When to Use Webhooks, WebSocket, Pub/Sub, and Polling

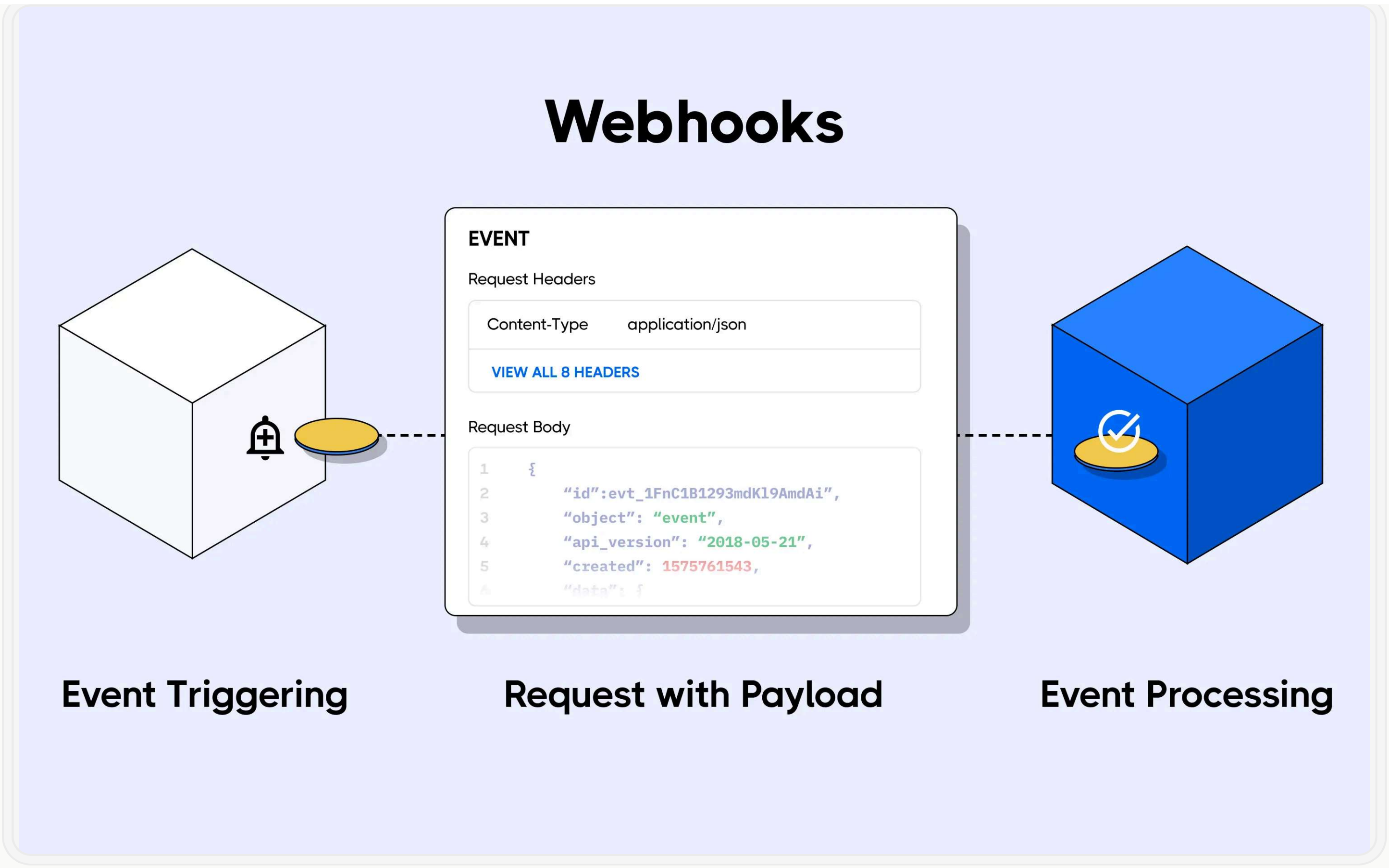
One of the simplest ways online applications share data is through the use of webhooks, a one-way communication format for moving data from one application to another. However, webhooks aren't the only method for data transfer between networked applications. As an engineer, it is very important to use the right tool for the right problem, or you could risk trivializing or overengineering the solution.

