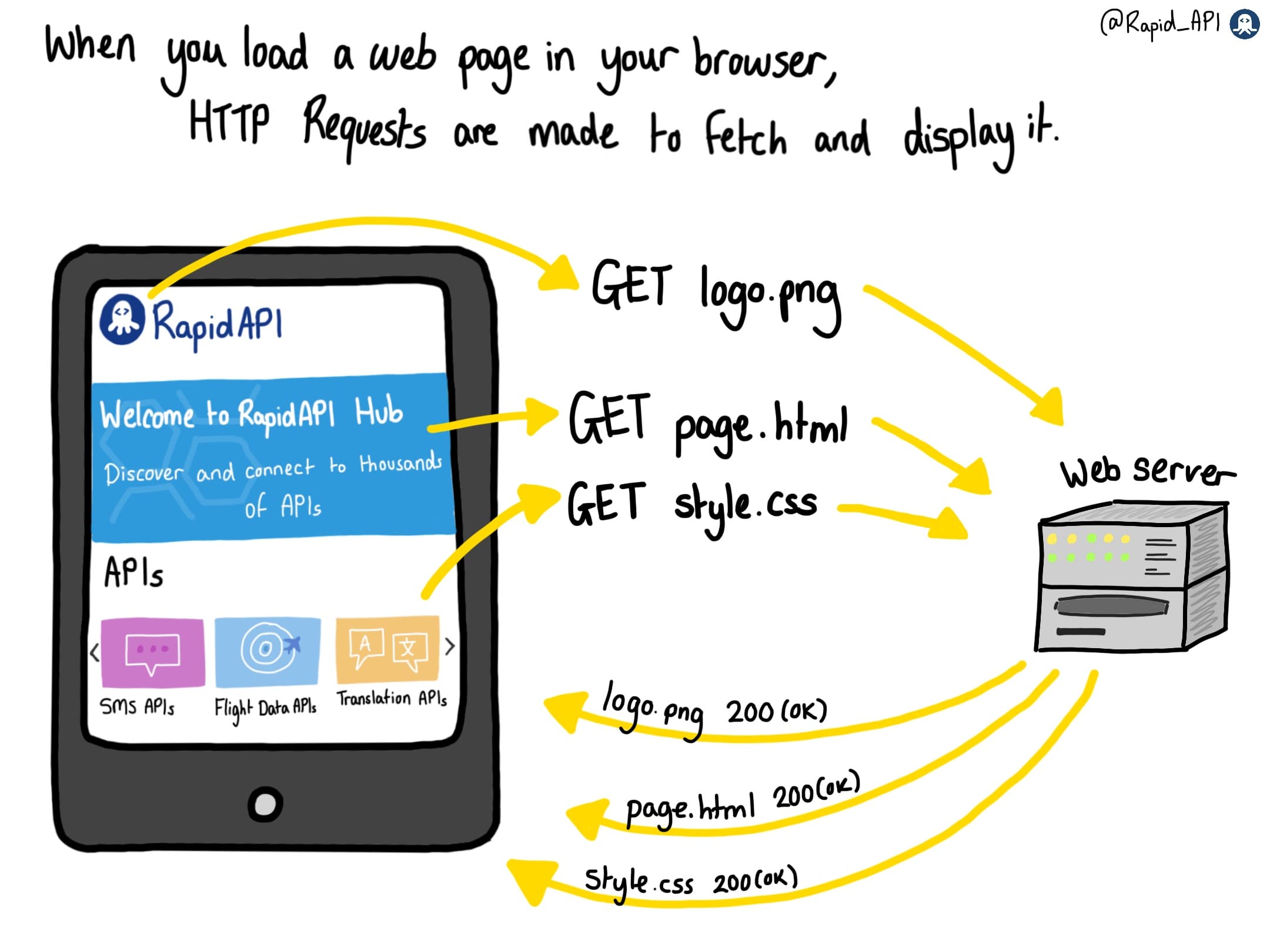
[What is HTTP?](https://devrelstack.kinsta.cloud/wp-content/uploads/2022/07/image2-2.jpeg" \t "_blank)

HTTP works in a Request and Response system. The client requests a specific resource from the server, and if the resource is available, it will be returned in a response from the server.

[](https://devrelstack.kinsta.cloud/wp-content/uploads/2022/07/image3-2.jpeg" \t "_blank)

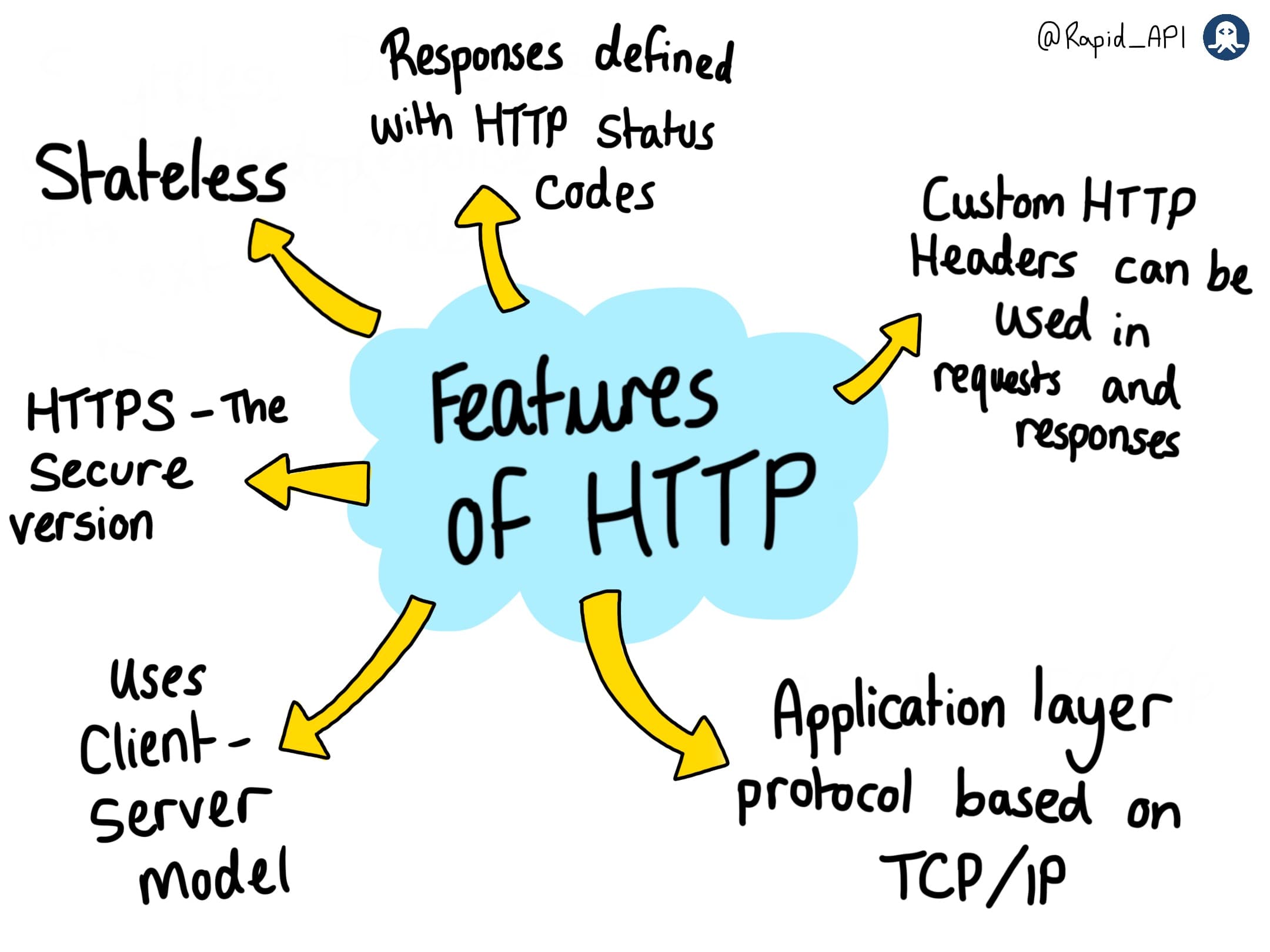
[What is HTTP?](https://devrelstack.kinsta.cloud/wp-content/uploads/2022/07/image3-2.jpeg" \t "_blank)

The response from the server will include an HTTP Status Code. This lets the client know the details of the response, whether it was successful or not, and why. Status codes are grouped into different numbers each indicating different response types. For example, codes 200-299 mean successful, and codes 300-399 indicate redirection.

[](https://devrelstack.kinsta.cloud/wp-content/uploads/2022/07/image4-2.jpeg" \t "_blank)

[What is HTTP?](https://devrelstack.kinsta.cloud/wp-content/uploads/2022/07/image4-2.jpeg" \t "_blank)

When you load a web page in your browser, the requests are sent to the corresponding server to fetch the various components of the page. Each request returns an individual response.

