In today's world if you say you don't interact application with microservices may not be true. Nowadays, every person has to interact with the microservices intentionally or unintentionally. The daily life use of microservices are following.

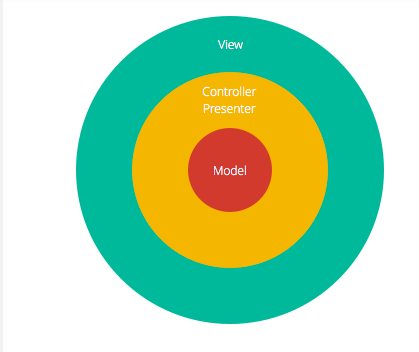
One of the most common taxi app uber is using microservice architecture

Netflix is one of the companies who is the first company to start using microservice and get an excellent result.

Before moving toward the microservices let's discuss the other software engineering Architecture.

# Layered Architecture

in this architecture, the application is divided into three layers as shown in the figure

The modal layer is simple the database. just above the database is the controller Presenter which contains business logic and information about the types of data in the database. furthermore, the top layer is view layer or presenter layer which is simple the frontend usually written in HTML, CSS, etc.

the biggest advantage of a layered architecture is each layer can focus on its role.

Event-driven Architecture

Program written with event-driven architecture mostly wait for an event to occur. when there is any need for the module to execute it is called otherwise it waits for the interruption. An example of this architecture can be on the web page where the small module reacts when the event occurs like a mouse click or keystroke. This module gets activated when the event concerning them is interrupted.

Space-based architecture

The website is alive until and unless the database can handle the load. If a database cannot handle it the website will be failed.that is the reason why space-based Architecture is very use full. The phenomena of working are it splitting up both the processing and the storage between multiple servers. The data is spread out across the nodes just like the responsibility for servicing calls.

Monolithic Architecture

Monolithic is an approach in which all code is in one piece. Monolithic application has a single code base with multiple modules.

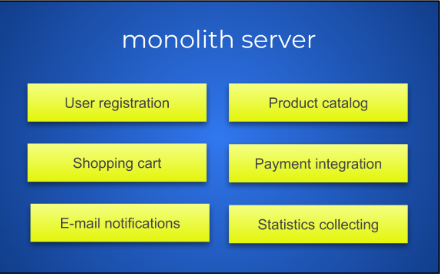
The most latest architecture is Microservice Architecture. now let's discuss it in more detail.

Microservices Architecture

Software is like the baby of the elephant, as it is baby it is cute but once it gets big it becomes difficult to control. Here comes the microservice where the program is divided into tiny pieces these pieces are then joined into one little program.

let's take the example of e-commerce website we want to add the following feature

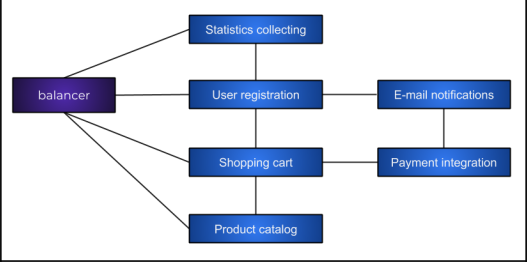
* User registration
* Product catalog
* Shopping cart
* Payment integration
* E-mail notifications
* Statistics collecting

if we follow the old fashion we will have to use a monolithic server. we can convert this into microservices.

Before moving forward you much know what are loosely coupled and high cohesive.

loosely coupled: In computing and systems design a loosely coupled system is one in which each of its components has, or makes use of, little or no knowledge of the definitions of other separate components.

Highly cohesive: High cohesion is when you have a class that does a well-defined job.

If you want to apply microservices on this e-commerce site instead of packing it into one program you have to make the separate module for each feature.

(3)

the load balancer has an important role in this situation it is working as the traffic cop which controlling the traffic and guide the client to their respective feature which they have asked for.

Disadvantage of microservices

* The services must be [largely independent](http://techbeacon.com/5-fundamentals-successful-microservice-design) or else interaction can cause the cloud to become imbalanced.
* [Not all applications](http://techbeacon.com/why-your-project-may-not-be-ready-microservices) have tasks that can’t be easily split into independent units.
* Performance can suffer when tasks are spread out between different microservices. Communication costs can be significant.
* Too many microservices can confuse users as parts of the web page appear much later than others.

Best for

* Websites with small components
* Corporate data centers with well-defined boundaries
* Rapidly developing new businesses and web applications
* Development teams that are spread out, often across the globe