**Micro Services**

**What is Micro service:**

\* Micro service is an architectural style that structures an application as

a small autonomous service modelled around a business domain.

\* In microservices, an application is divided into services. Each service runs a unique process and manages its own database.

\* All microservices are separated based on their domain and functionalities.

\* All microservices communicate with each other through a server either REST or Message bus.

\* All functionalities performed by micro service are communicated to clients via API Gateway

**Features of microservice:**

\* Decoupling

\* Agility

\* Componentization

\* Autonomy

**Advantages of microservice:**

\* Easily deployed

\* Requires less development time

\* Can scale quickly

\* can be reused in different projects

\* Work well with containers

**Disadvantages of microservice:**

\* Potentially too much granularity

\* Extra effort designing for communication between services

\* Latency during heavy use

\* Complex testing

**Practices to design microservices:**

\* Separate data storage for each microservice

\* Treat servers as state less

\* Deploy into containers

\* Separate build for each microservice

**Created project on how to convert xml format into json format:**

**Flow be like:**

HTTP Listener-> Transform message-> Logger

**HTTP listener:**

The HTTP listener is an event source that enables you to set up an HTTP server and trigger flows when HTTP requests are received.

You can choose what methods the source accepts, such as GET, POST or a list of methods, and on which path to accept requests, thereby allowing the routing of requests through different flows.

Once a request is accepted by the listener, the corresponding flow is triggered with the HTTP body as payload and the HTTP data as attributes (headers, query parameters and so on).

When the flow finishes its execution, the HTTP listener enables you to customize the HTTP response based on whether the execution was successful or not, so that different status codes can be returned.

**Transform Message**:

The Transform Message component carries out transformations over the input data it receives. You can explicitly write out a transformation in DataWeave language.

It is used to transform one format of data into any format of data.

**Logger**:

This component helps you to monitor and debug your Mule application by logging important information such as error messages, status notifications and payload.

**Flow Explanation**:

1. Take HTTP listener which listens the HTTP requests.

2. Our target is to transform the xml formatted data into json formatted data.

3. Logger to print the information on console and if in case we get any errors it will show you on console.

4. Go to postman to check the flow whether it is working properly or not.

5. Send some xml formatted data as input so that we'll get required output as well in result.