In this article, we will start by looking at how webhooks work. Then, I will compare the process with other data sharing techniques used by modern architectures in order to pair each technique with the scenarios where you should use them.

After the comparisons comes the answer to the question that might be heaviest on your mind: When should I be using webhooks?

How webhooks work

Webhook communication is achieved by sending an HTTP request from a source application to a destination application. When an event takes place in the source application, an HTTP request which might contain data relating to the event is triggered. This HTTP request is sent to the destination application's endpoint (often referred to as the webhook URL.)



```
{
    “id”:evt_1FnC1B1293mdKl9AmdAi”,
    “object”: “event”,
    “api_version”: “2018-05-21”,
    “created”: 1575761543,
    “data”: {
```

Webhook requests can be sent using the POST or GET request methods. This depends on the webhook provider's preferences — the information on how to consume the requests is always available on the provider's documentation.

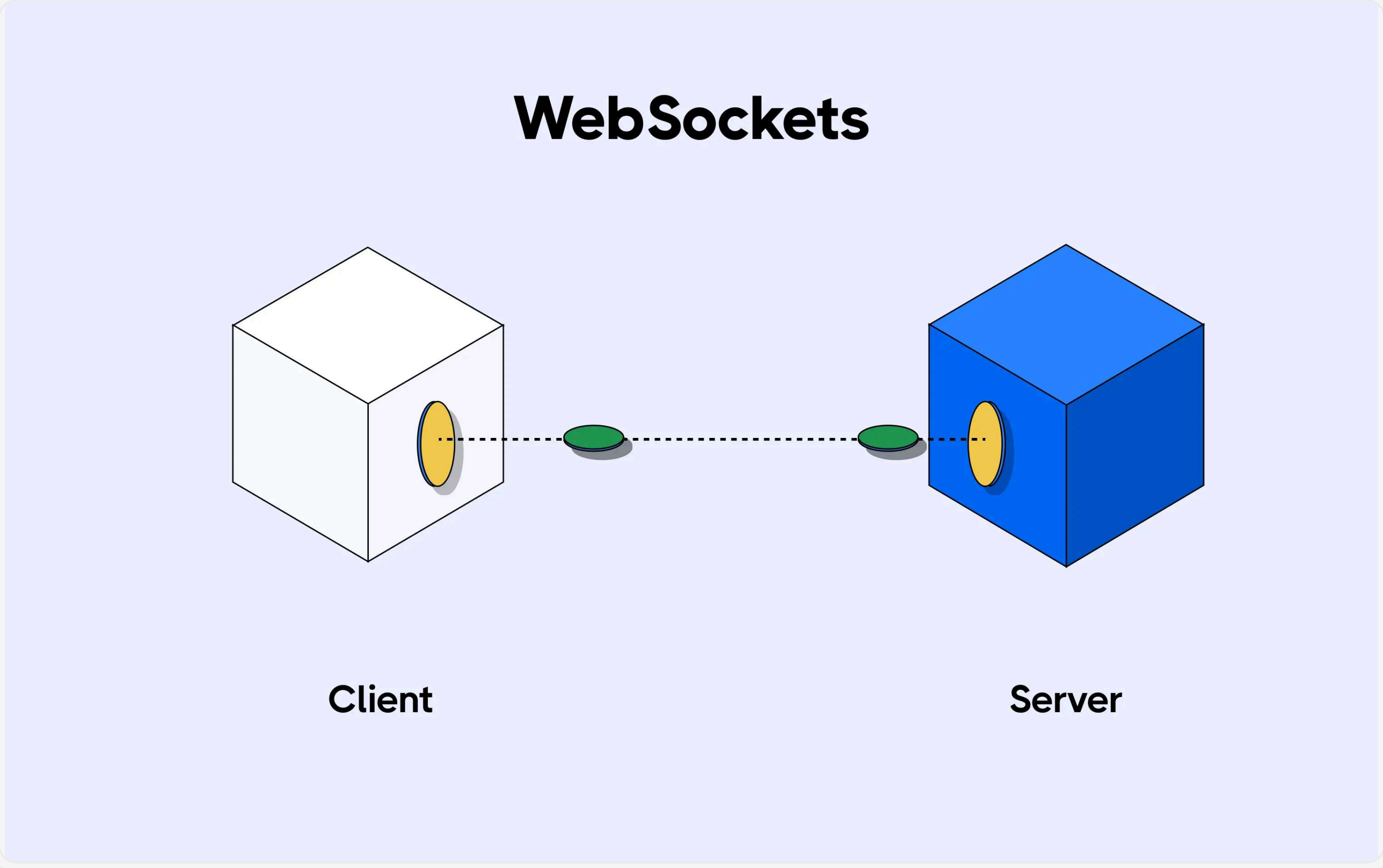
Now that we have a good understanding of how webhooks work, let's dive in to some comparisons.