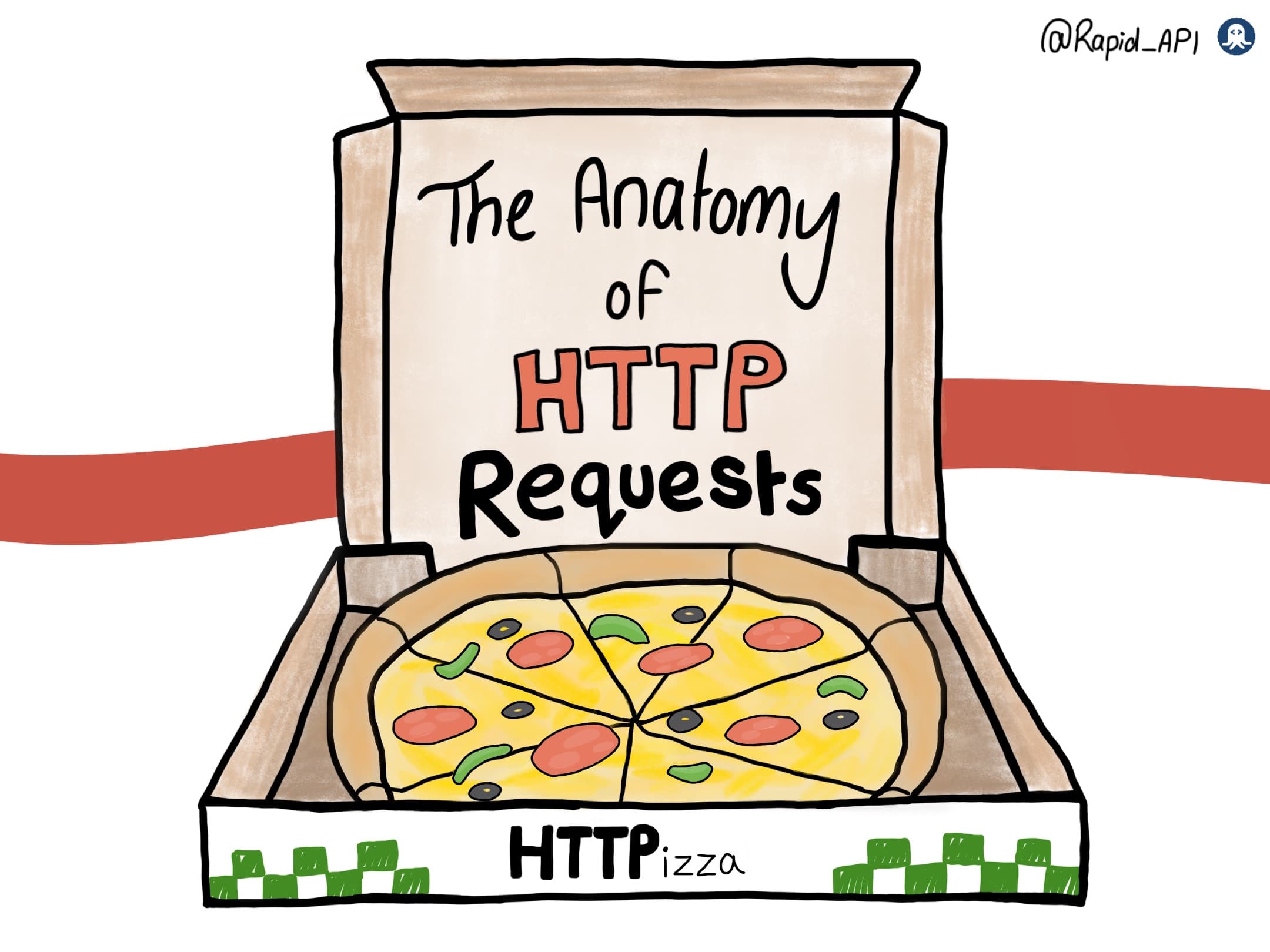
[](https://devrelstack.kinsta.cloud/wp-content/uploads/2022/07/image5-2.jpeg" \t "_blank)

[What is HTTP?](https://devrelstack.kinsta.cloud/wp-content/uploads/2022/07/image5-2.jpeg" \t "_blank)

HTTP has many features that make it the most widespread transfer protocol. HTTP is an application layer protocol based on TCP, a secure transport layer protocol. HTTP is also stateless, meaning each request and response cycle is separate from the ones before and after.

HTTP also enables the use of HTTP Headers in requests and responses, which allow them to be customized and provide extra information. HTTP also has a secure version - HTTPS- the standard network connection on the Internet.

**THE ANATOMY OF HTTP REQUESTS**

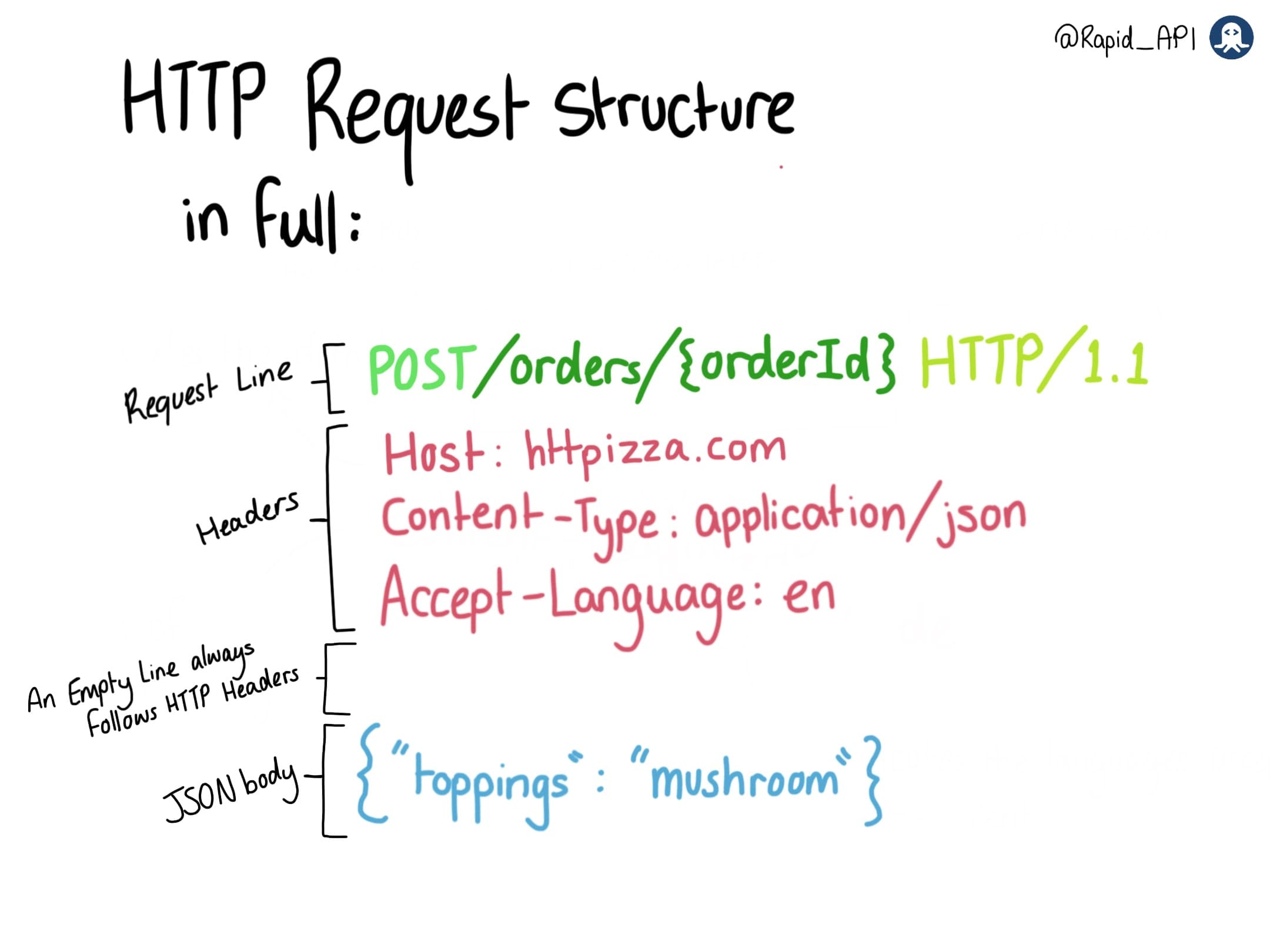
[](https://devrelstack.kinsta.cloud/wp-content/uploads/2022/07/image0-3-1.jpeg" \t "_blank)

HTTP Requests typically contain three things: a request line, header fields, and a body(This one is optional).

The request line is composed of three parts. Firstly, the HTTP Method, such as GET, POST, or DELETE. Secondly, the resource URL. This locates the requested resource on the server. Thirdly, the current HTTP version. The request line is like the foundation of a request, like a pizza base.

The second layer to our pizza is HTTP Header fields. These provide extra information to the server about the request. There are many different headers—for example, Content-type, which indicates the data format. Content-length indicates the length of the data in bytes. Accept-Language indicates the language of the content accepted by the client.

