Web Hooks

In this lesson, we'll understand the need for web hooks & how do they work?

We'll cover the following

What Are Web Hooks?

Imagine you've written an *API* which provides information on the latest exclusive events on *Baseball*. Now your *API* is consumed by a lot many third-party services, that fetch the information from the API, add their own flavour to it & present it before their users.

But so many API requests every now and then, just to check if a particular event has occurred is crushing your server. The server can hardly keep up with the requests. There is no way for consumers to know that the new information isn't available on the server yet, or an event hasn't occurred yet. They just keep polling the API. This would eventually pile up the unwanted load on the server and could bring it down.

What do we do? Is there a way we can cut down the load on our servers?

Yes!! *WebHooks*.

WebHooks are more like *call-backs*. It's like I will call you when new information is available. You carry on with your work.

WebHooks enable communication between two services without a middleware. They have an *event-based* mechanism.

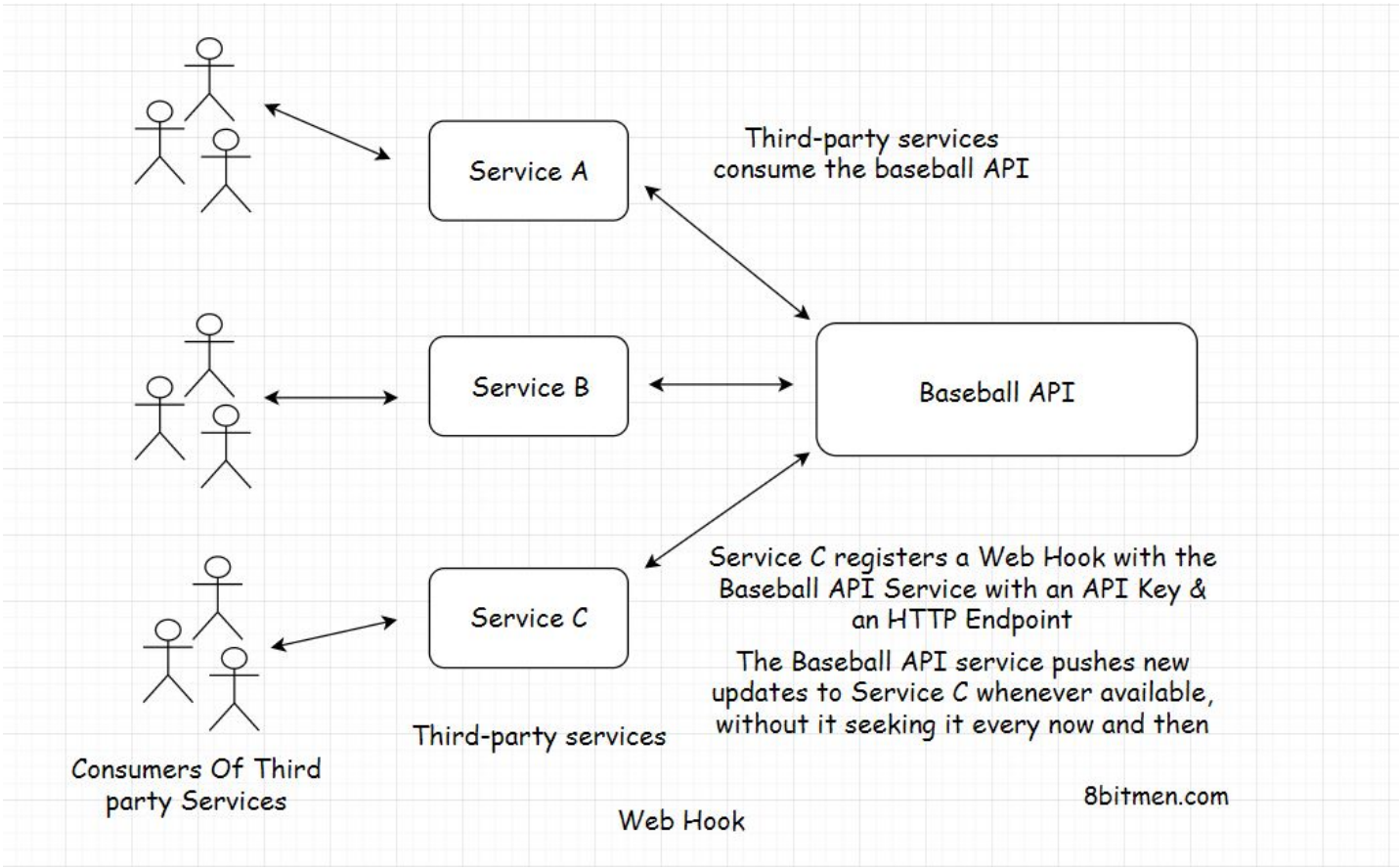
So, how do they work?

How Do Web Hooks Work?

To use the *Webhooks*, consumers register an *HTTP* endpoint with the service with a

unique *API* Key. It's like a phone number. Call me on this number, when an event occurs. I won't call you anymore.

Whenever the new information is available on the backend. The server fires an *HTTP* event to all the registered endpoints of the consumers, notifying them of the new update.



Browser notifications are a good example of *Webhooks*. Instead of visiting the websites every now and then for new info, the websites notify us when they publish new content.