**Mulesoft:** [https://training.mulesoft.com/](https://training.mulesoft.com/user/consume/course_pathway/b289702c-fdf1-3589-aa34-64077b11032e)

[stk.rajee@gmail.com](mailto:stk.rajee@gmail.com)

Rajee3483

**Anypoint platform login**: <https://anypoint.mulesoft.com>

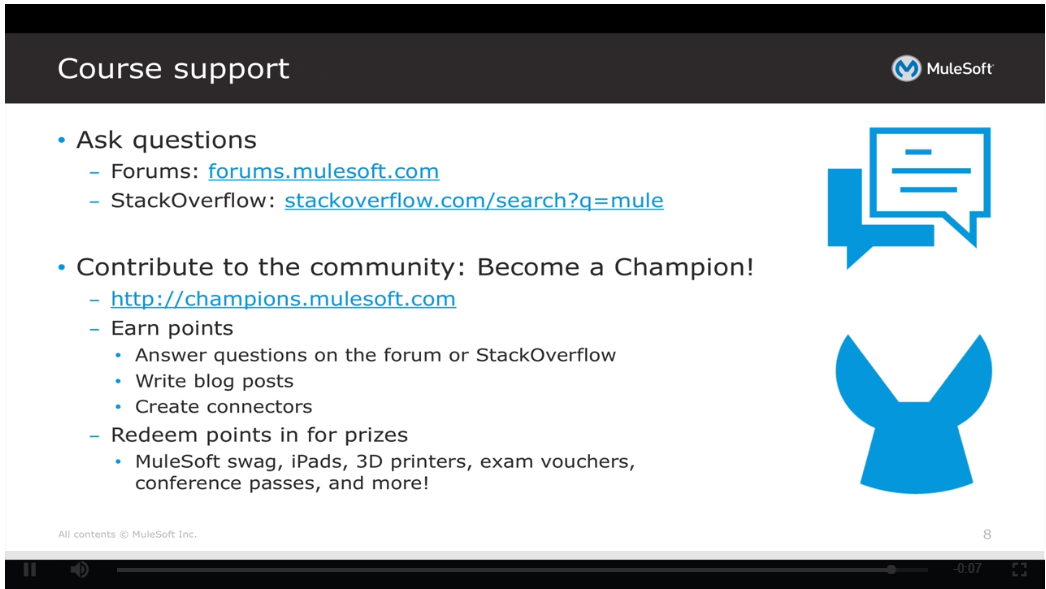
stk\_raajee/Rajee3483

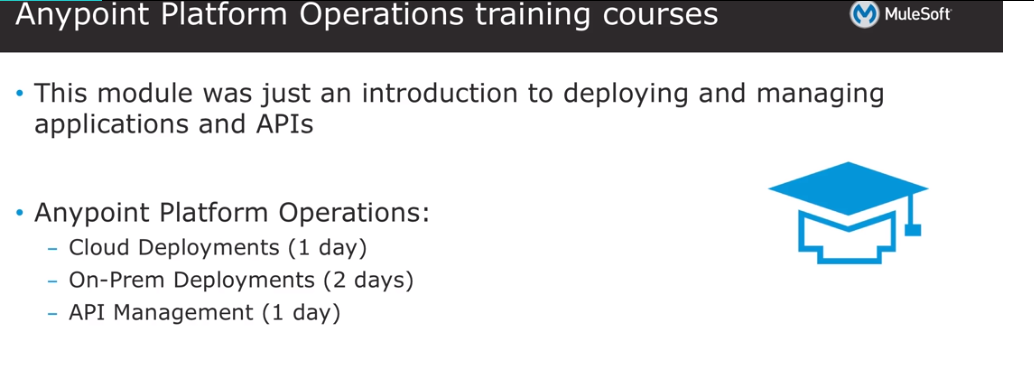
**Salesforce Login** : <https://developer.salesforce.com/>

[stk.rajee@gmail.com](mailto:stk.rajee@gmail.com) / Rajalekshmi3483

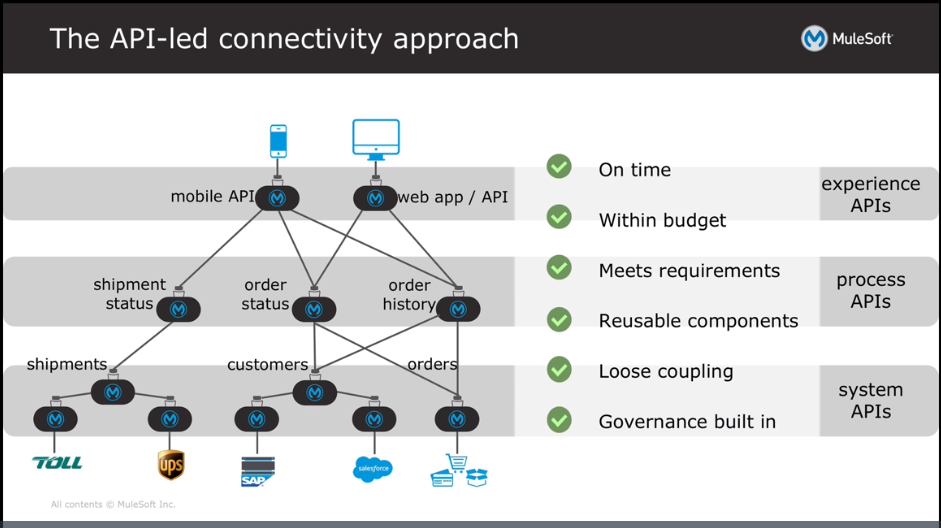
SalesForce Security token (case-sensitive): GzQV31klOCMS2a9VrWXz5ClU

How to create a mule domain project: <https://blogs.mulesoft.com/dev/training-talks/how-to-create-a-mule-domain-project/>