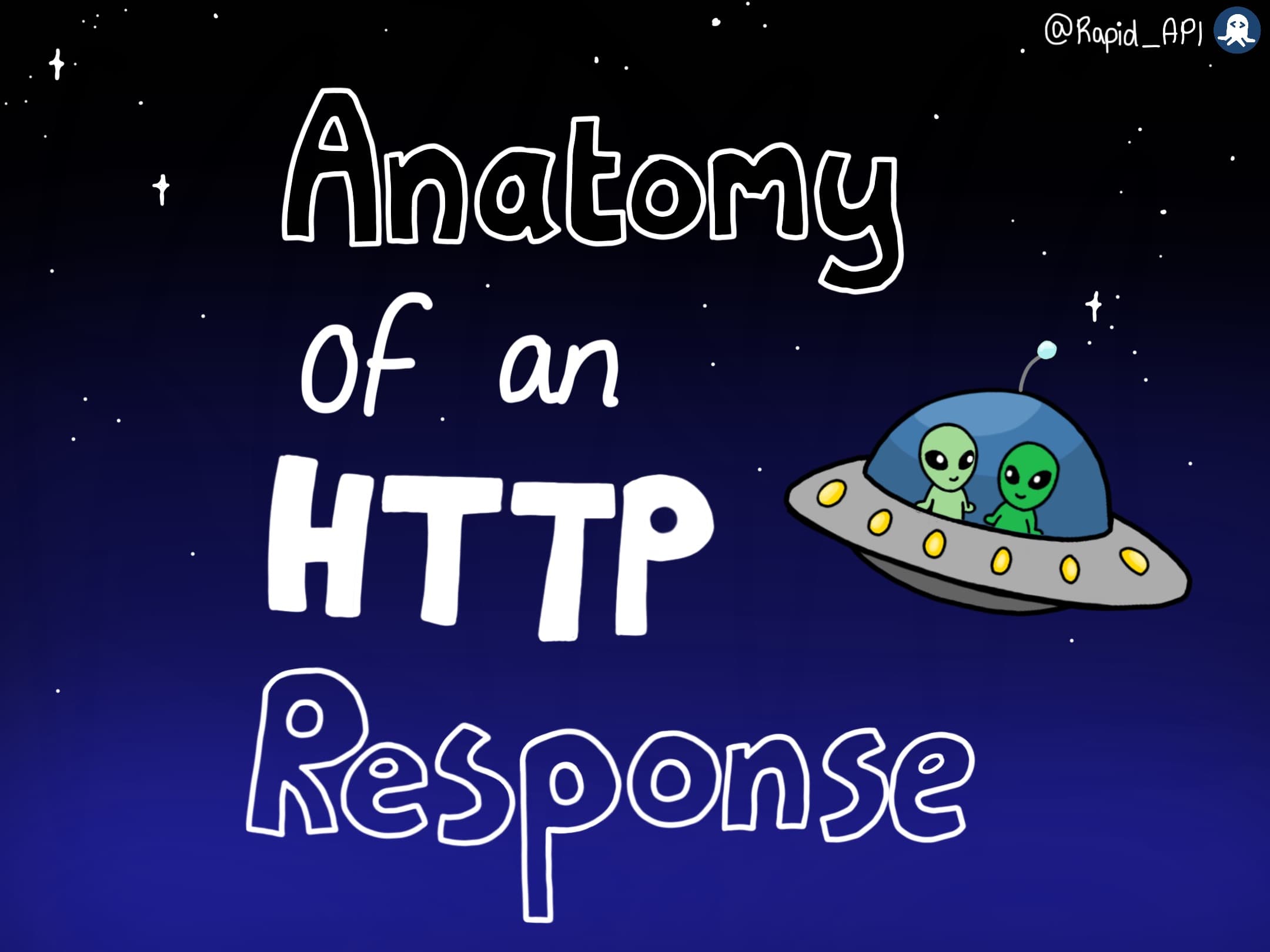
The last part of the request is the body(data). This part is usually only needed if the HTTP Method is POST, PUT, or PATCH because it contains the new data being sent to the server. For example, in a POST request, the body could contain the details of a pizza order form.

[](https://devrelstack.kinsta.cloud/wp-content/uploads/2022/07/image5-1.jpeg" \t "_blank)

Let's review the structure of an HTTP request in full. First, there's a request line that contains the HTTP Method, resource URL, and current HTTP version. Secondly, the HTTP Header fields provide extra information about the request. If the request is GET, this is all the information needed. However, if the request is POST, PUT, or PATCH, there is a body containing a message. An empty line always separates the Headers and the body.

That's the anatomy of an HTTP Request! It is now ready to be sent to the server.

**Anatomy of an HTTP Response**

[](https://devrelstack.kinsta.cloud/wp-content/uploads/2022/10/image0-8.jpeg" \t "_blank)

The Anatomy of an HTTP Response.

HTTP responses are made by a server in response to a client's request. They inform the client about whether the request was successful; if not, they provide error information. They also deliver resources to the client if that's what is requested. There are three components to an HTTP response.

The first part is the status line. This displays the HTTP version in use, followed by an informative status code indicating how the server handled the request. For example, a '200 OK'code means the request was successful, '404' indicates the requested resource was not found, etc.

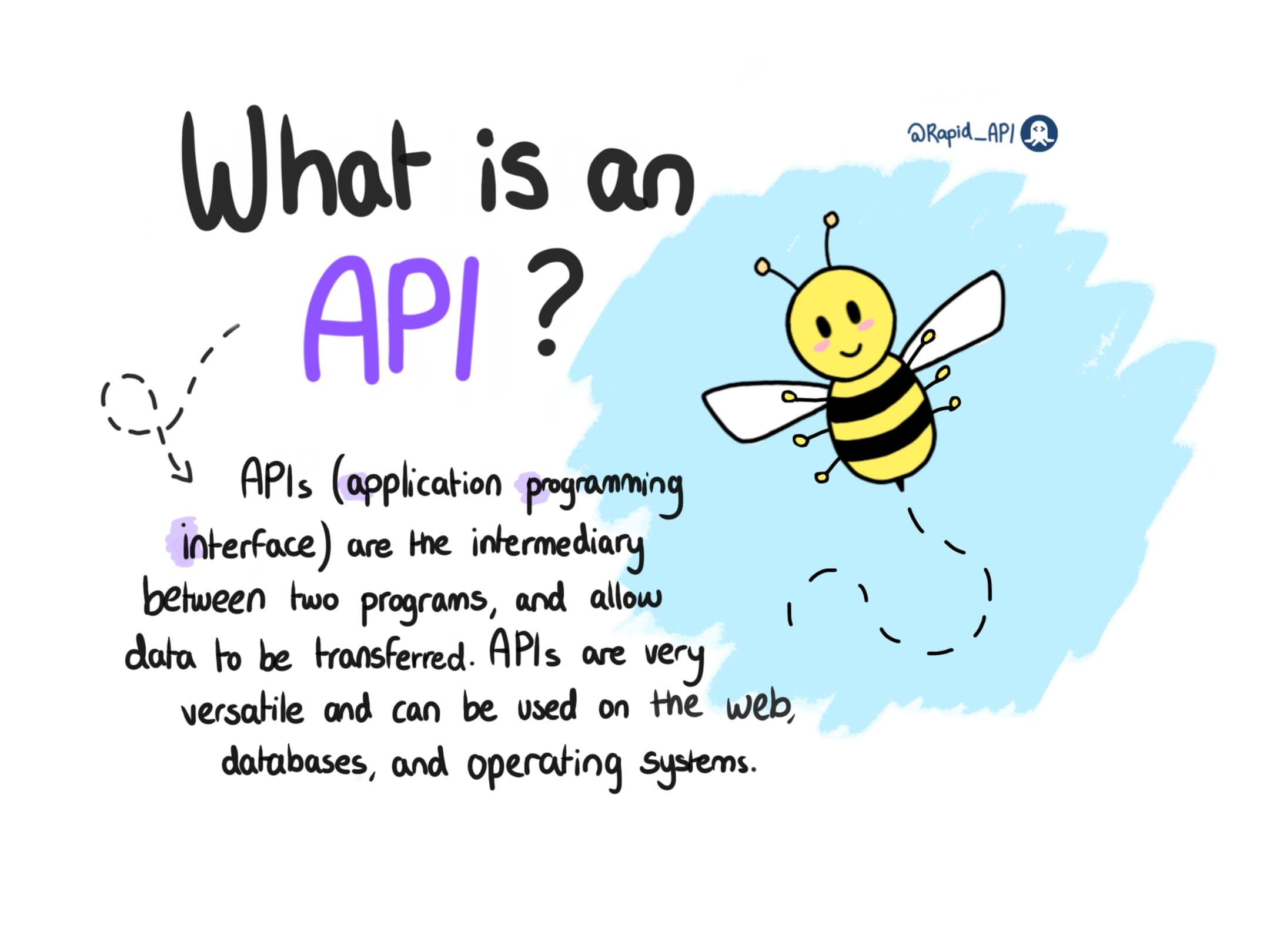
The second part of a response is the HTTP response headers. There are various headers to provide extra information about a response. There are headers for security, content, caching, date and time details, and more.

The last part of a response is the body(data). This part is optional, depending on the type of request sent. The body contains the data that is being sent back to the client. This could be an image, text, JSON, etc. In the response structure, there is always a blank line that precedes the body.

An HTTP response fully constructed with its's three parts: The status, headers, and body.

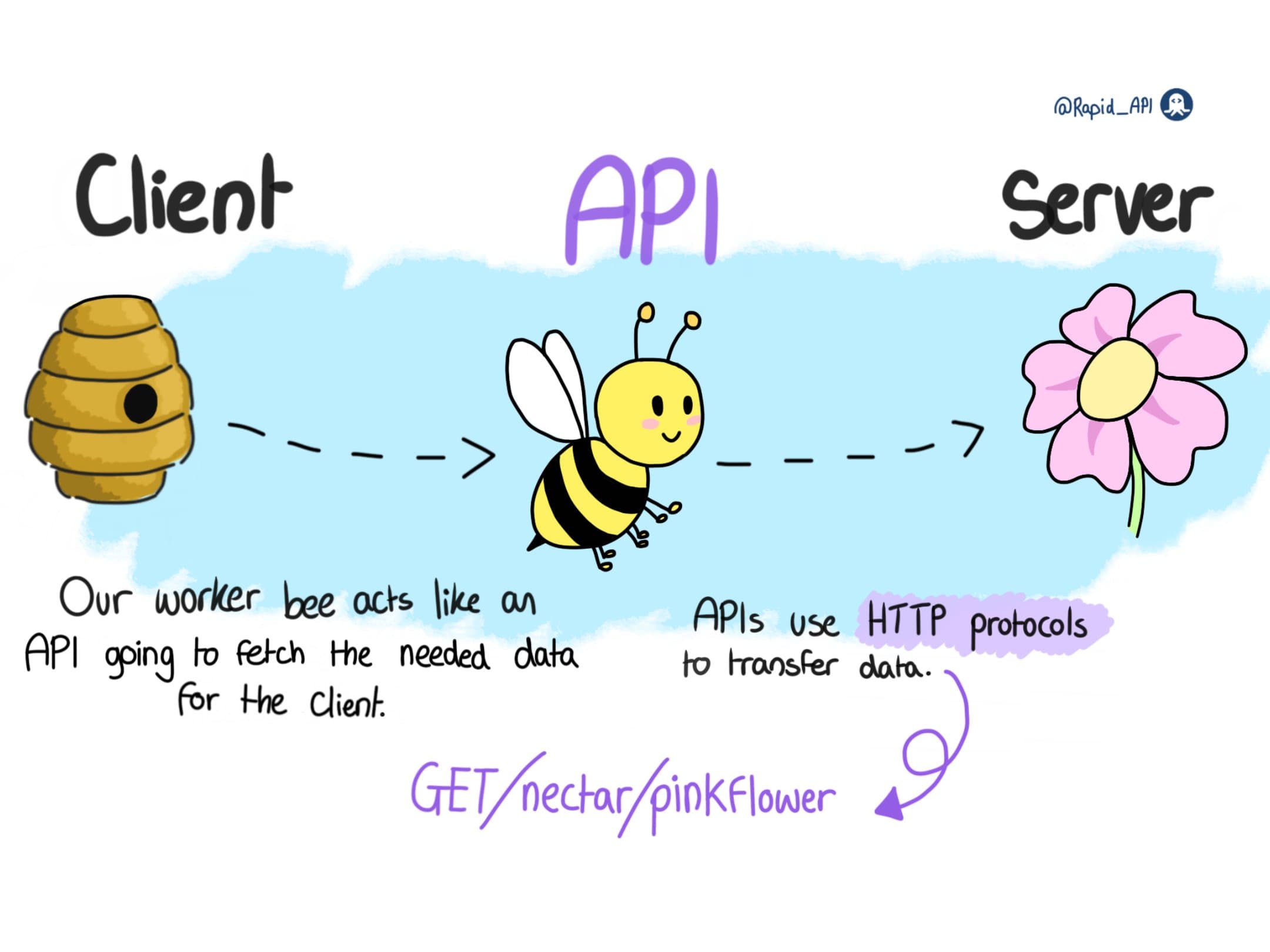
The server sends the response to the client as a reply to its request.