Webhooks or WebSockets

How WebSockets work

WebSockets are used to facilitate two-way real-time communication between two networked systems. WebSockets achieve this by keeping a socket open on both the client and the server for the duration of the conversation.

This opens a duplex communication stream enabling both the client and server to pass information between one another with no significant latency added.



Differences between webhooks and WebSockets

Webhooks are used for one-way communication from a source application to a destination application, while WebSockets facilitate two-way communication between server and client.

Webhooks are mostly used by two servers to pass information, while WebSockets are used primarily for server-to-client (mostly web browsers) communication.

Also, webhooks close the socket connection on the receiving application once a response has been sent back, while WebSockets keep the connection open for as long as required and not just for a single transfer of information.

In terms of communication protocols, WebSocket uses its own custom `WS` protocol while webhooks use regular `HTTP`.

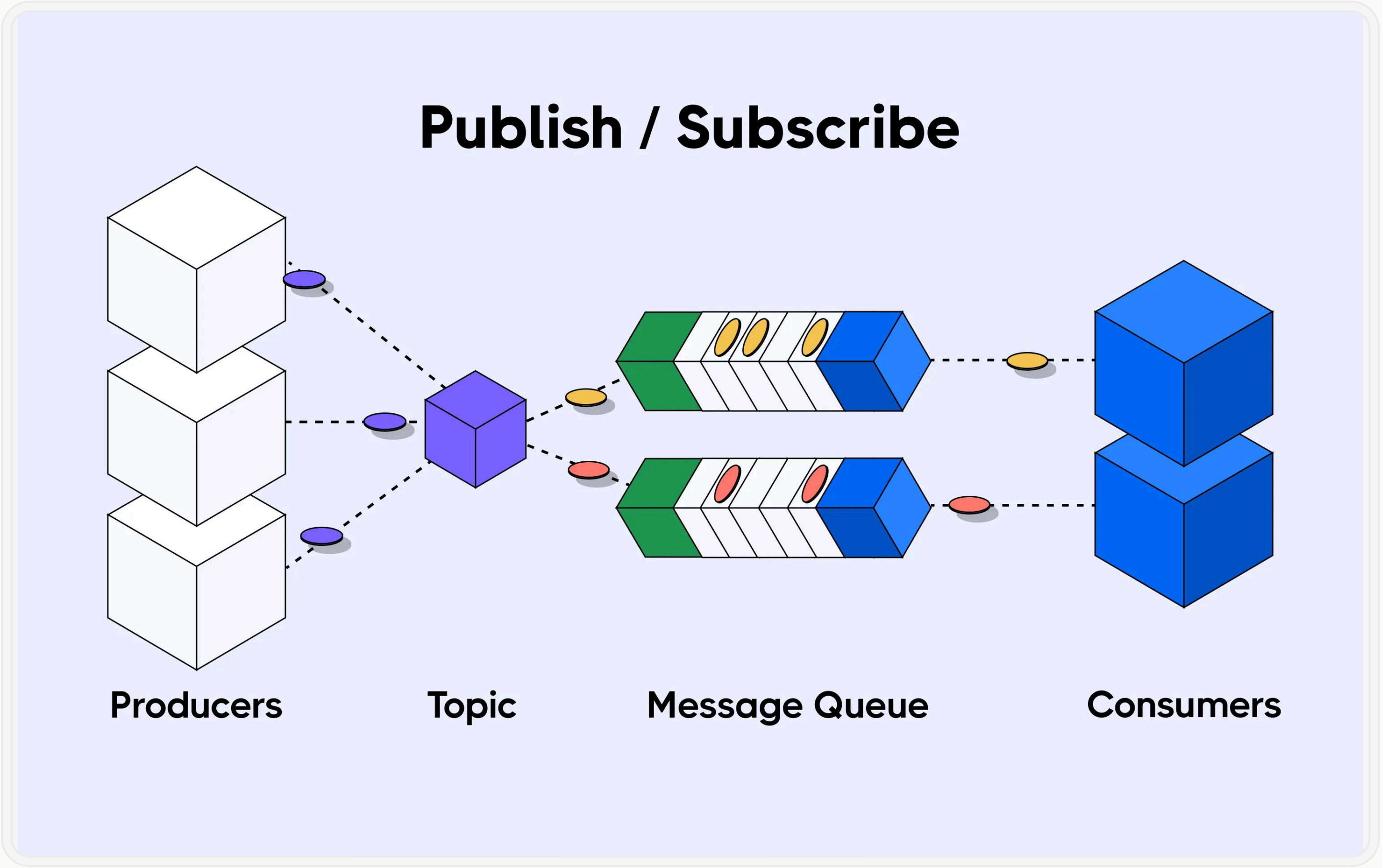
When to use WebSockets

Use WebSockets when you want two-way communication between two networked systems, for example when building a chat application. WebSockets are also very useful for data visualization dashboards or maps that need to reflect real-time data values.

Webhooks or pub/sub

How pub/sub works