**Introducing API led Connectivity**

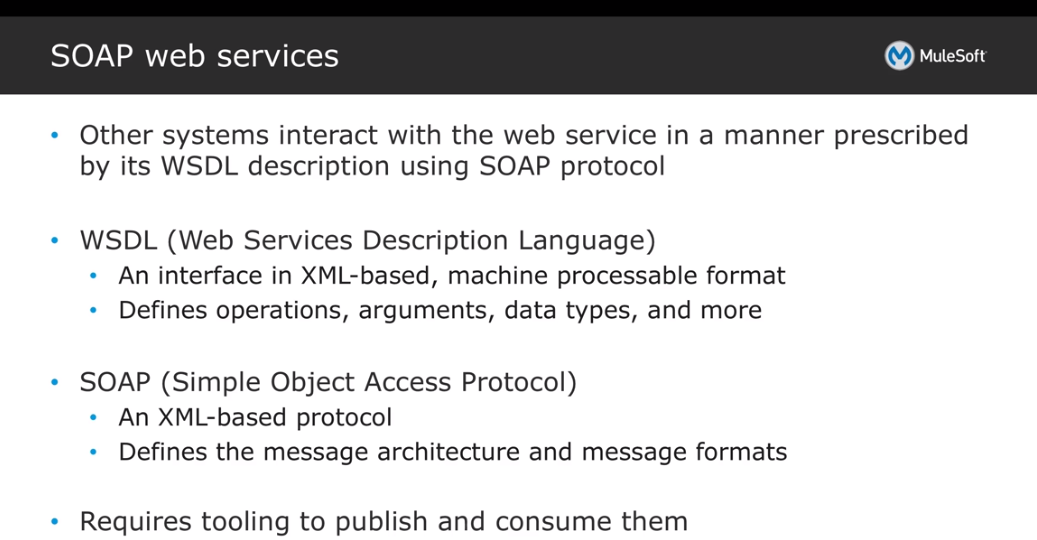


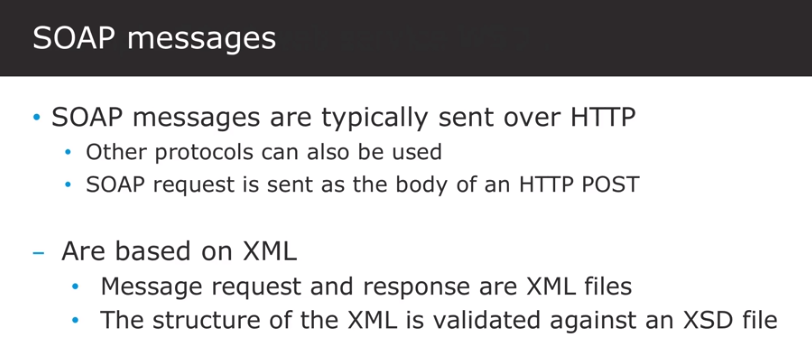
API: Application Programming Interface. It provides the information for how to communicate with a software component, defining the operations(what to call), Inputs(what to send in a call), outputs(what you get from a call),underlying datatypes

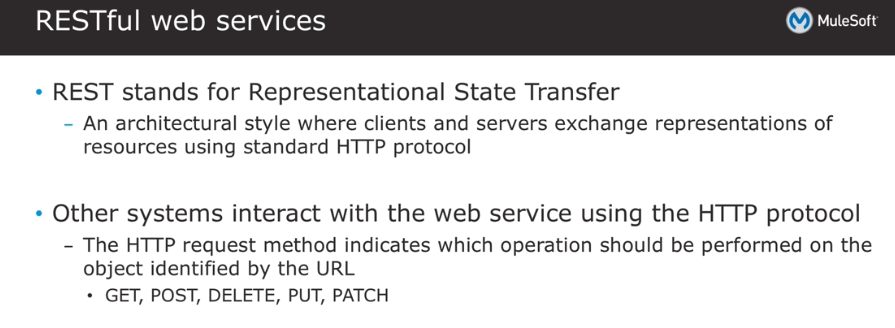
Webservice: Method of communication that allows two software systems to exchange data over the internet

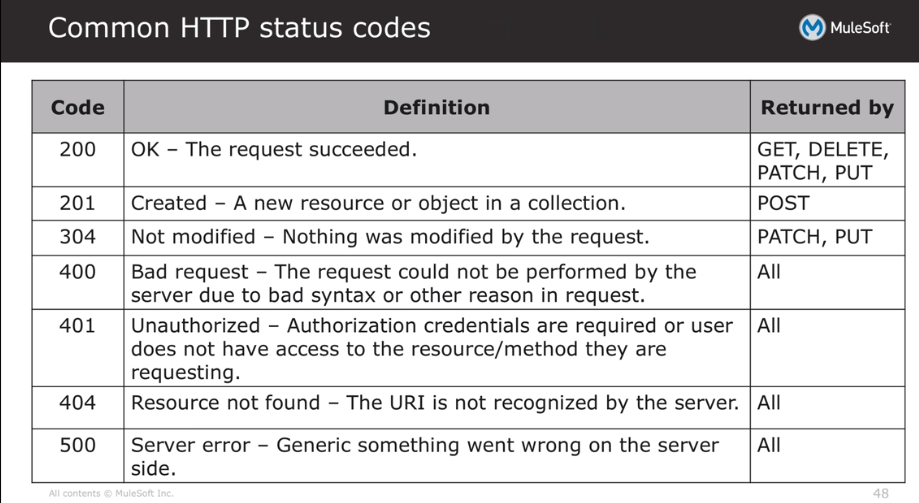
Parts of webservice : Webservice API, Web service interface implementing the API, Actual Webservice implementation

Two Types: SOAP(Communication Rules are defined in WSDL file), REST(Use the existing HTTP communication protocol)









**Anypoint Platform:** Collection of runtimes, frameworks, tools and web applications

* Tools and frameworks for building applications
* Mule runtime for running applications and applying policies on prem/cloud
* Web application for deploying running, managing, monitoring applications. Defining, managing, and discovering apis