**RapidAPI Comic on API**

[](https://devrelstack.kinsta.cloud/wp-content/uploads/2022/05/api-cover.jpeg" \t "_blank)

What is an API RapidAPI Comic cover.

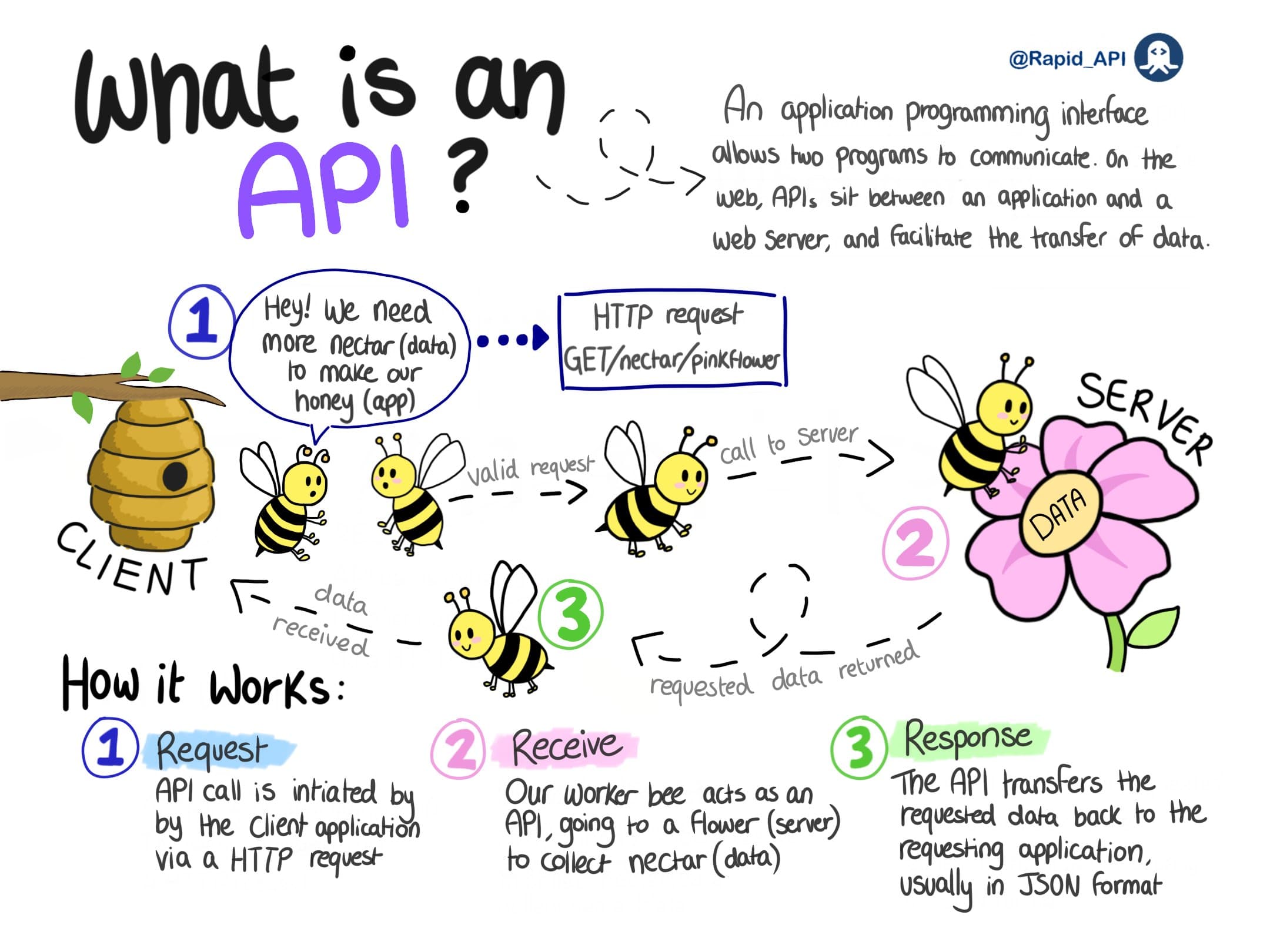
What is an API request? The client initiates the request to the server in order to fetch the required resource.

[](https://devrelstack.kinsta.cloud/wp-content/uploads/2022/05/api-3.jpeg" \t "_blank)

API basically establishes the connection between client and server.

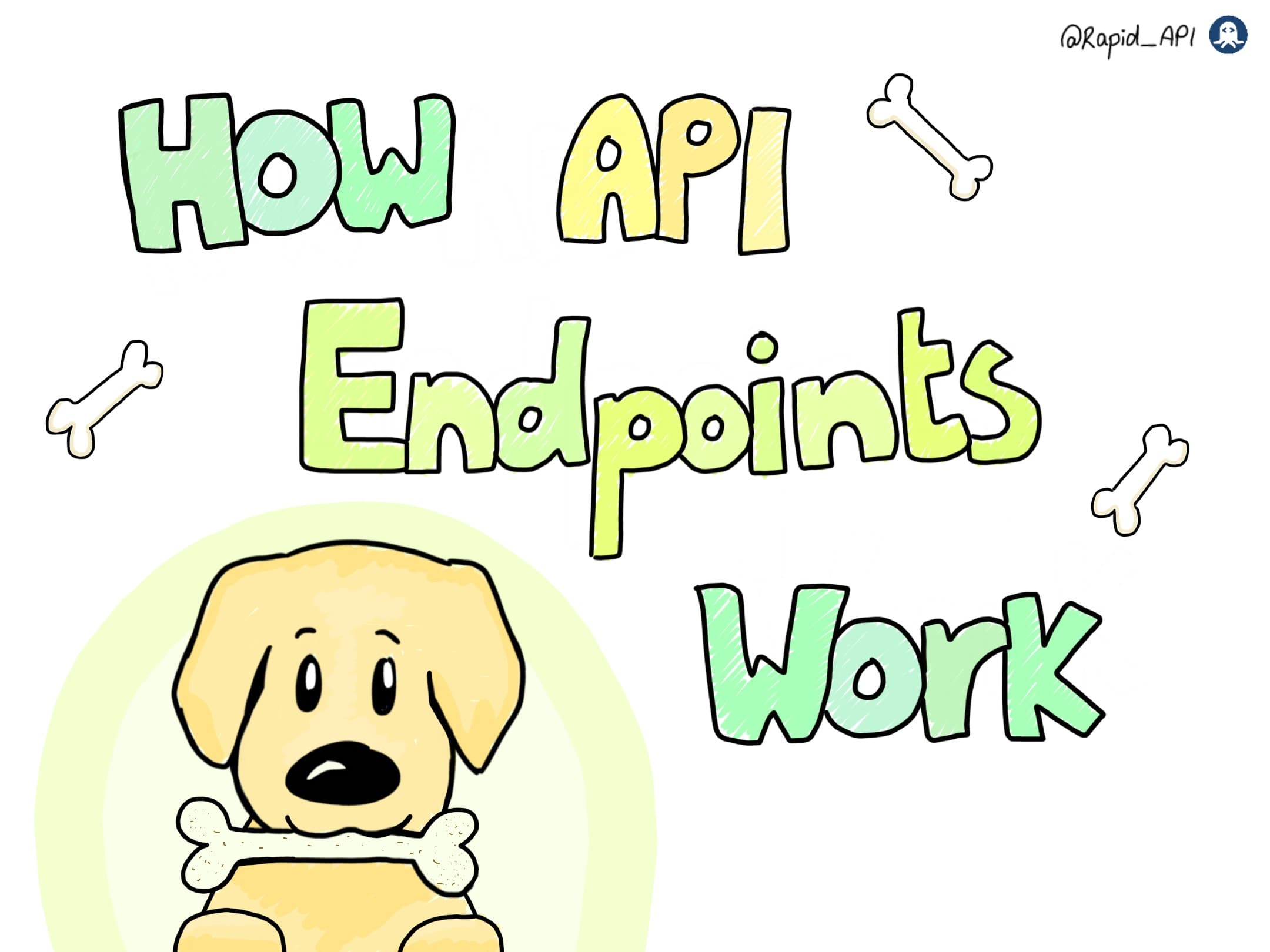
APIs have endpoints where clients can find the required data.