Pub/sub, which is short for publish/subscribe, is a communication system for distributing messages between a set of publishers (message producers) and subscribers (message consumers). A pub/sub system buffers messages from producers and routes them to subscribers through the use of dedicated channels known as topics. Publishers publish messages to topics and subscribers express interest by subscribing to the topics.



Once a message is published to a specific topic, the message is cloned and sent to all the subscribers subscribed to it.

Differences between webhooks and pub/sub

In a pub/sub system, message sources are decoupled from message consumers while in webhooks, the message producer is fully aware of the location of the consumer through the webhook URL.

Webhooks are a direct form of communication between the producer and consumer while pub/sub is a middle-man framework that routes messages from publishers to subscribers. The communication setup in a webhook is one-to-one, i.e. one producer to one consumer, while in a pub/sub system you can have many producers sending messages to multiple consumers.

When to use pub/sub

Use a pub/sub system when you have multiple consumers interested in the same message or more than one message producer.

Also, use a pub/sub system when you want to asynchronously process messages. Webhooks are synchronous.

An example of a situation where a pub/sub system is appropriate is in an e-commerce site. When a customer makes a purchase, you need to mail an invoice to the customer, send information to the delivery department, and enter the sale record in your CRM. Let's imagine that a mailing service, delivery service, and records service handle these responsibilities respectively.

That is three systems interested in the same message. You can publish this message to a pub/sub system, which will then route it to these three subscribed services.

