# Serverless란?

Server(서버) -less(없다).
이름 때문에 백엔드에 서버가 없다. 라고 생각할 수 있으나,
백엔드인데 '너가 직접 서버를 관리하지 않아'를 뜻한다.

서버를 관리하지 않는다고?
직접 하드웨어, 네트워크, 장비..들을 구성해서 IDC에 서버를 넣어놓거나, AWS, GCP, Azure 등 가상 호스팅을 이용을 하여 서버를 빌려서 사용을 하는데 서버가 어떤 사양으로 돌아가고 있는지, 서버의 갯수를 늘려야 할지, 네트워크는 어떤걸 사용할지, 보안에 대한 처리 등 이런걸 설정하고 관리할 필요가 없다는 뜻이다.

물리적인 서버가 아예 없는 구조는 아니고, 서버에서 처리하는 작업들을 클라우드 기반의 서비스로 처리해서 서버 구축 및 관리 비용을 줄이는 구조이다.

서버리스를 활용하면 백엔드를 서버에 올리는 것이 아니라 백엔드를 작은 함수단으로 쪼개서 직접 관리하지 않는 서버로 올린다. (대표적으로 AWS람다)
개발자는 로직이 담긴 함수 구현만 신경쓰면 된다.

사용자가 원하는 로직을 함수로 작성해놓으면 서버는 항상 대기하면서 사용자의 요청을 처리하는 것이 아니라, 이벤트가 있을때마다 실행된다. VM이 잠자고 있다가 request가 들어오면 깨어나서 함수를 실행하고 종료된다.

기존의 서버구축 방식대로는 사용자가 있던 없던 항상 서버를 유지하고 있어야 하기에 계속 비용이 발생하고 있지만, 서버리스에서는 함수가 실행되는만큼만 비용을 지불하면 된다.
갑자기 천명의 유저가 생겼다면 AWS는 같은 함수의 복사본을 천개를 만들어서 수행을 해버리고 유저들이 사라지면 복사본의 함수들은 종료한다.
퍼포먼스에 영향을 끼치지 않는다.

이러한 장점이 있는 반면 당연히 단점도 존재한다.

VM이 잠자고 있다가 request가 들어오면 깨어나서 함수를 실행하고 종료하기 때문에 항시 대기하고 있는 서버에 비해 실행속도가 더 필요하게 된다.
AWS람다는 어떤 함수가 자주 쓰이는지 파악을 해서 해당 함수는 아예 잠들지 않고 request

에 빨리 대응할 수 있도록 대기한다고 하지만 그래도 아주 미세하게라도 서버가 응답하는 시간이 걸릴 수 있다.

그리고 서버제공자에 대한 의존도가 높을 수 밖에 없다.
AWS에서 서버를 빌렸다가 구글 클라우드로 이사가는 것은 쉽지만 서버리스로 작업하면 어플리케이션의 구조 자체가 바뀌게 되니 서버리스에서 이사가는 것은 간단하지 않다.

그럼에도 불구하고 코드에만 집중할 수 있기 때문에 빠르게 프로토타입을 출시하고 싶은 경우라던가 사이드 프로젝트를 하는 경우에는 서버리스를 활용하는 것이 큰 장점이 될 수 있다.