The server responds to the client by sending back the response with the desired resource.



"What is an API" explained.

**HOW API ENDPOINTS WORK?**

[](https://devrelstack.kinsta.cloud/wp-content/uploads/2022/06/1-cover-1.jpg" \t "_blank)

RapidAPI Comic cover: How API endpoints work.

What is an API endpoint? Endpoints are the communication touchpoint between API and server. They are URLs that users interact with to access specific resources. They typically look like this:

**pets/dog/snacks/bone**

Endpoints dictate a resource's location and are where the request is sent. Each endpoint locates a different resource. Image shows a dog being commanded to fetch the resource for dog/snacks/bone and then finding the bone.

Its critical endpoints are named after the entity they represent to avoid ambiguity around naming and errors. Image shows the dog knowing the location of various snacks that are appropriately named, for example, dog/snacks/steak, dog/snacks/chicken, dog/snacks/fish.

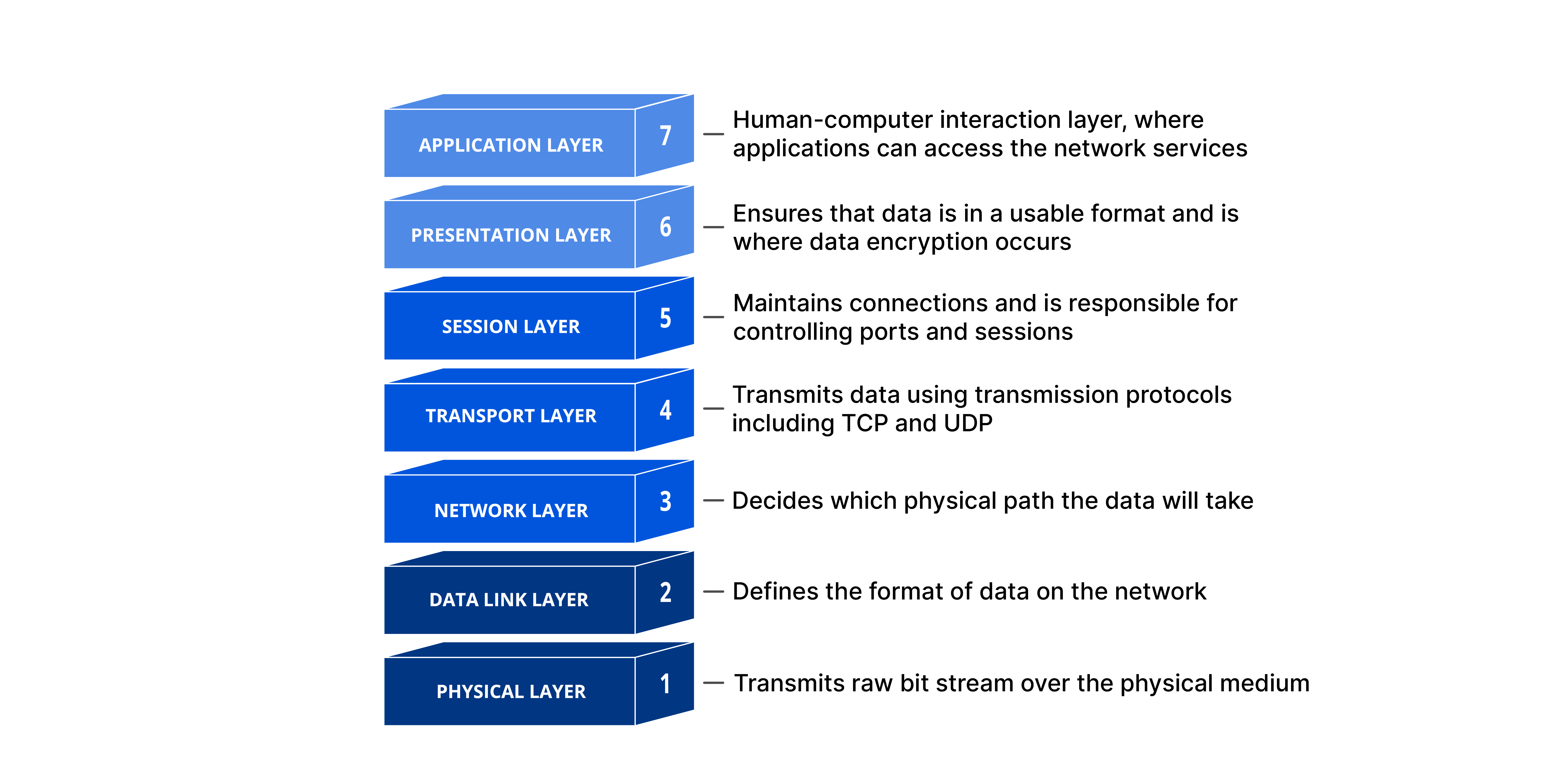
If endpoints are inaccurately named, the API will not be able to locate the resource or may find the wrong one. Image shows the dog attempting to locate the fish, but he finds a cat instead.

You can secure endpoints using HTTPS only, input validation, rate-limiting, and one-way password hashing. Image shows the dog wearing an HTTPS collar.

**WHAT IS THE OSI MODEL?**

The open systems interconnection (OSI) model is a conceptual model created by the International Organization for Standardization which enables diverse communication systems to communicate using standard [protocols](https://www.cloudflare.com/learning/network-layer/what-is-a-protocol/). In plain English, the OSI provides a standard for different computer systems to be able to communicate with each other.

The OSI Model can be seen as a universal language for computer networking. It is based on the concept of splitting up a communication system into seven abstract layers, each one stacked upon the last.



Each layer of the OSI Model handles a specific job and communicates with the layers above and below itself. [DDoS attacks](https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/) target specific layers of a network connection; [application layer attacks](https://www.cloudflare.com/learning/ddos/application-layer-ddos-attack/) target [layer 7](https://www.cloudflare.com/learning/ddos/what-is-layer-7/) and protocol layer attacks target layers 3 and 4.

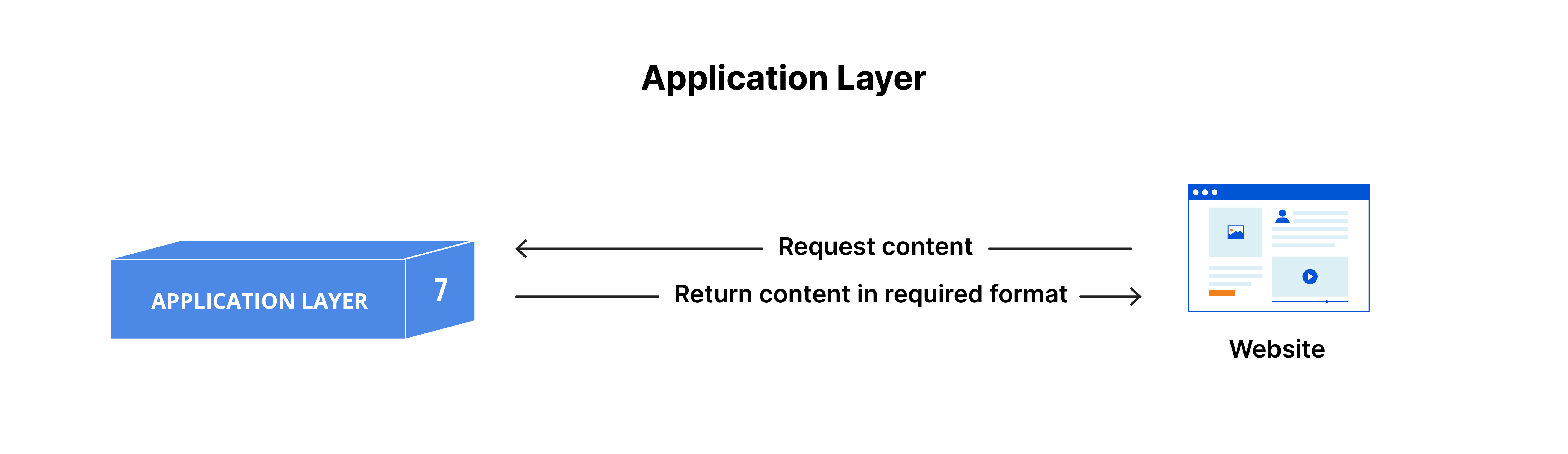
**Why does the OSI model matter?**

Although the modern Internet does not strictly follow the OSI Model (it more closely follows the simpler Internet protocol suite), the OSI Model is still very useful for troubleshooting network problems. Whether it’s one person who can’t get their laptop on the Internet, or a website being down for thousands of users, the OSI Model can help to break down the problem and isolate the source of the trouble. If the problem can be narrowed down to one specific layer of the model, a lot of unnecessary work can be avoided.