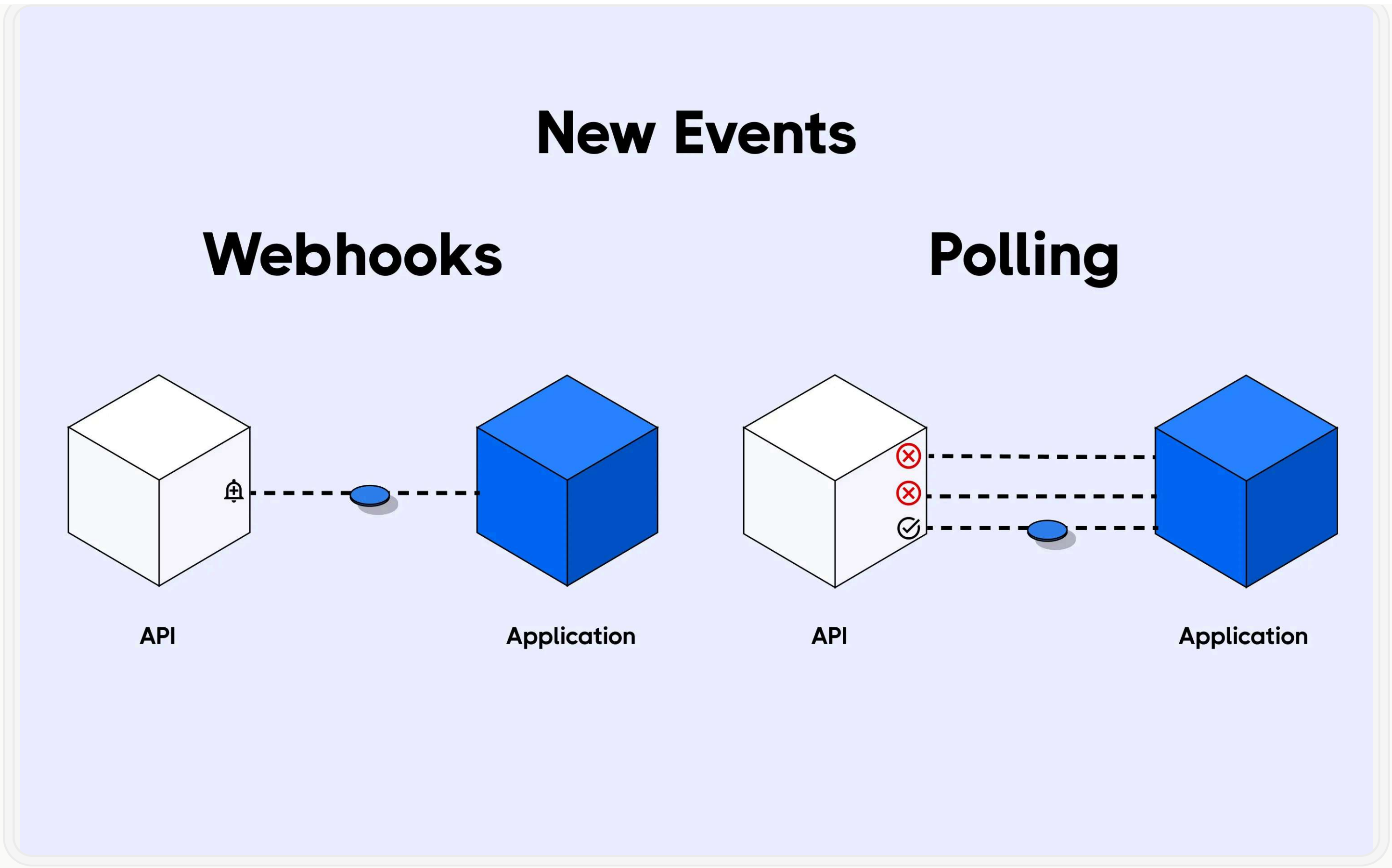
Webhooks or polling

How polling works

Polling involves periodically making requests to a system to check for new events or data. If new data is found, a response is returned with the new data in its payload. If no new data is available, nothing is returned.