Anypoint Designer – To design apis and get the RAML files

API Console/Mocking service – Simulating

API Portal/Exchange – Feedback

API Notebook – Validating

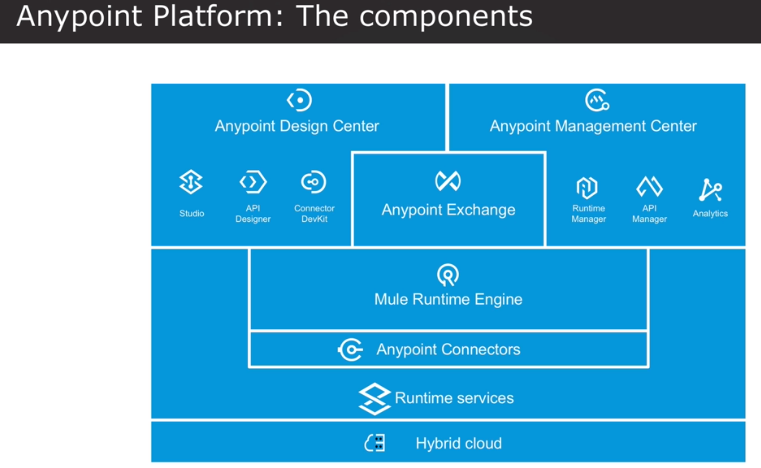
Anypoint Studio – Building APIs

Munit – Writing testcases

API Manager – Versioning & Security

Run time Manager – Deploy & Register, Monitoring

API Analytics – Analyze



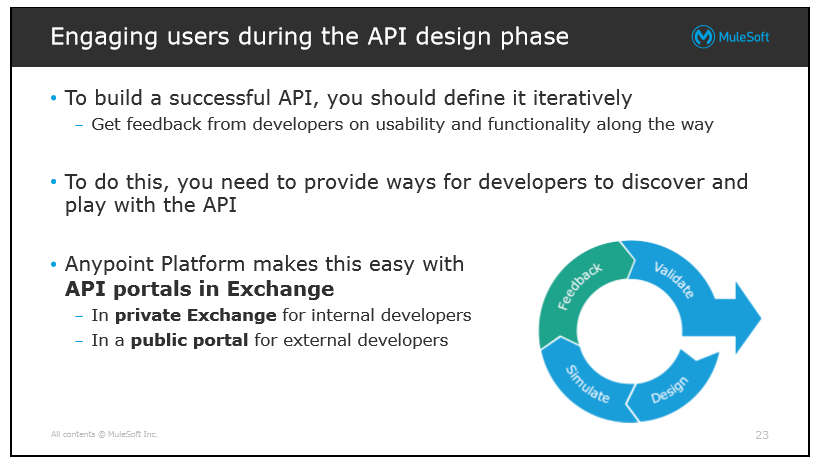
**Designing APIs**

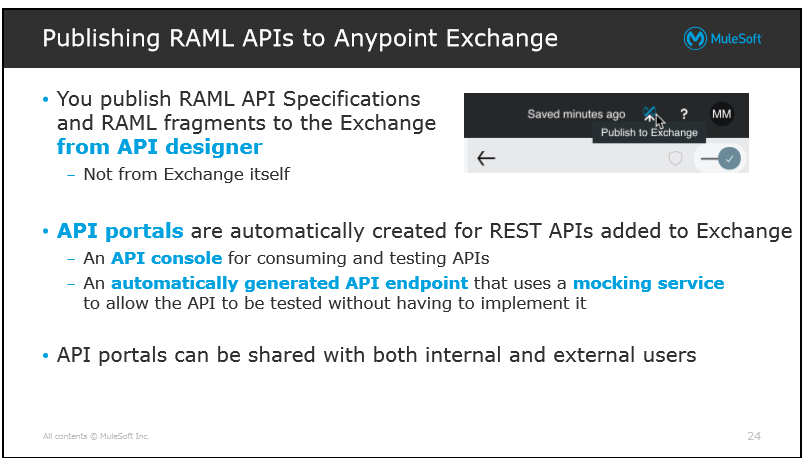
Different approaches for designing API : Hand coding, Swagger, Blueprint, RAML(Restful API Modelling Language)

RAML :

1. Simple RAML Creation
2. Including fragments in the RAML file from design center
3. Including different files and folders within the project

Engaging Users:





Publishing it will make it available for internal users

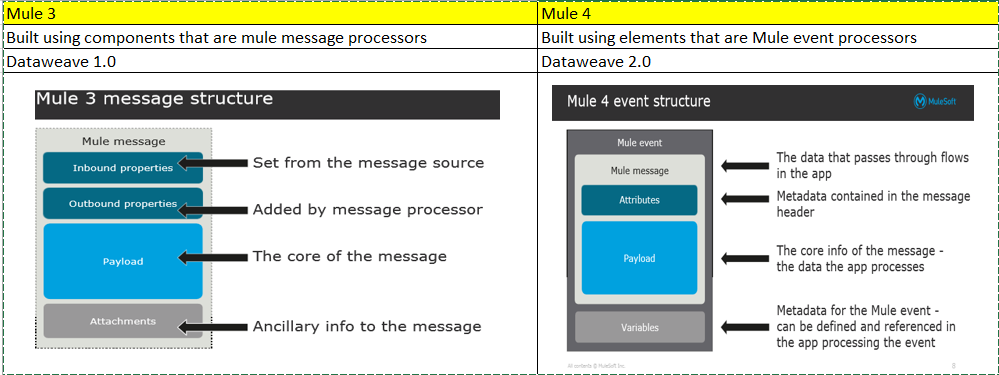
Go to exchange and hit share button to make it publicly available for users.

**Building APIs**

Mule Applications: It accepts and process messages through a series of message processors plugged together in a flow.

Mule applications are deployed to a mule runtime.

A typical flow has a message source(accepts a message from external source triggering the execution of flow), message processors(Transform, filter, enrich and process the message).



Application Building Blocks:

Connectors – Connect to external resources

Scopes – Wraps snippets of code to define fine grained behavior within a flow

Components – executes specific logic upon a message, including custom logic in java, javascript, python, groovy, ruby

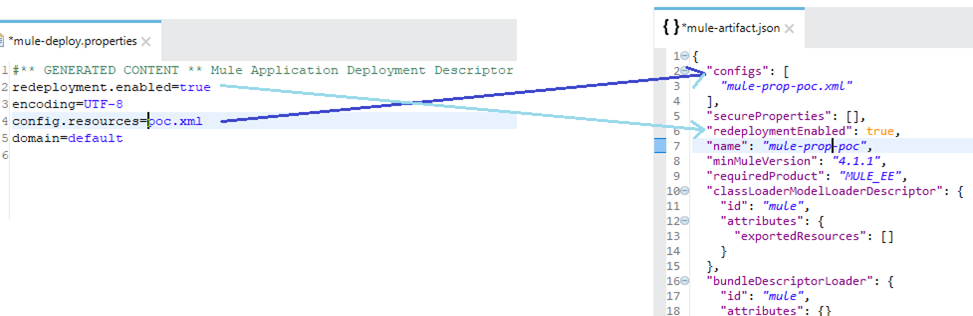
Transformers – Modifies the value of a part of a message

Filters – Limits the processing of message based on set criteria

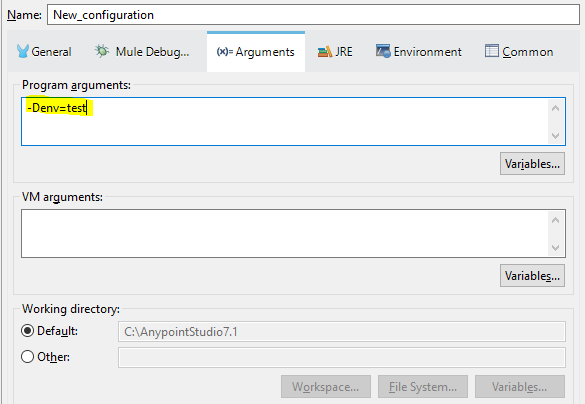
Flow control – Direct messages through different pathways in an application depending upon criteria or content