**What are the 7 layers of the OSI Model?**

The seven abstraction layers of the OSI model can be defined as follows, from top to bottom:

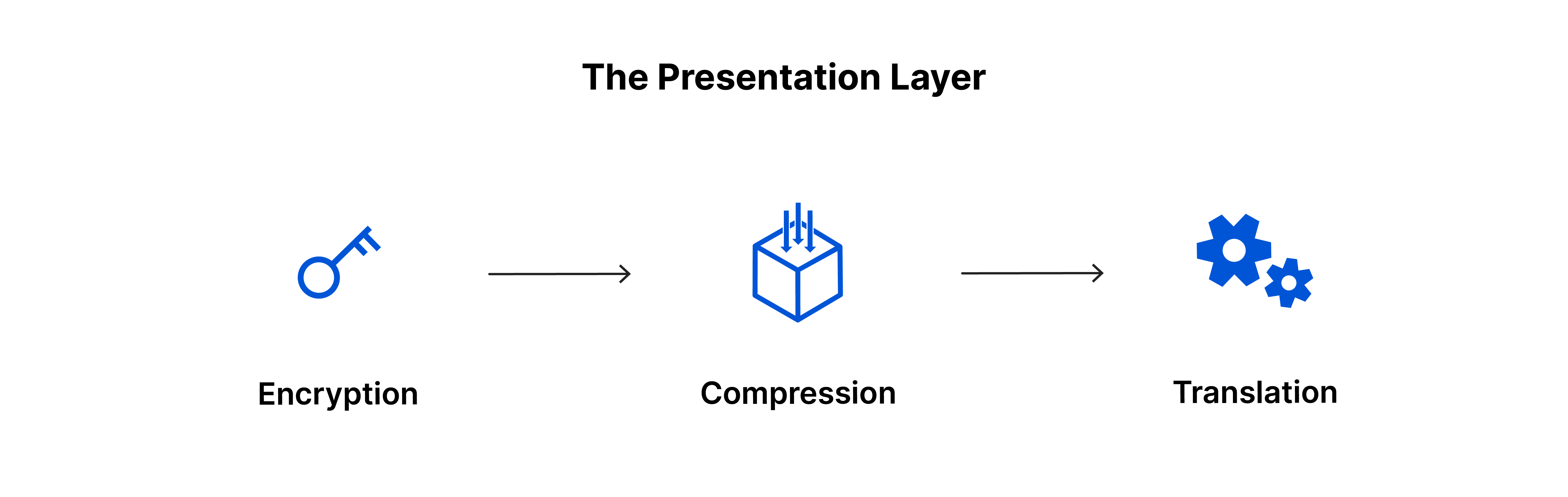
**7. The application layer**



This is the only layer that directly interacts with data from the user. Software applications like web browsers and email clients rely on the application layer to initiate communications. But it should be made clear that client software applications are not part of the application layer; rather the application layer is responsible for the protocols and data manipulation that the software relies on to present meaningful data to the user.

Application layer protocols include [HTTP](https://www.cloudflare.com/learning/ddos/glossary/hypertext-transfer-protocol-http/) as well as [SMTP](https://www.cloudflare.com/learning/email-security/what-is-smtp/) (Simple Mail Transfer Protocol is one of the protocols that enables [email](https://www.cloudflare.com/learning/email-security/what-is-email/) communications).

**6. The presentation layer**



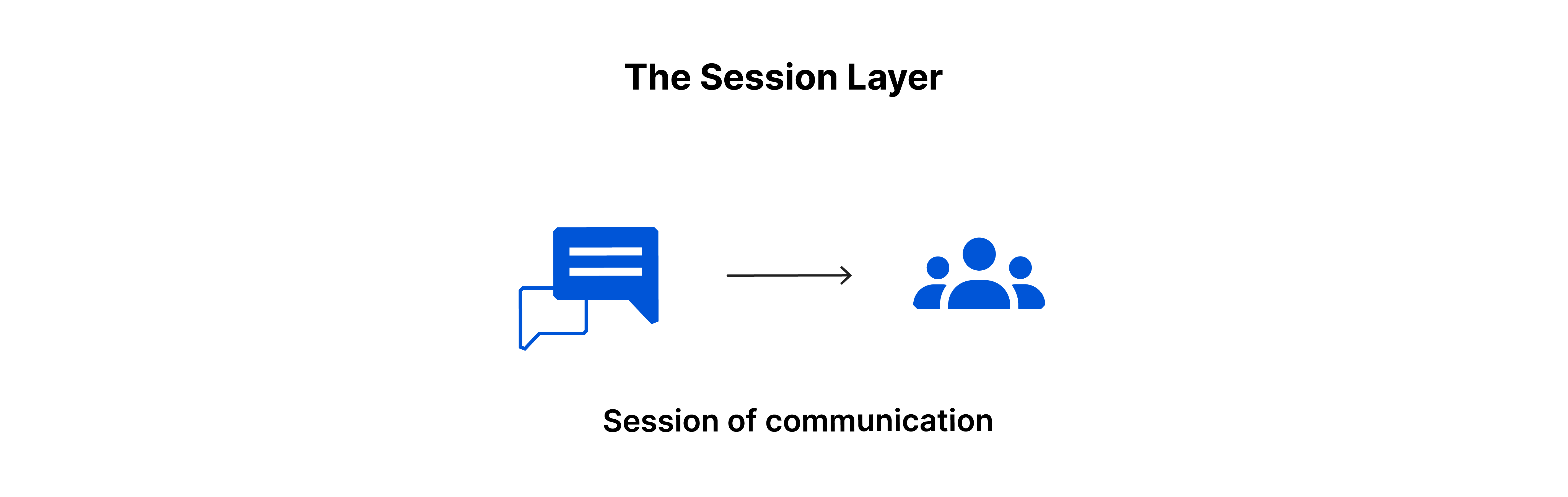
This layer is primarily responsible for preparing data so that it can be used by the application layer; in other words, layer 6 makes the data presentable for applications to consume. The presentation layer is responsible for translation, [encryption](https://www.cloudflare.com/learning/ssl/what-is-encryption/), and compression of data.

Two communicating devices communicating may be using different encoding methods, so layer 6 is responsible for translating incoming data into a syntax that the application layer of the receiving device can understand.

If the devices are communicating over an encrypted connection, layer 6 is responsible for adding the encryption on the sender’s end as well as decoding the encryption on the receiver's end so that it can present the application layer with unencrypted, readable data.

Finally the presentation layer is also responsible for compressing data it receives from the application layer before delivering it to layer 5. This helps improve the speed and efficiency of communication by minimizing the amount of data that will be transferred.

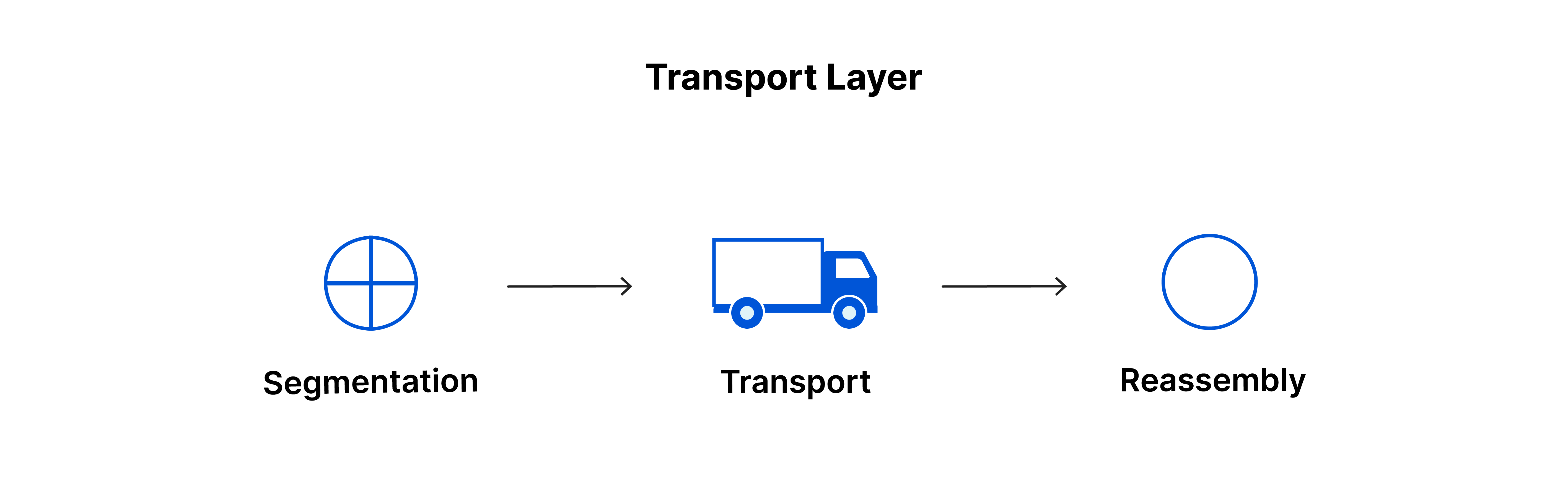
**5. The session layer**



This is the layer responsible for opening and closing communication between the two devices. The time between when the communication is opened and closed is known as the session. The session layer ensures that the session stays open long enough to transfer all the data being exchanged, and then promptly closes the session in order to avoid wasting resources.

The session layer also synchronizes data transfer with checkpoints. For example, if a 100 megabyte file is being transferred, the session layer could set a checkpoint every 5 megabytes. In the case of a disconnect or a crash after 52 megabytes have been transferred, the session could be resumed from the last checkpoint, meaning only 50 more megabytes of data need to be transferred. Without the checkpoints, the entire transfer would have to begin again from scratch.