Polling is used to query systems for new information and can be set up as an automated cron job that runs at certain intervals.

To capture the difference between these two approaches with a relatable example, polling is like going to the post office to check if you have new mail. Using webhooks is basically having mail delivered to your house every time you have new mail simply by giving the postman your house address.



How webhooks differ from polling

Polling uses the pull model of communication where a system pulls information from another system, while webhooks use the push model by pushing information from a source application to a destination application.

Polling requests are made by a client, while webhook requests are made by a server. Webhooks are also automatically triggered when an event occurs, whereas polling is set up to run at fixed intervals and runs whether there is a new event or not.

Polling can be resource-intensive and you need to make calls on whether the efforts will be fruitful or not. This is not the case for webhook requests, which are only made when there is new information.

When to use polling

Polling can be used when you don't need real-time updates, which cause frequent changes in data and can crash your system if you decide to receive updates every single time something changes.

Imagine you're running a successful startup and the number of users is growing by a hundred users each second. You want a big screen in the middle of the office to display your current subscriber count. Subscribing for new data each time a new user signs up would be chaos.

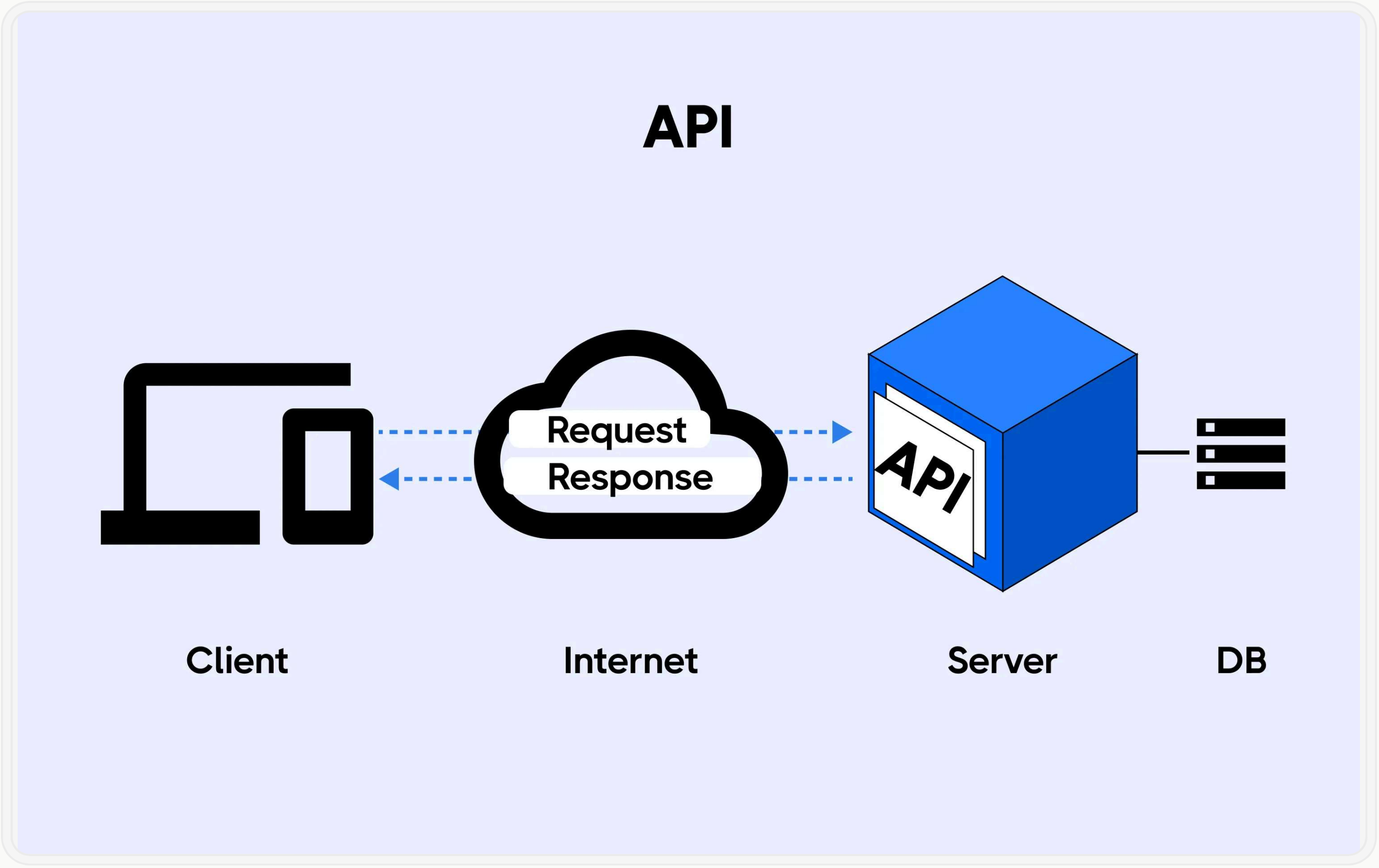
What you should do is poll your server every five to ten minutes to fetch the current total number of users. This would be lesser work for your system and ultimately would consume lesser resources.

