Serverless

This lesson explains Serverless and its relation to FaaS.

We'll cover the following

Often the term *Serverless* is used interchangeably with *Functions as a Service*. So, can we safely assume that these two terms mean the same thing?

Well, not really!!

> *Serverless* is an approach. *FaaS* is a way to implement the Serverless approach.

Let's begin this lesson with an understanding of what *Serverless* is.

What is Serverless?

Serverless does not mean there are no servers involved when running the service. It simply means the servers have been abstracted away from the developers, and they do not have to worry about them.

Serverless is an approach that enables developers to focus on the code and not on the servers. Via this approach, the onus of the infrastructure administration on the developer is zero. That means there is no need for them to manage the server fleet, *OS*, and so on.

Well, that's the whole idea of cloud computing, isn't it?

To take the responsibility of managing the infrastructure away from the

developers as much as possible. The Serverless approach provides the highest level of freedom from infrastructure management to the developers.

Also, Serverless, being a relatively new approach, has no standard definition. A general idea is if you are heavily using managed services like *Database as a Service (DBaaS)*, *Messaging as a Service*, and so on for your application, you just want to focus on creating value-- that is writing the business logic and limiting your exposure to the infrastructure. You have a Serverless architecture.

As I stated earlier, *FaaS* is a way to implement the *Serverless approach*. We can also have a non-Serverless approach using *FaaS*. That makes the developer responsible for running his functions, monitoring and maintaining them, making them talk with other backend services, provisioning the servers, and so on. This disputes the fact that FaaS is not *Serverless*; it's a way to implement *Serverless*.

Ways to implement the Serverless approach

There are two primary ways cloud providers enable businesses to go Serverless:

1. Using - Backend as a Service (BaaS)

2. Using - Function as a Service (FaaS)

We've already discussed *FaaS*, so let's have a quick insight into the *BaaS* service model.

Backend as a Service (BaaS)

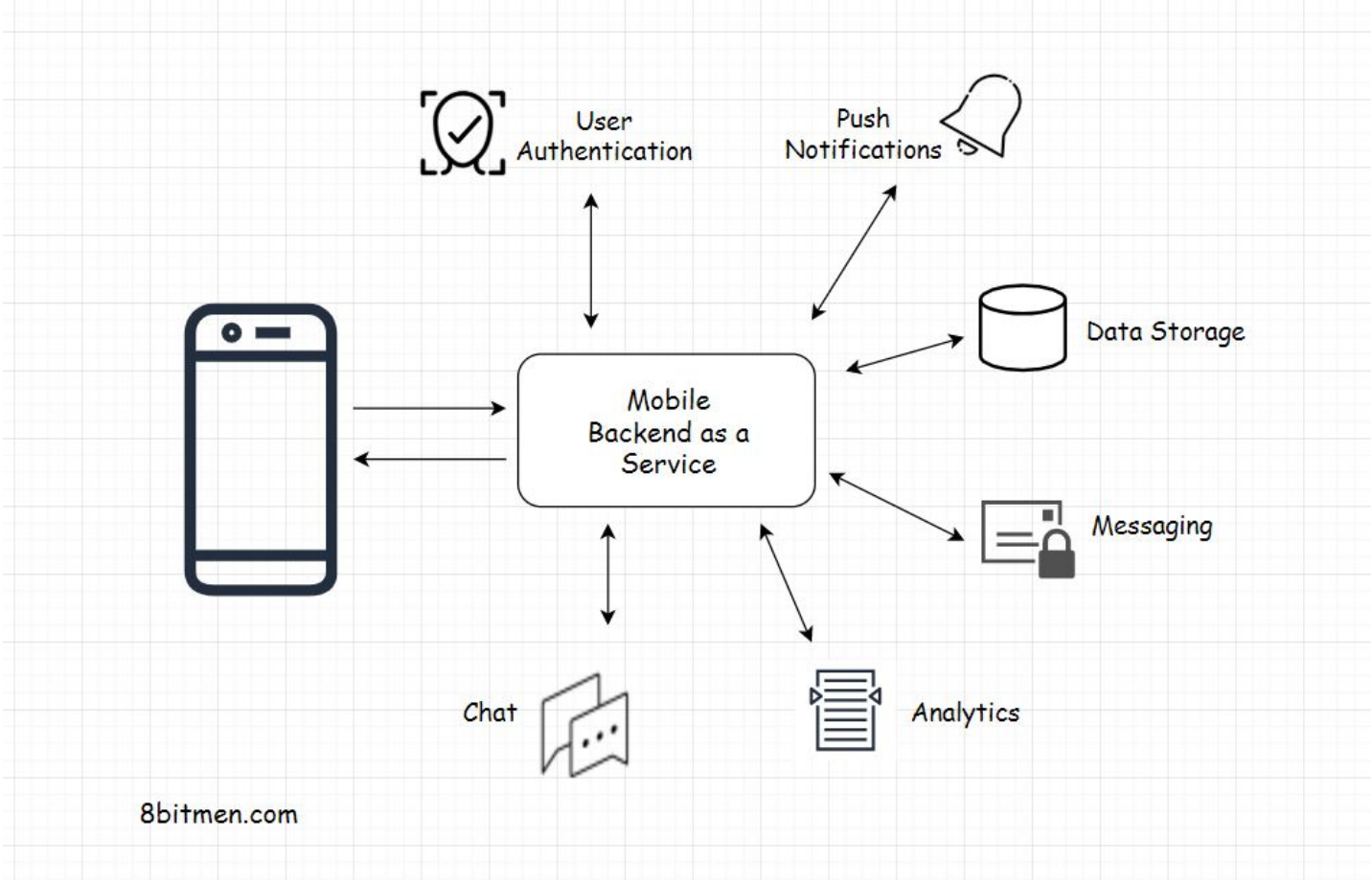
Backend as a Service as the name implies, is an entire backend available to the developers as a managed service. The idea is to enable the developer to focus on the business logic and let the platform take care of the rest.