**4. The transport layer**

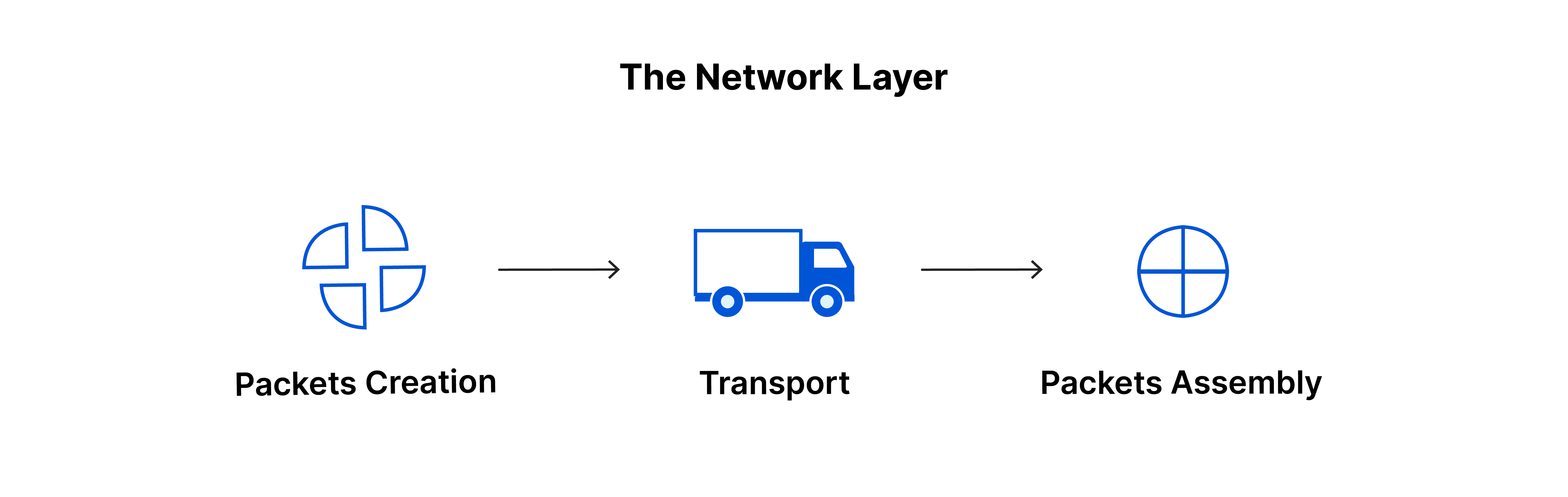


Layer 4 is responsible for end-to-end communication between the two devices. This includes taking data from the session layer and breaking it up into chunks called segments before sending it to layer 3. The transport layer on the receiving device is responsible for reassembling the segments into data the session layer can consume.

The transport layer is also responsible for flow control and error control. Flow control determines an optimal speed of transmission to ensure that a sender with a fast connection does not overwhelm a receiver with a slow connection. The transport layer performs error control on the receiving end by ensuring that the data received is complete, and requesting a retransmission if it isn’t.

Transport layer protocols include the [Transmission Control Protocol (TCP)](https://www.cloudflare.com/learning/ddos/glossary/tcp-ip/) and the [User Datagram Protocol (UDP)](https://www.cloudflare.com/learning/ddos/glossary/user-datagram-protocol-udp/).

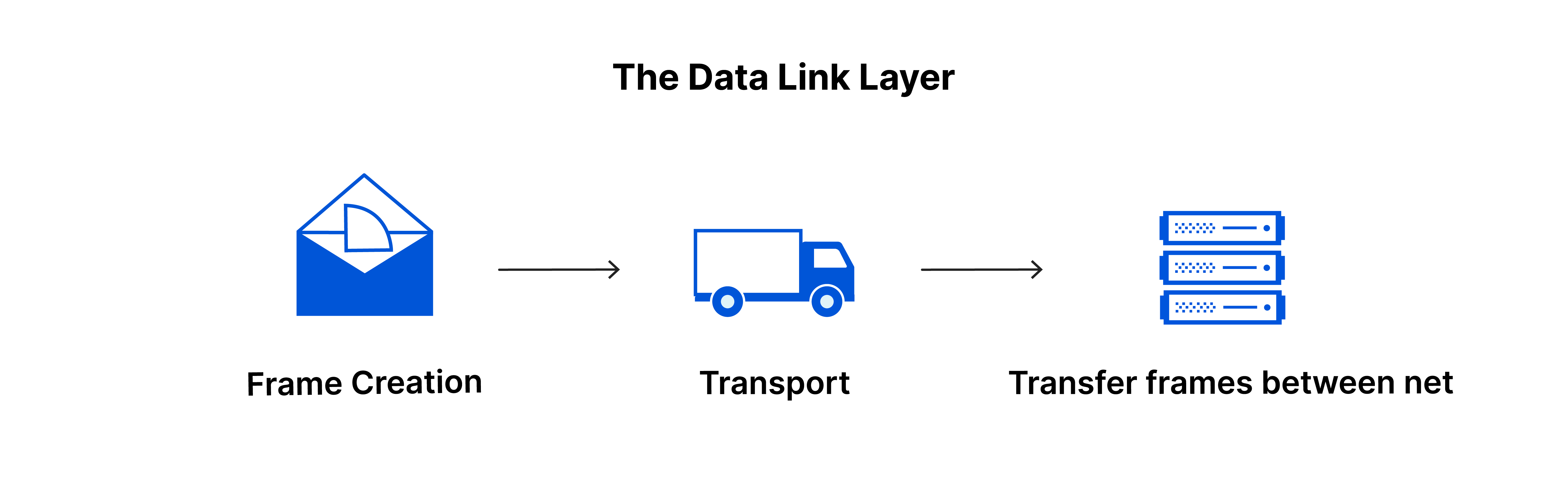
**3. The network layer**



The [network layer](https://www.cloudflare.com/learning/network-layer/what-is-the-network-layer/) is responsible for facilitating data transfer between two different networks. If the two devices communicating are on the same network, then the network layer is unnecessary. The network layer breaks up segments from the transport layer into smaller units, called [packets](https://www.cloudflare.com/learning/network-layer/what-is-a-packet/), on the sender’s device, and reassembling these packets on the receiving device. The network layer also finds the best physical path for the data to reach its destination; this is known as [routing](https://www.cloudflare.com/learning/network-layer/what-is-routing/).

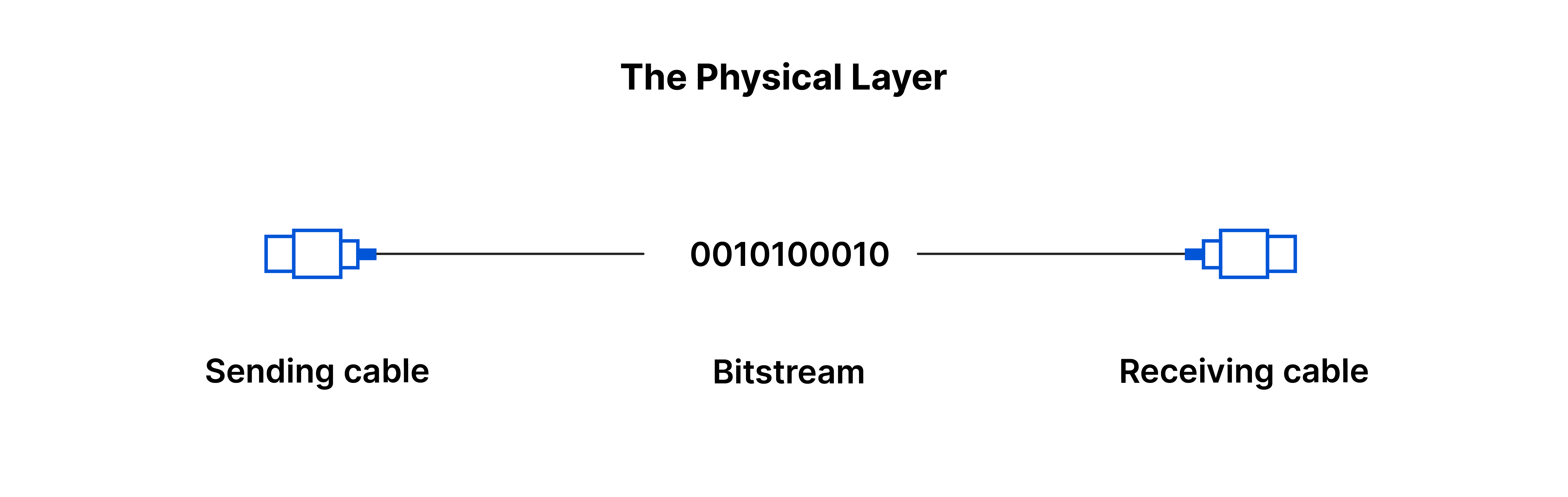
Network layer protocols include IP, the [Internet Control Message Protocol (ICMP)](https://www.cloudflare.com/learning/ddos/glossary/internet-control-message-protocol-icmp/), the [Internet Group Message Protocol (IGMP)](https://www.cloudflare.com/learning/network-layer/what-is-igmp/), and the [IPsec](https://www.cloudflare.com/learning/network-layer/what-is-ipsec/) suite.

**2. The data link layer**



The data link layer is very similar to the network layer, except the data link layer facilitates data transfer between two devices on the *same* network. The data link layer takes packets from the network layer and breaks them into smaller pieces called frames. Like the network layer, the data link layer is also responsible for flow control and error control in intra-network communication (The transport layer only does flow control and error control for inter-network communications).

**1. The physical layer**



This layer includes the physical equipment involved in the data transfer, such as the cables and [switches](https://www.cloudflare.com/learning/network-layer/what-is-a-network-switch/). This is also the layer where the data gets converted into a bit stream, which is a string of 1s and 0s. The physical layer of both devices must also agree on a signal convention so that the 1s can be distinguished from the 0s on both devices.

**How data flows through the OSI Model**

In order for human-readable information to be transferred over a network from one device to another, the data must travel down the seven layers of the OSI Model on the sending device and then travel up the seven layers on the receiving end.

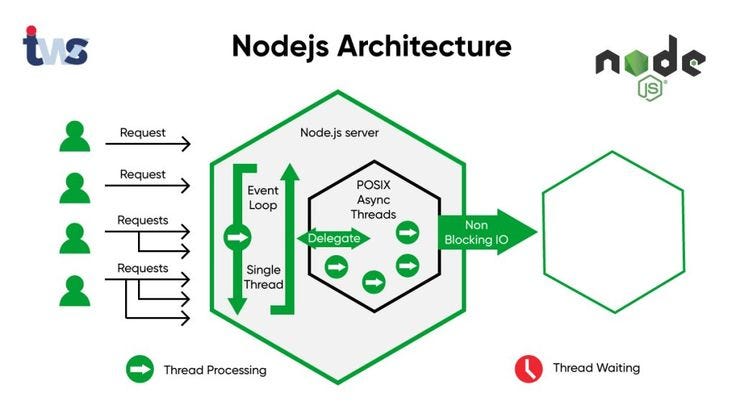
For example: Mr. Cooper wants to send Ms. Palmer an email. Mr. Cooper composes his message in an email application on his laptop and then hits ‘send’. His email application will pass his email message over to the application layer, which will pick a protocol (SMTP) and pass the data along to the presentation layer. The presentation layer will then compress the data and then it will hit the session layer, which will initialize the communication session.