Webhooks and APIs

What is an API?

An API is an application's gateway to the world. It is the interface that allows other systems to interact with the application. The application can expose as much as it wants and keep its remaining parts abstracted.

APIs are usually a collection of endpoints that clients can call to communicate with the system. This communication is done via request methods such as GET or POST, and may carry some information related to the action the client is willing to perform.



How webhooks differ from APIs

Webhooks make calls to APIs. An API provides webhooks with the entry point to push data to an application. When an event occurs in a source application, a webhook request is triggered to one of the API endpoints.

When to use webhooks

Now that you have a good understanding of how webhooks and other communication systems work, in what scenarios is it then appropriate to go with webhooks?

Webhooks are a simplified model of communication thus, you should use webhooks when you require the following:

- Real-time one-way communication (from source to destination)
- A non-persistent connection between the two systems' communication
- You want to respond immediately to an event from a SaaS application that supports webhooks
- You want to use the push model to immediately push updates
- The communication is one-to-one

A lot of SaaS applications use webhooks for communication — for example, Shopify uses webhooks to communicate events like when a shopping cart is updated or a sale is made. Stripe uses webhooks to communicate events like account updates, payments, etc.

Examples of these types of scenarios include:

1. An e-commerce store notifying your invoicing application about a sale
2. E-commerce stores notifying merchants when a particular item is out of stock
3. Payment gateway notifying merchants about a payment
4. Version control systems notifying team members about a commit to a repository
5. Monitoring systems alerting administrators about an error or unusual activity in a system
6. Synchronizing information across systems — for example when a user changes their email in your HR or CRM system, their email is also changed in the payroll or invoicing system

Examples of sites that use webhooks

Sites that send webhooks for notifications and information sharing

1. Twilio webhooks convey information about events such as delivered SMS messages, voice calls, and authentication
2. Slack webhooks post messages from apps into Slack
3. Shopify webhooks sync with Shopify and execute code when an event takes place in your store
4. Stripe webhooks notify your application when an event occurs in your account ... there are many more examples.

Sites that process webhooks

1. Hookdeck: ingests, scales and monitors webhooks traffic
2. Zapier: uses webhooks to connect different apps in your workflows
3. IFTTT: connects different apps and devices to extend their functionality

Conclusion

Knowing the right tools to solve a problem is what sets engineers apart. The fact that a technique is simple, fancy, or largely adopted doesn't automatically make it the best tool for the job. Webhooks are straight-forward and simple to implement, but may not always be the right way to go for the problem you're trying to solve. Being able to understand the advantages and limitations of the webhooks process will hopefully help you understand the right places to implement it.

Gain control over your webhooks

Try Hookdeck to handle your webhook security, observability, queuing, routing, and error recovery.