A few examples of this are *Google Firebase* and *AWS Amplify*. These products are backend services for mobile apps also known as _Mobile Backend as a Service (MBaaS) _.

Mobile Backend as a Service (MBaaS)

Besides business logic and the user interface, a mobile app-based service contains several other key features that collectively make the service functional, top-notch, and worthy of getting the user's attention. These features are user authentication, integration with social networks, push-notifications, real-time database, caching, data storage, messaging, chat integration, integration of third-party tools, analytics,

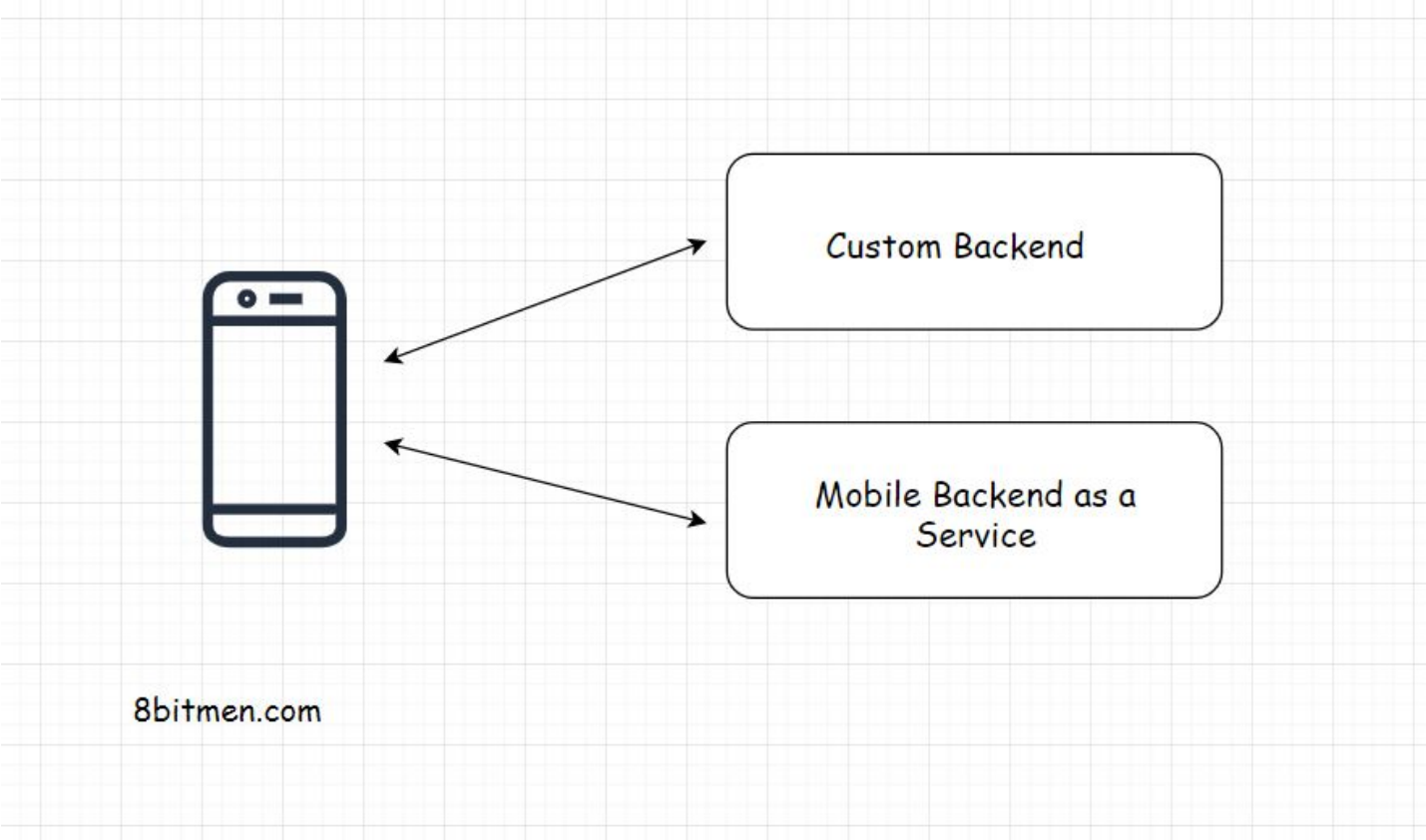
crash reporting, and so on.



MBaaS takes care of all these features making a developer's life a lot easier during the bootstrapping phase. It typically offers an *API* for every feature. There will be an *API* for user authentication, an API for real-time database, an API for messaging, and so on. Our code can directly interact with the respective API to exchange information.

MBaaS is great for mobile-only services and for use cases where you do not need, or you don't already have, a custom backend up and running for your service. In the case of an *MBaaS*, all the business logic resides on the client, which is the mobile app. The app is a *Fat client*.

We can also use *MBaaS* and a custom backend set up in the same app in scenarios when we are required to integrate a legacy enterprise system into our mobile app or when we need to leverage some additional features that the custom backend server hosts. Think of a banking app built using an *MBaaS* that needs to interact with the legacy enterprise backend to cross verify the data entered by the user every time.



In the next few lessons, we will have a look at some cloud computing jargon, such as the *cloud workload, instances*, and so on. You might have already seen these terms in the prior lessons, and they will continue to appear over and over throughout the course.