The data will then hit the sender’s transportation layer where it will be segmented, then those segments will be broken up into packets at the network layer, which will be broken down even further into frames at the data link layer. The data link layer will then deliver those frames to the physical layer, which will convert the data into a bitstream of 1s and 0s and send it through a physical medium, such as a cable.

Once Ms. Palmer’s computer receives the bit stream through a physical medium (such as her wifi), the data will flow through the same series of layers on her device, but in the opposite order. First the physical layer will convert the bitstream from 1s and 0s into frames that get passed to the data link layer. The data link layer will then reassemble the frames into packets for the network layer. The network layer will then make segments out of the packets for the transport layer, which will reassemble the segments into one piece of data.

The data will then flow into the receiver's session layer, which will pass the data along to the presentation layer and then end the communication session. The presentation layer will then remove the compression and pass the raw data up to the application layer. The application layer will then feed the human-readable data along to Ms. Palmer’s email software, which will allow her to read Mr. Cooper’s email on her laptop screen.

**THE NODE.JS ARCHITECTURE**



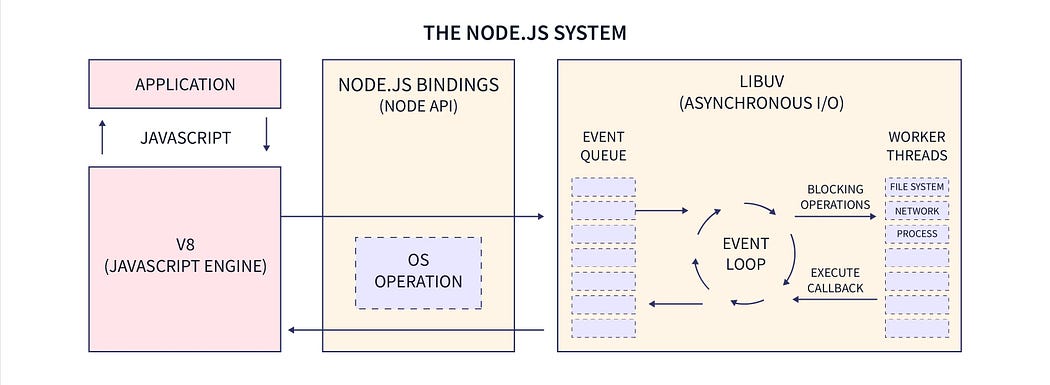
**Introduction:**

Node.js is an extremely powerful JavaScript-based platform that’s built on Google Chrome’s JavaScript V8 Engine, used to develop I/O intensive web applications like video streaming sites, single-page applications, online chat applications, and other web apps.

Node.js is used by large, established companies and newly-minted startups alike. Open-source and completely free, the platform is used by thousands of developers around the world. It brings plenty of advantages to the table, making it a better choice than other server-side platforms like Java or PHP in many cases.

Node.js has its core part written in C and C++. Node js is based on a single-threaded event loop architecture which allows Node to handle multiple client requests. Node js uses the concept of an asynchronous model and non-blocking I/O. We will look at these terms in detail.

Node.js is one of the best options for other server-side platforms because of the architectural pattern it follows.



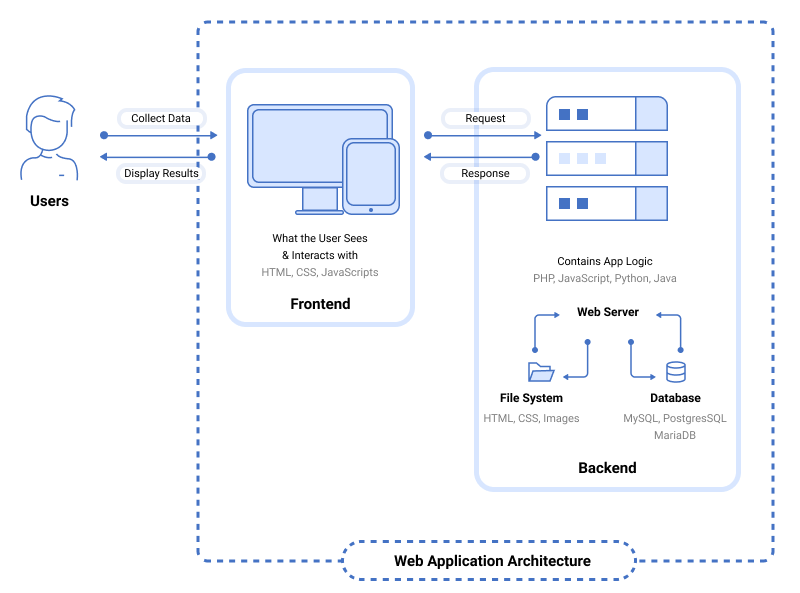
**Table of contents:**

* Basic components of web applications
* Node.js Architecture.
* Work flow of Node.js Architecture
* Advantages of Node.js Architecture

**Basic components of web applications:**

A web application, as you may already know, is a program that runs on a server and is rendered by a client browser, using the internet to access all the resources of that application. It usually can be easily broken down into three parts:

1. Client
2. Server
3. Database



Client:

The user interacts with the front-end part of a web application. The front-end is usually developed using languages like HTML and CSS styles, along with extensive usage of JavaScript-based frameworks like ReactJS and Angular, which help with application design.

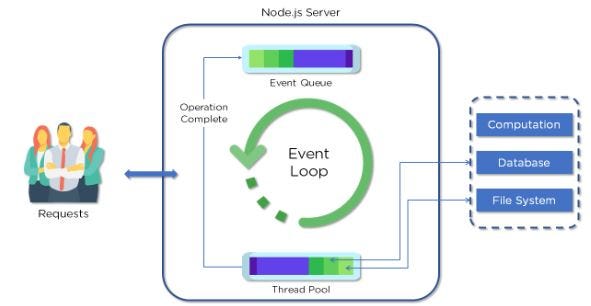
Server:

The server is responsible for taking the client requests, performing the required tasks, and sending responses back to the clients. It acts as a middleware between the front-end and stored data to enable operations on the data by a client. Node.js, PHP, and Java are the most popular technologies in use to develop and maintain a web server.

Database:

The database stores the data for a web application. The data can be created, updated, and deleted whenever the client requests. MySQL and MongoDB are among the most popular databases used to store data for web applications.

**Node.js Architecture:**



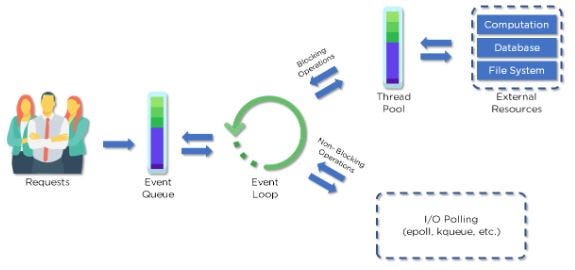
Node.js uses the “Single Threaded Event Loop” architecture to handle multiple concurrent clients. Node.js Processing Model is based on the JavaScript event-based model along with the JavaScript callback mechanism.

Now let’s understand each part of the Node.js architecture and the workflow of a web server developed using Node.js.

Parts of Node js Architecture:

* Requests : Incoming requests can be blocking (complex) or non-blocking (simple), depending upon the tasks that a user wants to perform in a web application
* Node.js server: Node.js server is a server-side platform that takes requests from users, processes those requests, and returns responses to the corresponding users
* Event Queue: Event Queue in a Node.js server stores incoming client requests and passes those requests one-by-one into the Event Loop
* Thread pool: Thread pool consists of all the threads available for carrying out some tasks that might be required to fulfill client requests
* Event loop: Event Loop indefinitely receives requests and processes them, and then returns the responses to corresponding clients.
* External Resources: External resources are required to deal with blocking client requests. These resources can be for computation, data storage, etc.

**The workflow of Node js Architecture:**



A web server developed using Node.js typically has a workflow that is quite similar to the diagram illustrated below. Let’s explore this flow of operations in detail.

* Clients send requests to the webserver to interact with the web application. Requests can be non-blocking or blocking e.g Querying the data , updating the data or deleting the data .
* Node.js retrieves the incoming requests and adds those requests to the Event Queue.
* The requests are then passed one-by-one through the Event Loop. It checks if the requests are simple enough to not require any external resources.
* Event Loop processes simple requests (non-blocking operations), such as I/O Polling, and returns the responses to the corresponding clients.

A single thread from the Thread Pool is assigned to a single complex request. This thread is responsible for completing a particular blocking request by accessing the external resources, such as compute, database, file system, etc.

Once, the task is carried out completely, the response is sent to the Event Loop that in turn sends that response back to the Client.

**Advantages of Node.js Architecture:**

Node.js Architecture comes with several advantages that give the server-side platform a distinct upper-hand when compared to other server-side languages:

* Handling multiple concurrent client requests is fast and easy: With the use of Event Queue and Thread Pool, the Node.js server enables efficient handling of a large number of incoming requests.
* No need for creating multiple threads: Event Loop handles all requests one-by-one, so there is no need to create multiple threads. Instead, a single thread is sufficient to handle a blocking incoming request.

All of these advantages contribute to making the servers developed using Node.js much faster and responsive when compared to those developed using other server development technologies.

In conclusion ,Node js Architecture is widely known because of its capabilities in solving real world problems with the use of technology. Node js is a very powerful tool for creating scalable web applications with variety of usage in various technological niche e.g E commerce , streaming applications etc. Thanks for Reading.