**Mule Flow:**

\* Mule apps process messages and other parts of Mule events through Mule components, connectors, and modules that are set up within the scope of Flow and Subflow components within an app.

\* Flows provide the most robust and flexible way to build Mule applications, because you can arrange convenient building blocks. Flows support mechanism and asynchronous synchronization of the flows, one-way and request - response exchange - the model and other architectural plans.

**Sub Flow:**

\* Subflow is a normal flow without any message source.

\* A subflow can be called by the flow-reference element of Mule. When the main flow (calling) calls the subflow using the flow-reference element, it passes the whole message structure (message properties, payload, attachments, etc) and the context (session, transaction, etc). In the same way, when the processing of the message is done in the subflow, the complete message and context are returned to the main calling flow. In other words, everything in the subflow behaves as if it were in the same flow. It's important to note that the message is executed in the same thread.

**Usage:**

Subflow can be used when some block of code (components) can be reused across the application and you don't want to handle the exceptions of the subflow separately from the main calling flow.

**Mule event:**

A Mule event contains the core information processed by the runtime. Mule events are immutable, so every change to an instance of a Mule event results in the creation of a new instance.

**A mule event consists of:**

- Attributes

- Payload

- Variables

\* In Mule 3, when the request is received by Mule message source, it is converted into a Mule message or flow and starts processing it when it receives an inbound endpoint in a flow.

Variables in Mule:

Variables are used to store per-event values for use within a flow of a Mule app. We can store the current message (using the message keyword), the current message payload (using the payloadkeyword) or just the current message attributes (using attributes keyword).

You can create or update variables in these ways:

- Using the Set Variable component.

- Using a Target Variable from within an operation, such as the Read operation to

the File connector or the Store operation to the Database connector.

- Using the Dataweave Transform Component (EE-Only).

- Using Scripting Component (in scripting module)

- We can delete variable using remove variable component

**JMS Connector:**

The JMS connector enables your application to do messaging using the JMS implementation of your choice.

Its main features include:

- Pub/Sub pattern support on any given destination.

Listen/Reply pattern support on any given destination.

Publish-Consume pattern support on any given destination, with fixed or temporary reply Queue.

Fully compliant with JMS 2.0, 1.1, and 1.0.2 specifications.

**Uses:**

JMS Connector JMS (Java Messaging Service) is mostly used API enabling the application to communicate through the exchange of message. JMS connector is capable of sending and receiving message to and from Topics/Queues.

JMS supports two models for messaging:

Queue (point to point).

Topic (publish-subscribe).

**Queue**:

It enables one-to-one communication. It is also called point-to-point communication.

The sender will deliver a message to the queue and single receivers will pick the message from the queue.

The receiver doesn't need to listen to queue at the time when the message is sent to the queue.

**Topic:**

\* It enables one-to-many communication. It is also called publish-subscribe

communication.

\* The publisher will deliver the message to a topic and it will be received by all

subscribers who are actively listening to the topic.

\* A subscriber will miss the published message if it is not actively listening to the

topic unless messages are made durable.