Error Handling – Handles error during message processing

*Differences between Mule 3 and Mule 4:*

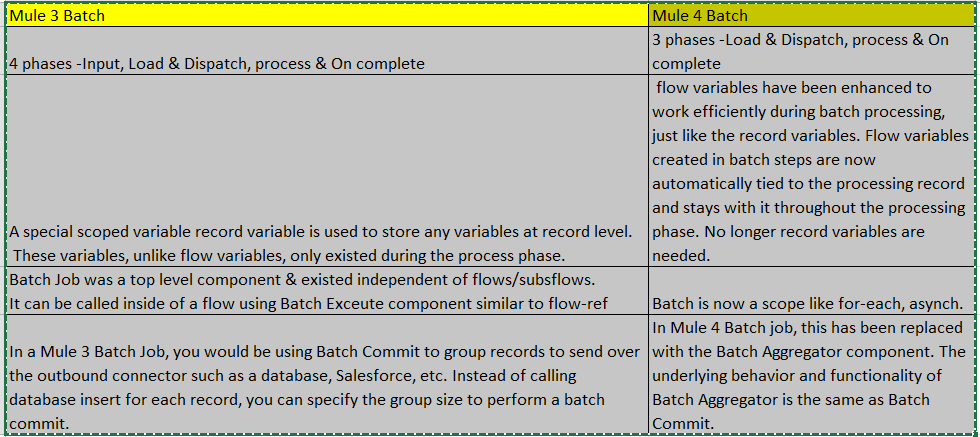
1. Implicit data conversion in Mule 4. No need for any transformations like object to json or object to string> Binary Representation is transparent. This is achieved by Dataweave expression language.
2. Improved Streaming
3. No need for enricher[This is used where the state of the variable is not be changed]. This is achieved by setting a different target. Target can be set in HTTP request component. In other cases, use Transform message to set a different target.
4. Error Handling. New Try block. Catch errors anywhere in the flow.See errors at design time. Re-propagate errors. Java Exceptions aren’t needed. But u can still use them
5. Structure of Mule 4 has changed in terms of packaging. We bundle all connectors in the applications in Mule 3. SO if you need a new bug fix for that connector, you have to update the whole runtime.In Mule 4, connectors are distributed outside runtime. Enables faster access to fixes and enhancements.Reduces risk of upgrade.
6. Choose your own spring version in mule app.Class loader isolates and hides the internals so that they don’t affect you. Just add your spring module in your project/POM
7. Mule SDK – Single way to extend Mule. It interacts with Mule through API. In Mule 3, we had Dev kit
8. Performance - Mule 4 has Self Tuning Runtime Engine[Reactive & non-blocking]. Only 3 pools(CPU Light, CPU Heavy, IO) are used. This eliminates the need for exchange patterns, processing strategies which we have in Mule 3.
9. Flows are collapsible. Also, right click on any component to go its xml, it will take you to the actual component’s line in xml
10. Class loader functions have changed. All default run time will be loaded by different class loaders. All the libraries which you use will be loaded separately by different class loaders. This will give better upgradability path
11. Mule 3 has property placeholder for configuring properties. Mule 4 has Configuration properties for configuring properties
12. **mule-deploy.properties** of Mule3 is replaced with mule-artifact.json. Whenever new Mule flow is created, mule-artifact.json will get updated automatically.

Newly introduced descriptor to Mule application — mule-artificat.json file. This will manage all application and configuration specific settings, Mule version and configuration for class loader.

1. **mule-config.properties** file of Mule 3 has been replaced with configuration.yaml file.
2. Entries of **mule-app.properties** of Mule 3 has been migrated to **Run Configuration -> Argument**section in any point. Also these properties can be configured from the runtime manager console under **Properties**section on cloud hub.**Adding properties Runtime manager console**
3. Mule 3 has inbound properties. Mule 4 has attributes
4. Mule 3 has outbound,session properties, inbound & outbound attachments. Mule 4 has variables

Mule 3 has flow variables, session variables and record variables. Mule 4 just has variables

1. Mule 3 has a poll scope whereas Mule 4 has scheduler
2. Mule 3 has SOAP CXF component. API Kit for SOAP fault, API kit for SOAP Router are the equivalent in Mule 4
3. Batch :



1. Mule 3 uses MEL. Mule 4 uses Dataweave Expression Language. MEL forces you to convert your payloads from binary data (such as XML or JSON documents) into Java objects so that you can write expressions that access that data, for example, when routing to a specific location. In Mule 4, DataWeave is the default expression language. Combined with the built-in streaming capabilities, this change simplifies many common tasks:

* Events can be routed based on payload data, without requiring conversion to Java objects.
* Binary data can easily be queried from an expression anywhere in your flow, for example, when logging.
* Streaming now happens transparently. You no longer need to worry about larger-than-memory data streams or about consuming a stream twice.

*API Kit: Open source tool kit*

This includes :

A router element(routes inbound requests and validates it against the RAML),

A proxy element(validates the request against the RAML and passes the request to the separate API implementation),

APIKit mapping strategy,

APIKit Anypoint studio plugin(to generate an interface automatically from a RAML API definition, Generates a main routing flow(that uses APIKit Router) and flows for each of the method/resource pairs, Processors can be added tp these flows to hook up to the backend logic)

<https://dzone.com/articles/property-file-handling-in-mule4>

**Deploying API**

Before deploying to cloud hub, host has to be changed to 0.0.0.0 from local host. Port has to be 8081 for http. 8082 for https specified using the properties ${http.port}

Cloudhub Load Balancing

https://docs.mulesoft.com/runtime-manager/cloudhub-networking-guide#load-balancing

CloudHub provides a load balancing service for all integrations, you can use the default one or obtain the [dedicated load balancer](https://docs.mulesoft.com/runtime-manager/cloudhub-dedicated-load-balancer).

This service does round robin load distribution across workers, allowing workers to scale linearly as they receive more requests as well as providing transparent switchover when an application is upgraded (see zero downtime upgrades for more information).

Each application deployed on CloudHub has a CNAME record that refers to the load balancer - example: myapp.cloudhub.io. Mule applications deployed on CloudHub must listen on 0.0.0.0 and ports assigned by CloudHub for HTTP and HTTPS. The load balancer then forwards requests from port 80 and 443 (SSL) to these ports on the Mule worker. Forwarded traffic is still HTTP & HTTPS, this means that you can’t listen for HTTPS on ${http.port} nor for plain HTTP on ${https.port}.  
These ports must be referenced using the reserved properties ${http.port} and ${https.port} respectively, so CloudHub services can then dynamically allocate a port at deployment time.

Here is an example of a Mule configuration that utilizes this to expose an HTTPS endpoint:

<http:listener-config name="HTTP\_Listener\_Configuration" protocol="HTTPS" host="0.0.0.0" port="${https.port}" doc:name="HTTP Listener Configuration" > <tls:context name="TLS\_Context\_Custom\_Keystore" doc:name="TLS Context"><tls:key-store type="jks" path="server.jks" keyPassword="keypassword" password="storepassword" alias="cloudhubworker" /> </tls:context> </http:listener-config>

You may also refer to this knowledge-base article: [Sample App: Configuring HTTPS endpoint for deployment in CloudHub](https://support.mulesoft.com/s/article/Sample-App-Configuring-HTTPS-endpoint-for-deployment-in-Cloudhub)

**Important**: On the Mule worker, the CloudHub load balancer proxies port :80 to :8081 for HTTP and proxies port :443 to :8082 for HTTPS.  
The http.port property is automatically set to port 8081 for HTTP, and https.port is set to port 8082 for HTTPS. If other values for http.port and https.port are specified in the mule-app.properties file, these are overwritten at deployment time.

1. **Deploying to cloud hub:** Check the flows locally. Then Deploy to cloud hub. First thing you have to do is to change the host name from localhost to 0.0.0.0. Then right click on the project to Anypoint Platform 🡪 Deploy to cloud hub. Choose the environment and hit deploy. If the deploy is successful, you can see a green circle. You can view the artifacts in Run time manager in Anypoint platform. After deploying, url will be as follows,

<http://apdev-accounts-ws-raajee.us-e2.cloudhub.io/api/accounts/4401>

1. **Creating a proxy :**  **Go** to API Manager in Anypoint Platform. Manage API 🡪 Choose the RAML that’s published in exchange. Before that go to design center in Anypoint and add the base uri in the RAML file which you are going to choose and publish that version to exchange.

