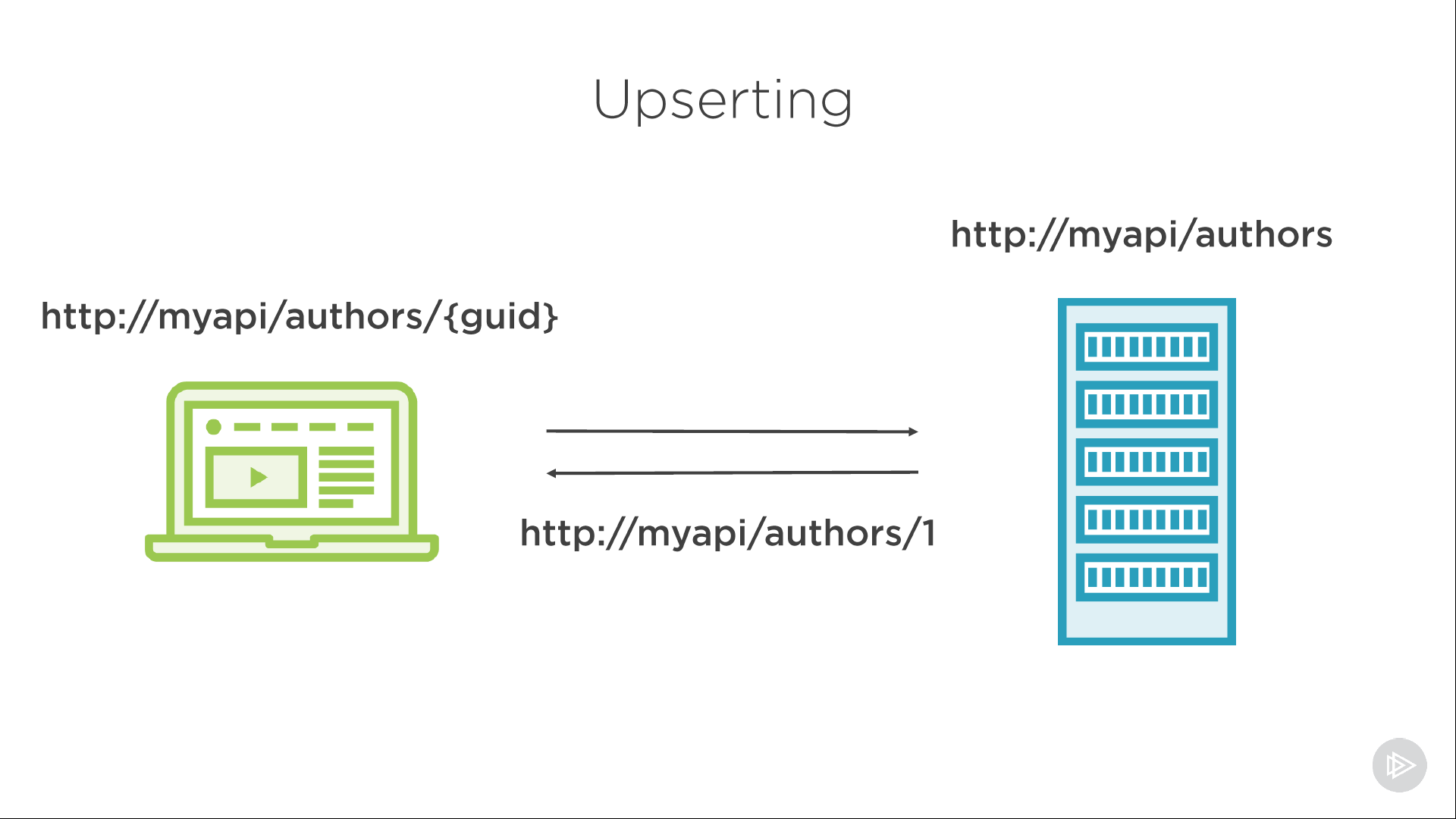
Upserting

* In the beginning of the module, we touched upon the possibility to create a new resource with PUT or PATCH instead of POST. That principle is called **upserting**



* We've got a consumer of our API on the left and the API server on the right. In a lot of systems, it's the server responsible for creating the identifier of the resource. Part of it is often the underlying key in the data store, an integer value, for example.
* A consumer might post a new author to the author's resource and the server responds with that newly created author in a response body and the location of the author resource, including the Id of the server generated in the location header.
* In fact, in most systems, the server decides on the resource URI, but REST doesn't have this as a requirement. It's perfectly valid to have a system where the consumer can do this, or where it's allowed for both consumer and the server.
* Now, if the key in the database is part of the resource URI, this doesn't just work with num fields. It does, however, when working with GUIDs. So, yeah, you got me; this is one of the reasons I chose to use GUIDs instead of ints to store data in the back-end store.
* Now let's think about what could happen. Say the server is responsible for creating the resource URI and we want to send an update request.
* We need to get the URI to a resource from the server to be able to update it. The resource must exist already, and if it doesn't exist, we must return a 404 Not Found.
* Now, let's imagine the consumer is also allowed to create resource URIs. In that case, we no longer need to get the URI from the server. The author requirement for the resource having to have been created before vanishes.
* We can now send a PUT request to a previously unexisting resource identifier that is valid because the consumer is allowed to create it.
* If that's the case, that resource must be created by sending the PUT request. Or, in a way, it's updated from being empty.
* So, if the server is responsible for creating the resource identifiers, we must use POST to create resources. We cannot know the URI of the resource in advance.
* But if the consumer of the API is allowed to create a resource identifier, well, we can use PUT as well, and this is called upserting.
* Now, let's think back about the method idempotency to see if this still fits. We learned that POST isn't idempotent; sending the same request more than once will result in different outcomes. PUT, however, is, and that fits.
* If the consumer of the API chooses the ID, sending the request once will create the resource. Sending it again after that will have the exact same result. We are now updating their resource. So the outcome is the same.

