->Business world is changing fast, migrating to cloud, mobile, IOT support. Companies want to adopt new technologies quickly.

->In Traditional COE (Center of Excellence) model, business users having requirement asks the COE to implement that requirement. A CoE's responsibilities include running an API platform, developing APIs on that platform, and managing the operation of those APIs.

->This model may have historically worked well; but, in today’s digitally disruptive environment, the CoE model presents a number of challenges like

creates a delivery bottleneck for business users and is subject to scalability challenges.

->Leads to inefficiency due to the time it takes to gather requirements. There is also no focus on reusing code and development across projects—making each project as time-consuming and difficult as the next.

->Prevents consumers from experimenting, because they do not have access to underlying business data and processes through APIs. As a result, they cannot embrace a fail fast model.

**Now businesses are transitioning from COE** (Center of Excellence) **to C4E (Center for Enablement) to address issues with COE model .In C4E model , it enables delivery teams to design and build productized APIs in a decentralized way and promote reuse through an API ecosystem -improving efficiency and accelerating innovation**

In C4E model

**Focus on getting API design right before investing in building it. After creating API specification document publish it to the exchange and let users review it, test it and give feedback and then Incorporate the changes based on their feedback.**

**Instead of building bulk API’s, modularize them for reusability**. MuleSoft divides API's into 3 categories (system APIs, Process APIs, presentation API's) and use API fragments when building an API to further reuse the components.

->**API fragments are reusable components that we can create and publish to exchange in order to be consumed across different API's in our organization, we could also have fragments within our API specification to improve modularity of your API spec.**

->Fragments can be data types, examples, traits (Traits--It's a block of code in your API definition document, Example -description, headers, query string parameters, and responses…),…

**API Development Life cycle**:

1)**Designing API's**

->Focus on getting API design right before investing in building it

->In **Any point platform->Design Center->'Create API Specification'** using RAML language (RAML is 'Restful API modeling language, it can auto generate API documentation and can generate mocked endpoints for API, so an API can be interactively tested before it is built)

->Basically, will define resources and nested resources, methods under resources

->It's very important that we **modularize** the API's (**Instead of including all code in one RAML file, modularize it and compose it of reusable fragments**)

->**API fragments** are reusable components that we can create and publish to exchange in order to be consumed across different API's in our organization, we could also have fragments within our API specification to improve modularity of your API spec.

->**Fragments** can be **data types**, **examples**, **traits** (Traits--It's a block of code in your API definition document, Example -description, headers, query string parameters, and **responses**…),…

**Trait**: When creation of record is successful, API will return response with HTTP status code->underneath ->body->underneath response type (application/json)->underneath->an example payload

->Use Mocking service to test your API, it generates a mock endpoint url

->Publish '**API specification**' to **exchange (**Click on '**Publish**' button ->'**Publish to Exchange**' in API Designer)

->Locate your API in 'Any point exchange'-->Click on it and review the default generated "**API portal**" and do needful changes

->**Click on 'Share' button to share** to share your API spec to targeted users for review (specific users based on role or to public)

The API Designer 
O) API Manager 
File 
Browser 
ISFO, LAX, 
DOCS 
Editor 
Shelf 
Resources 
DESCRIPTION 
URI PARAMETERS 
API 
Console 

->Any point exchange is the library of connectors, templates (ex: RAML, WSDL,API fragments, examples…)

2)Building API's

->In Any point studio create Restful interface automatically from a RAML API definition with APIKit plugin

->Create a new flow having actual implementation to connect to backend systems and do needful transformations

->In first flow use 'flow reference' component to connect to implementation which has backend logic

->Create one more flow to place all **global elements** that are shared or referenced across different xml files (like connector configurations (d/b, HTTP listener…)

->We can test the webservice in **APIKit console view** or from any **REST client like POSTMAN**

**->Environmental variables** are maintained in **'mule-project.xml' file**

**->**Where do you configure/maintain external parameter values **--->mule-deploy.properties** file

->

3)**Deploying and Managing API's**

Now we need to create a proxy and select API from exchange and configure implementation URL (we get this from Runtime manager) and deploy to cloud hub

We create proxy, because we don't want to expose API endpoint URL directly to end user, instead will share proxy end point URL, proxy internally makes a call to API.

We restrict access to API by writing policies and SLA tiers to Proxy

-->Now deploy your mule application from any point studio to cloud hub

->Right click on the project->Any point platform->Click on '**Deploy to cloud hub**'->Select the **environment** in which you want to deploy->Give the **unique 'Deploying application name'**->Select the **'runtime version**' and **'worker size**' and click on '**Deploy application**' button

Runtime 
Runtime version 
properties 
Worker size 
0.1 vCores 
Insight 
Logging 
Workers 
Static IPS 

->We can see the **deployment status** and **logs** in Any point platform->Manage center->Runtime manager

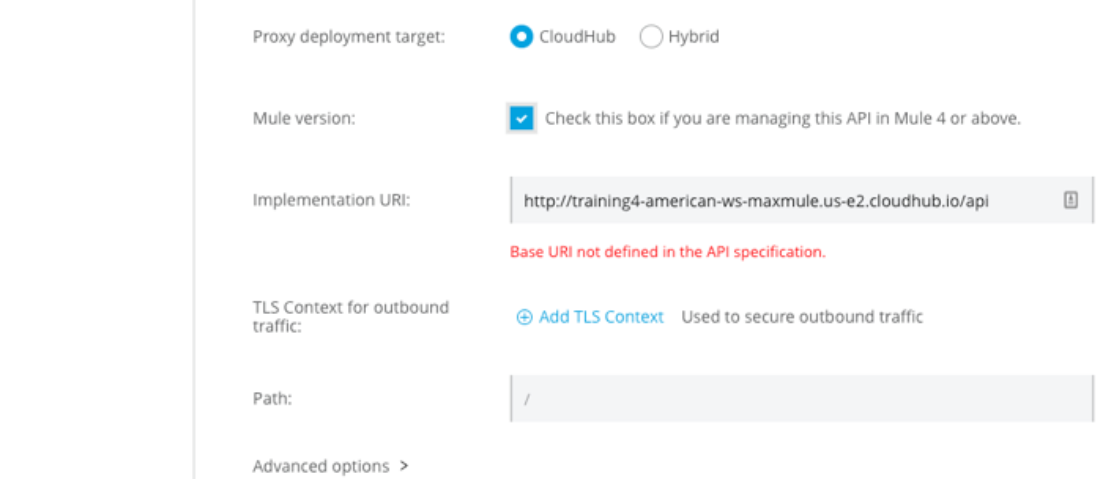
->Now **create and deploy Proxy Application**

->Any point platform->Management center->API manager->Click on 'Manage API' button and select 'Manage API from Exchange'

API Manager 
API Administration (Sandbox) 
SANDBOX 
API Administration 
Manage API v 
Manage API from Exchange 
Manage an Exchange asset 

->Select your API and configure implementation URL(We can get this from 'Runtime Manager') ,proxy application name and version and deploy to 'cloud hub'.

Machine generated alternative text:
API Manager 
API Administration (Sandbox) Get from Exchange 
SANDBOX 
€- API Ad"'.mstra'eon 
Manage API from Exchange 
API Configurations 
API name: 
Asset type: 
API version: 
Asset version: 
Managing type: 
American Flights API 
RAMUOAS 
VI 
1.0.1 
C) Basic Endpoint 
View API in Exchange 
O Endpoint with Proxy 



->We restrict API access with **policies** and

Ex: SLA based rate limiting policy

Client ID enforcement

API 
American Flights API 
There are no apVied for se•cted 
API 
Rate . 