**BaseUri should be something like** <http://apdev-accounts-ws-raajee.us-e2.cloudhub.io/api/>

Now you can g**o** to API Manager in Anypoint Platform. Manage API 🡪 Choose the RAML that’s published in exchange. Then provide all the details and deploy. Now a proxy is created with the provided name. Now the Gateway url will be something like this,

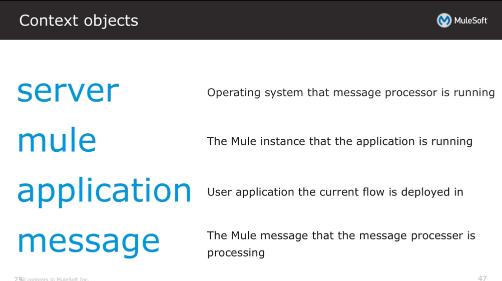
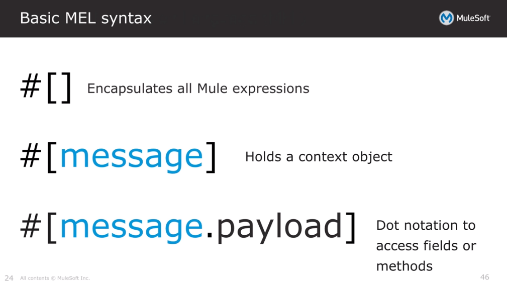
<http://accounts-api-raajee.us-e2.cloudhub.io/accounts/4401>

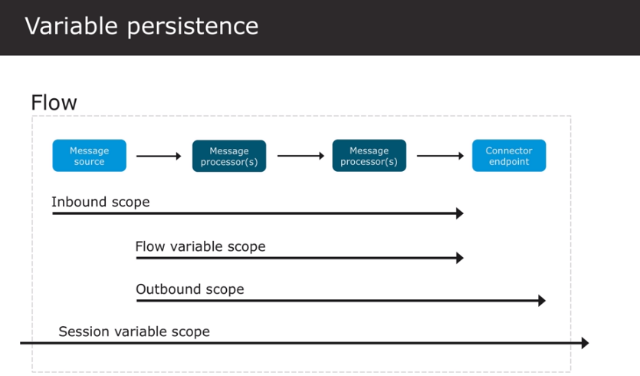
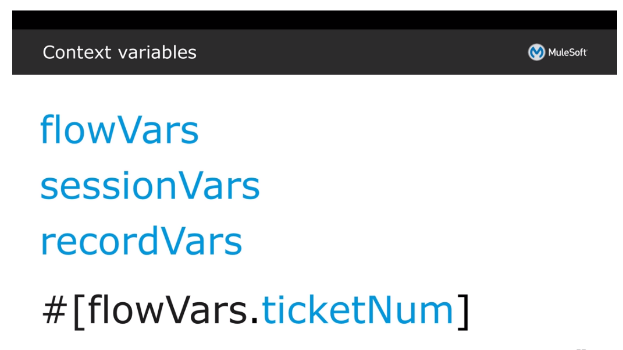
1. Now we can add policy and SLA tier using the API Manager. When SLA of manual type is added, you have to request access to the API by going to exchange and request access by clicking on the API. Then this request has to be approved by going to “my applications” in exchange
2. Go to API Manager, click policies to get the RAML snippet to add the client\_id and client\_secret to RAML file. Then go to design center add the traits snippet to RAML file. Then publish to exchange and make a note of the version. Then go to API manager choose the API and change the API specification under actions in the top right.
3. Go to exchange and click my applications, choose your application and copy the client id and client secret. Now the API can be accessed

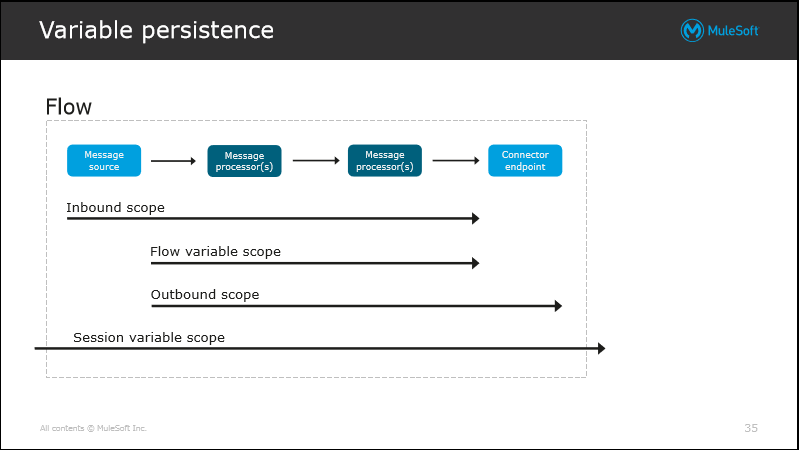
**Accessing & Modifying Mule Messages:**

**Datasense:** Is ANypoint studio’s ability to proactively discover metadata from internal and external resources.

Mule 3

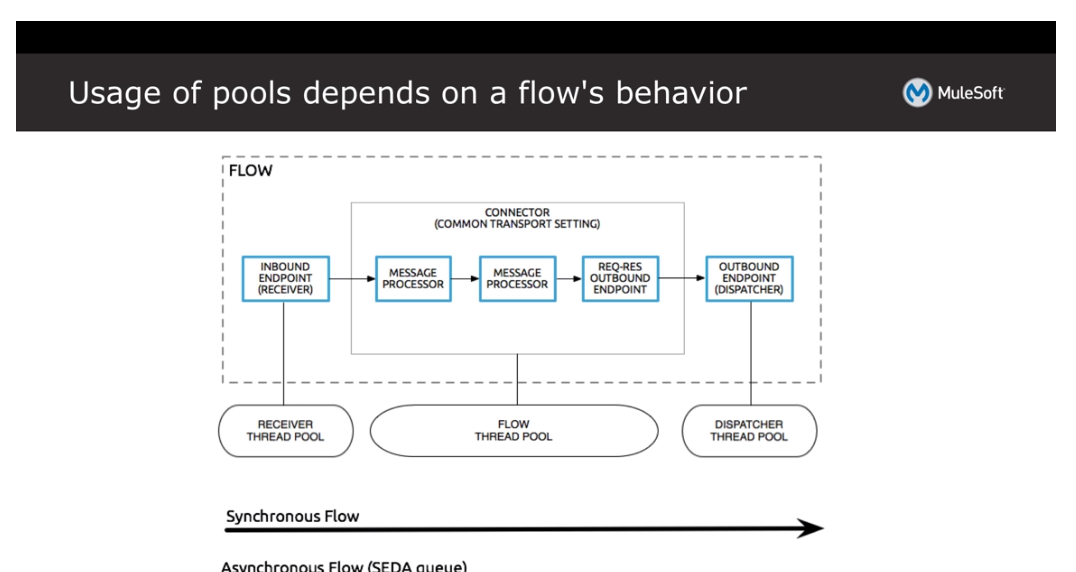
****

****

****

**Structuring Mule Application**

Flows use 3 thread pools: Receiving, Flowprocessing & Dispatching

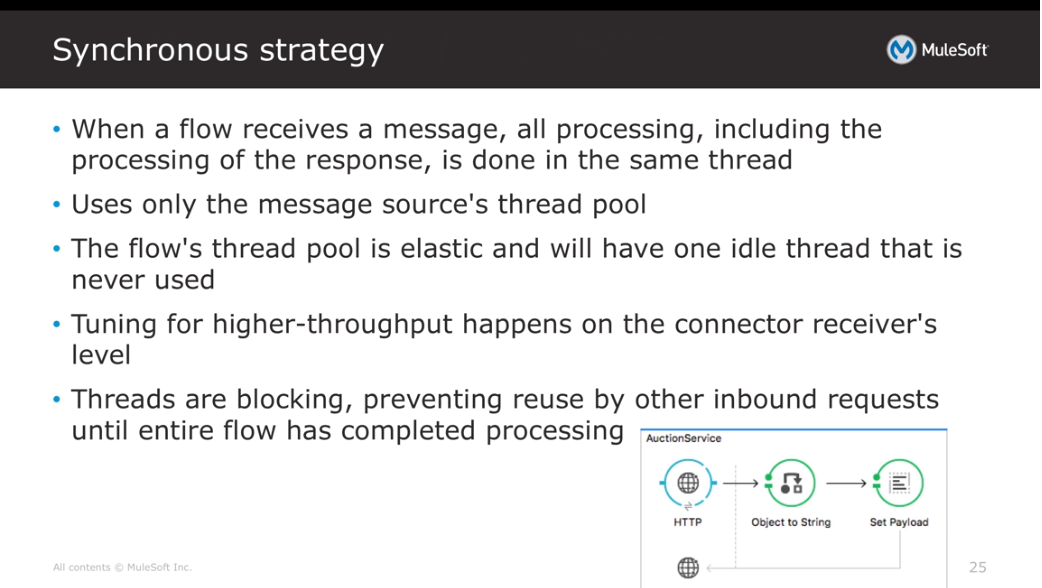
****

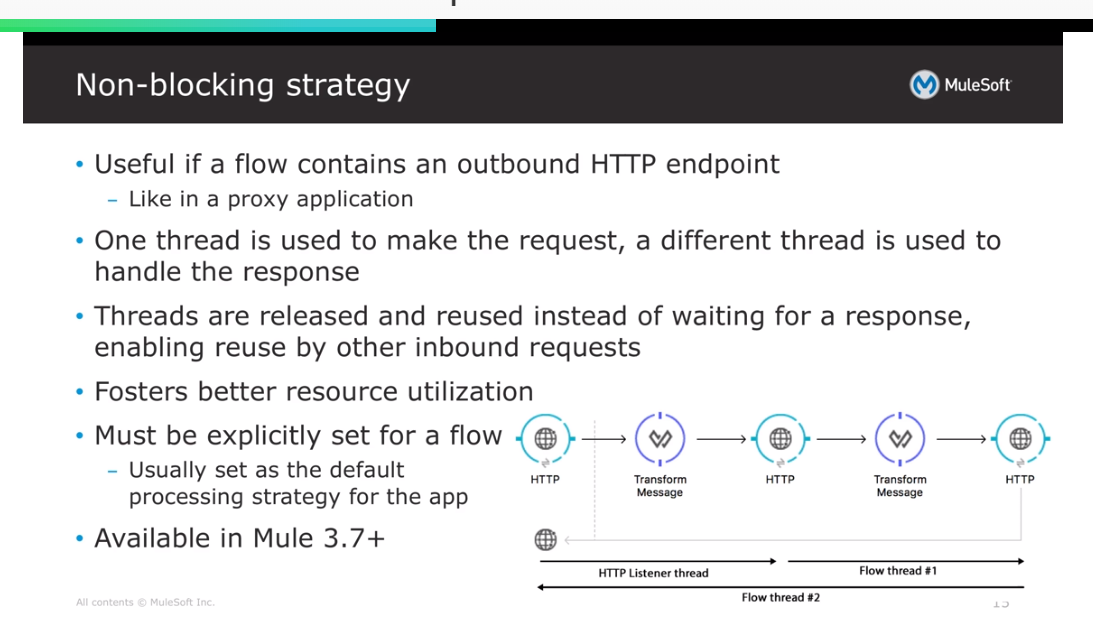
Architecture followed by Mule is SEDA

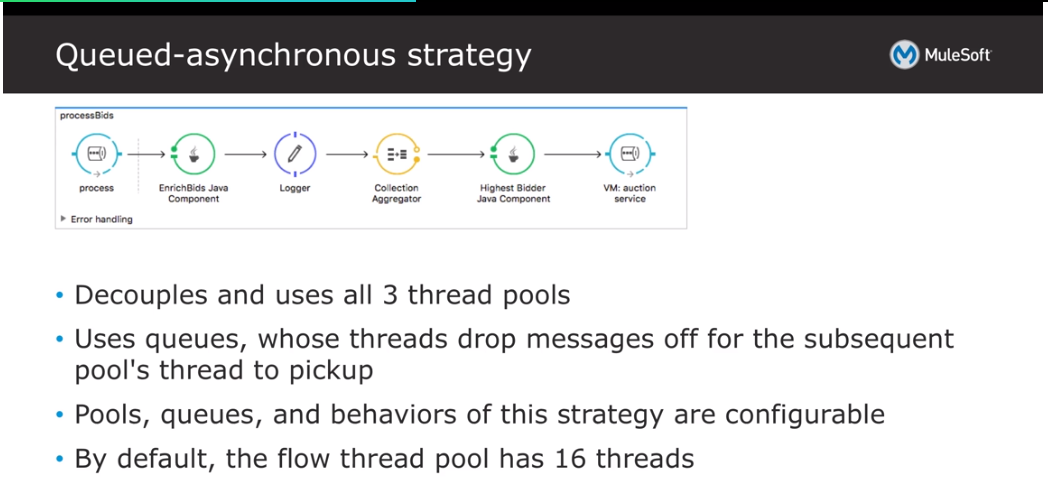
**SEDA – Staged Event Driven Architecture**

1. Decouples receiving, processing and dispatching phases.
2. Supports higher level of parallelism is specific stages of processing
3. Allows for more specific tuning of areas within a flow’s architecture

*Processing Strategies:*

**

**

**

Before Mule 3, VM transport was needed to pass a message from one flow to another. In Mule 3, Flow Reference was added to let flows directly reference one another without a transport in the middle.

VM transport is now used to

* Achieve higher levels of parallelism in specific stages of processing
* Allows for more specific tuning of areas within a flow’s architecture
* Calls flows in other application that are in the same domain

*Variable scopes:*

Flowvars, session vars will not cross the original flow where it is set. When it crosses the HTTP transport, it will not be available in the flow which is called by the HTTP request component, but once the control comes back to the original flow where the HTTP request component is present, variables are seen again.

Flowvars will not cross the original flow where it is set whereas session variables will. When it crosses the VM transport outbound, session variables will be available in the flow[flow vars will not be available] which is called by the VM transport outbound component, but once the control comes back to the original flow where the VM transport component is present, variables are seen again.

Flow Variables persist through all flows unless the message crosses a transport boundary like VM or HTTP

Session variables persist across some but not all transport barriers(VM yes, Http NO)

<https://dzone.com/articles/scope-of-session-and-flow-variables>

**Consuming Web Services**

Project Name : apdev-flights-ws

Workspace : C:\Users\stkra\Documents\Mulesoft\workspace\apdev-flights-ws

REST call : HTTP REST Component

SOAP call : Web service Consumer Component, CXF

**Handling Errors**

Types of exceptions : System Exceptions(Exceptions when the application starts or when the connection to the external system fails), Message Exceptions(Exceptions within the flow)

Mule 3 has catch exception strategy, choice exception strategy, rollback exception strategy

Mule 4 has On error propagate & On error Continue

<https://blogs.mulesoft.com/dev/mule-dev/how-to-error-handling-mule-4-beta/>

**Writing Dataweave Transformations – Refer Dataweave Transformation slide**

Project Name : apdev-flights-ws

Workspace : C:\Users\stkra\Documents\Mulesoft\workspace\apdev-flights-ws

**Connecting to Additional Resources**

Project Name : apdev-examples, apdev-jms-connector

Workspace : C:\Users\stkra\Documents\Mulesoft\workspace\apdev-examples

Salesforce Connector:

1. Handles all 5 of the integration patterns for connecting with Salesforce
2. Can perform operations that salesforce exposes via 4 of their APIs

File Connector:

Poll: Replaced by Scheduler in mule 4

JMS Queues: Install ActiveMQclient and start to access it from your machine locally

<http://localhost:8161/admin/>

Admin/admin

JMS Messaging Models :

Queues(Point to point) – One to One; A sender sends the message to the Queue and a single receiver pulls the message off of the queue; The receiver need not be listening to the Queue at the time of the message is send.

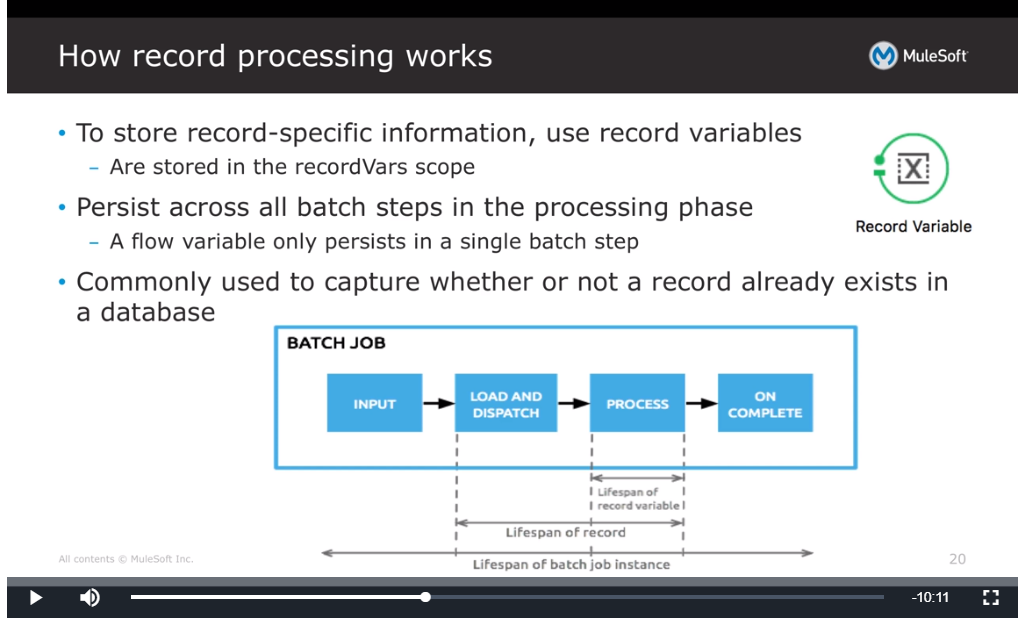
Topics(publish/Subscribe) – One to Many; A publisher sends a message to the topic and all active subscribers of the topic receive the message; Subscriber that are not actively listening to the topic will miss the published message unless the messages are made durable

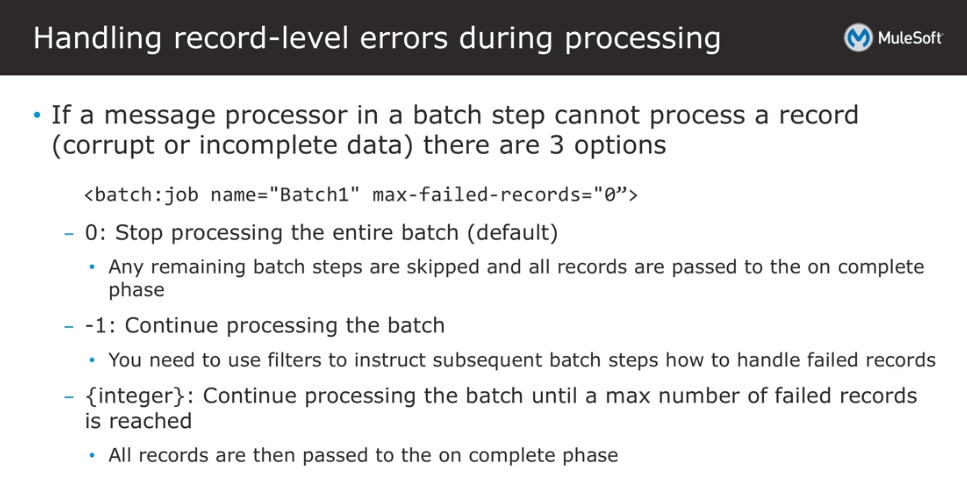
**Processing the Records:**

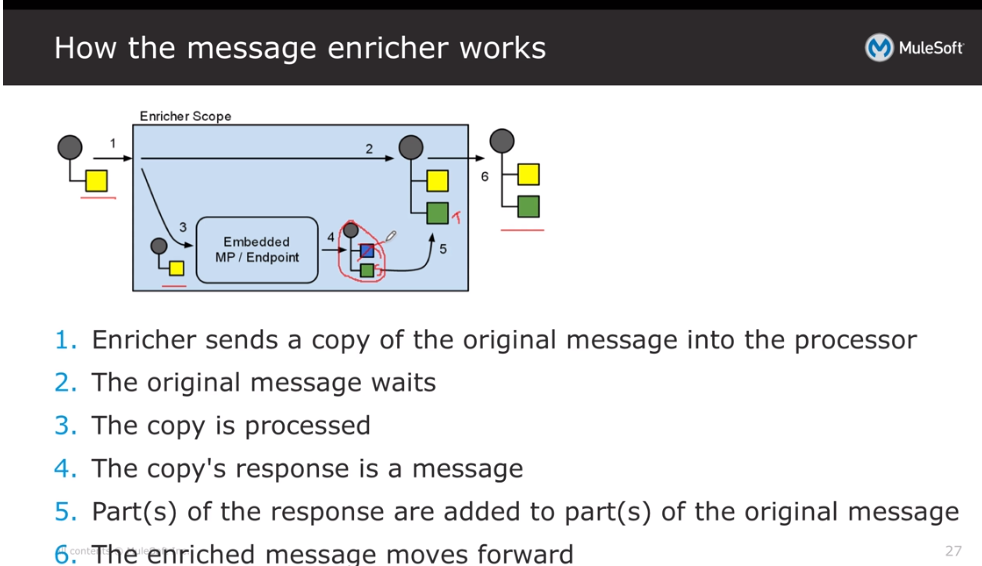
Project Name : apdev-batch

Workspace : C:\Users\stkra\Documents\Mulesoft\workspace\apdev-batch

Batch Phase : Input(optional),load & dispatch(in built), process records(required),On complete(Optional phase- Payload is a BatchJobResult, which has the properties statistics including loaded records, processed records,successful record, failure records and total records)







Reference URLs:

Connect 2018 : Mastering Mule 4 and Studio 7 <https://videos.mulesoft.com/watch/gtWiNRFapPb9MAS4bxfevy>

Batch <https://dzone.com/articles/batch-processing-in-mule-4-vs-mule-3>

<https://dzone.com/articles/property-file-handling-in-mule4>

<https://docs.mulesoft.com/mule4-user-guide/v/4.1>

1. Why Mule soft ? To connect 4 resources, we need 6 connections(n(n-1)/2). To avoid these complexities, we went ESB(Enterprise Service Bus). Mule, is a runtime engine of Anypoint Platform, which is Java-based enterprise service bus (ESB) and integration platform that allows **developers** to connect applications together quickly and easily, enabling exchanging of data.

2. Since Salesforce acquired and it provides a connector to integrate with Salesforce, Does mulesoft provide connectors for any other CRM vendors such as Microsoft Dynamics/SAP or any others ? – There are connectors. They can be got from exchange.

3. Can I use .NET run time as a custom component in Mulesoft ? Can I use any other programming language platform runtime in Mulesoft cloud ? We can create a .net component and cone

4. Capabilities of Mule:

* **Service creation and hosting —** Mule as an ESB can expose and host reusable services using the ESB as a lightweight service container
* **Service mediation —** it can shield services from message formats and protocols, separate business logic from messaging, and enable location-independent service calls
* **Message routing —** Mule as an ESB routes, filters, aggregates, and re-sequences messages based on content and rules
* **Data transformation —** it exchanges data across numerous and varying formats and transport protocols
* **APIs**
  + Build new APIs quickly, design new interfaces for your existing APIs, and simplify API management
  + Rapidly expose valuable data to mobile devices, web apps, and connected devices in a secure and controlled way
  + Enable and empower the entire organization with Experience APIs, Process APIs, and System APIs
* **B2B**
  + Modernize B2B by extending Mulesoft’s API-led connectivity approach to B2B and EDI
  + Build reusable services across multiple trading partners and B2B processes
* **Data Integration**
  + Combines batch and real-time processing for unified application and data integration
  + Offers a template-driven approach to development
* **DevOps**
  + Streamlines adoption of popular DevOps frameworks for continuous integration and efficient deployment
* **ESB**
  + Combines the power of data and application integration across legacy systems and SaaS applications, with a seamless path to benefitting from other capabilities in the Anypoint platform and API-led connectivity
  + Quickly build integrations ranging from simple to advanced with pre-built connectors and templates,
* **Internet of Things (IoT)**
  + Connect and orchestrate data from your enterprise and the cloud to devices at the edge of your network—including point of sale systems, medical devices, sensors, and more—using open standards, developer-friendly tools, and out-of-the-box transport protocols.
* **Microservices**
  + Enables your organization to develop new solutions in a manageable, resusable, and governed way
* **Mobile**
  + Enables fast, easy, and governed mobile access to any data from backend systems, legacy databases, and SaaS Applications

5.What are the major differentiators to go towards Mulesoft ? In other words, What are the technical drivers to go towards Mulesoft ?

* **Lightweight** — Mule as an ESB is the most [lightweight](https://www.mulesoft.com/platform/api)[integration platform](http://www.mulesoft.com/platform/enterprise-integration) available. The fully loaded distribution is 40 MB. It’s modular by design so you can remove unneeded modules to reduce its footprint. But being “lightweight isn’t just about size; it also encompasses the time, effort, and financial cost of making changes to existing integrations. The Mule runtime has modularization, fast deployment, and a configuration model that makes it simple to re-order, add, or change functionality.
* **Doesn’t only provide mediation** — many vendors think of an ESB as only a mediation engine between systems and therefore have separate products for hosting business logic and publishing services. This creates unnecessary complexity. Instead, the Mule runtime provides a light and scalable service container for publishing REST and SOAP services. And, since Mule integrates tightly with Spring, developers can also leverage the capabilities of Spring to implement business logic.
* **Accessible** — Mule as an ESB uses common tools that all Java developers are familiar with, like Maven, Eclipse, JUnit and Spring. It utilizes an XML configuration model (similar to Spring) to define logic. Custom code can be written in numerous languages, including Java, Groovy, JavaScript, Ruby or Python. In addition, [Anypoint Studio](https://www.mulesoft.com/platform/studio) helps new developers get up to speed quickly with a graphical development environment.
* **Scales up and scales down** — Mule as an ESB was designed for horizontal scale on commodity hardware. Mule’s runtime is easily embeddable into an application, and can also be embedded in your application server such as Tomcat, JBoss or WAS or directly in your application. Mule as an ESB also provides JUnit support so that it can be embedded in a JUnit test case. This is important because you can create repeatable unit tests for integrations that can run on a developer laptop and can be incorporated into a continuous build.
* **Message agnostic** — Mule as an ESB does not force XML messages on its users. While XML is commonly used, there are many scenarios where you will want to use JSON, flat files, Cobol Copybooks, binary and file attachments, streams, and Java objects. Mule streaming allows developers to process large messages efficiently.
* **Can be deployed on-premises or in the cloud** — there are numerous advantages to deploying your applications in the cloud, and there are advantages to deploying on-premise, or a [hybrid](http://www.mulesoft.com/resources/esb/hybrid-cloud-integration-solutions) of the two. Mule as an [ESB](http://www.mulesoft.com/platform/soa/mule-esb-open-source-esb) can accommodate any of these approaches. And whether you are are deploying on-premise or the cloud or in a [hybrid](http://www.mulesoft.com/resources/esb/hybrid-cloud-integration-solutions) fashion, there are no new concepts to learn and the developer experience is identical.

6) What are the different licensing requirements for Mulesoft ? ex, Is it per user based or subscription based on features utilized? Understand more details around it.

Mule ESB Community – Free

Mule Studio – Free

Mule ESB Enterprise – 30 days trial in Pre-prod & In prod, subscription based