

Introduction

Welcome to NJC Labs Architecture Accelerator program. This is one of the carefully crafted unique training programs that enables you to become an Integration Architect quickly, without compromising on the core skills that are crucial to embark on a successful career in the software industry.

This training program is an outcome of years of expertise in enabling senior software engineers of our clients and partners to overcome technological challenges, by rapidly ramping up the required skills, so that they are enabled to drive software projects from inception to production, delivery and maintenance, independently.

NJC Labs Architect accelerator program is designed to enable a senior developer or architect to master the integration and enterprise architecture concepts that are required to kick start on a project from day one and deliver customer success.

A Detailed description of the program is described below. The entry to Architecture Accelerator program is based on a preliminary assessment, which would be shared along with brochure.

We want you to complete this assessment to assign you to a custom accelerator program level catering to your needs.

## 

## An Overview of Accelerator Program

Integration & Enterprise Architecture is a broad field both in depth and breadth. The core skills required to be successful in each of these domains are different. For instance, to be successful an Integration Architect needs to have a broader understanding of software systems, architectures, protocols, data structures and patterns to connect systems and applications, where as an enterprise architect will make recommendations for adjusting policies and projects to achieve target business outcomes that capitalize on relevant business disruptions. EA is used to steer [decision making](https://en.wikipedia.org/wiki/Decision_making) toward the evolution of the future state architecture.

Our preliminary skill assessments are aligned to identifying the basic skills needed to participate in these architecture accelerated programs.

Based on your technical background, experience, knowledge and expertise, we may be able to guide you through the relevant skills that you need to pick up to be successful working on enterprise software projects.

Different modules of the accelerator program will help you deliver enterprise integration projects successfully. Through each module we want you to be a master in the technology and concepts so that you can deliver projects.

Through every module we will introduce you to a concept and then the candidate will perform an exercise to implement the concept. This will be reviewed by the mentor of the program to see the quality of the output.

## 

## Module 1: Architecture

## Introduction - whats my-trains project

This is an IP (Intellectual Property) project of NJC to alert subscribers of the train service to notify any delays in train service. This project uses the web services provided by a train service provider and uses the delay notifications through APIs or web services. my-trains project will use this information to identify delays and send notifications to customers subscribed for alerts. Refer the use case document to know more about the details.

## Architecture Concepts

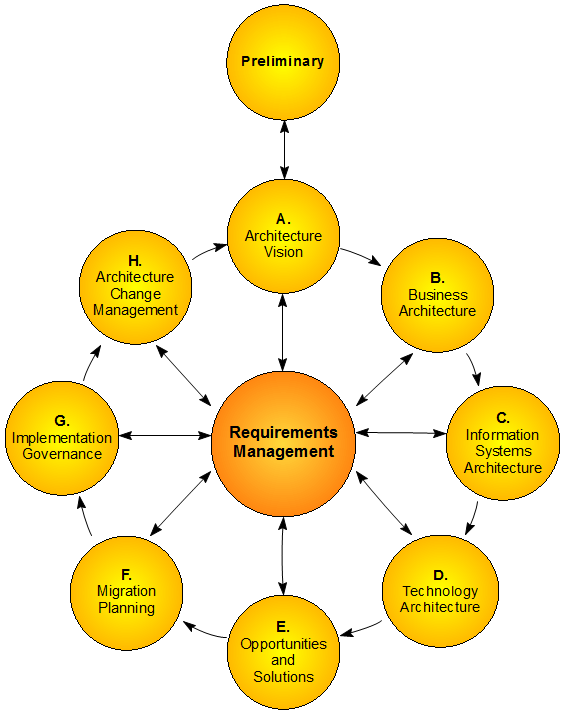
## Enterprise Architecture

Long term strategy and vision of the organization is achieved using enterprise architecture. To achieve this vision, enterprise architects identify multiple IT projects for next few years. MuleSoft proposes to use API Led connectivity architecture to build an application network, which is the state of enterprise architecture of an organization.

## TOGAF / Zachman / 4+1 Viewpoint

**TOGAF**:

The TOGAF standard is an architecture framework. It provides the methods and tools for assisting in the acceptance, production, use, and maintenance of an Enterprise Architecture. It is based on an iterative process model supported by best practices and a re-usable set of existing architecture assets.



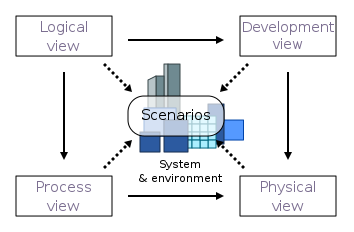
**Zachman Framework:**

The framework: descriptive representations The framework is a matrix, consisting of six columns and six rows, which are created in combining two idea : x Representations: rows, representing the different perspectives on a product of the different participants; x Descriptions: columns, describing the product with different types, for different purposes, in different ways.



4+1 View point:

4+1 is a view model used for "describing the architecture of software-intensive systems, based on the use of multiple, concurrent views”. The views are used to describe the system from the viewpoint of different stakeholders, such as end-users, developers, system engineers, and project managers. The four views of the model are logical, development, process and physical view. In addition, selected use cases or scenarios are used to illustrate the architecture serving as the 'plus one' view. Hence, the model contains 4+1 views.

****

Logical View:

The logical architecture primarily supports the functional requirements — what the system should provide in terms of services to its users.

Process View:

The process architecture takes into account some non-functional requirements, such as performance and availability. It addresses issues of concurrency and distribution, of system’s integrity, of fault-tolerance, and how the main abstractions from the logical view fit within the process architecture

Physical View:

The physical architecture takes into account primarily the non-functional requirements of the system such as availability, reliability (fault-tolerance), performance (throughput), and scalability. The software executes on a network of computers, or processing nodes (or just nodes for short). The various elements identified — networks, processes, tasks, and objects—need to be mapped onto the various nodes.

Developer View:

The development architecture focuses on the actual software module organization on the software development environment. The software is packaged in small chunks — program libraries, or subsystems — that can be developed by one or a small number of developers.

Scenario View:

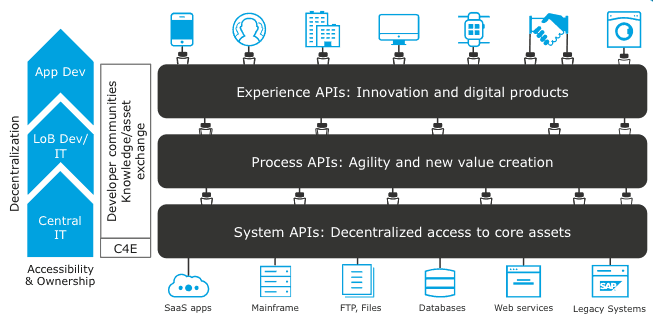
The elements in the four views are shown to work together seamlessly by the use of a small set of important scenarios —instances of more general use cases—for which we describe the corresponding scripts (sequences of interactions between objects, and between processes).

## API Led Connectivity / Application Network

API-led connectivity is a methodical way to connect data to applications through reusable and purposeful APIs. These APIs are developed to play a specific role – unlocking data from systems, composing data into processes, or delivering an experience.

Large enterprises have complex; interwoven connectivity needs that require multiple API-led connectivity building blocks. In this context, putting in a framework for ordering and structuring these building blocks is crucial. Agility and flexibility can only come from a multi-tier architecture containing three distinct layers:

1. System Layer
2. Process Layer
3. Experience Layer

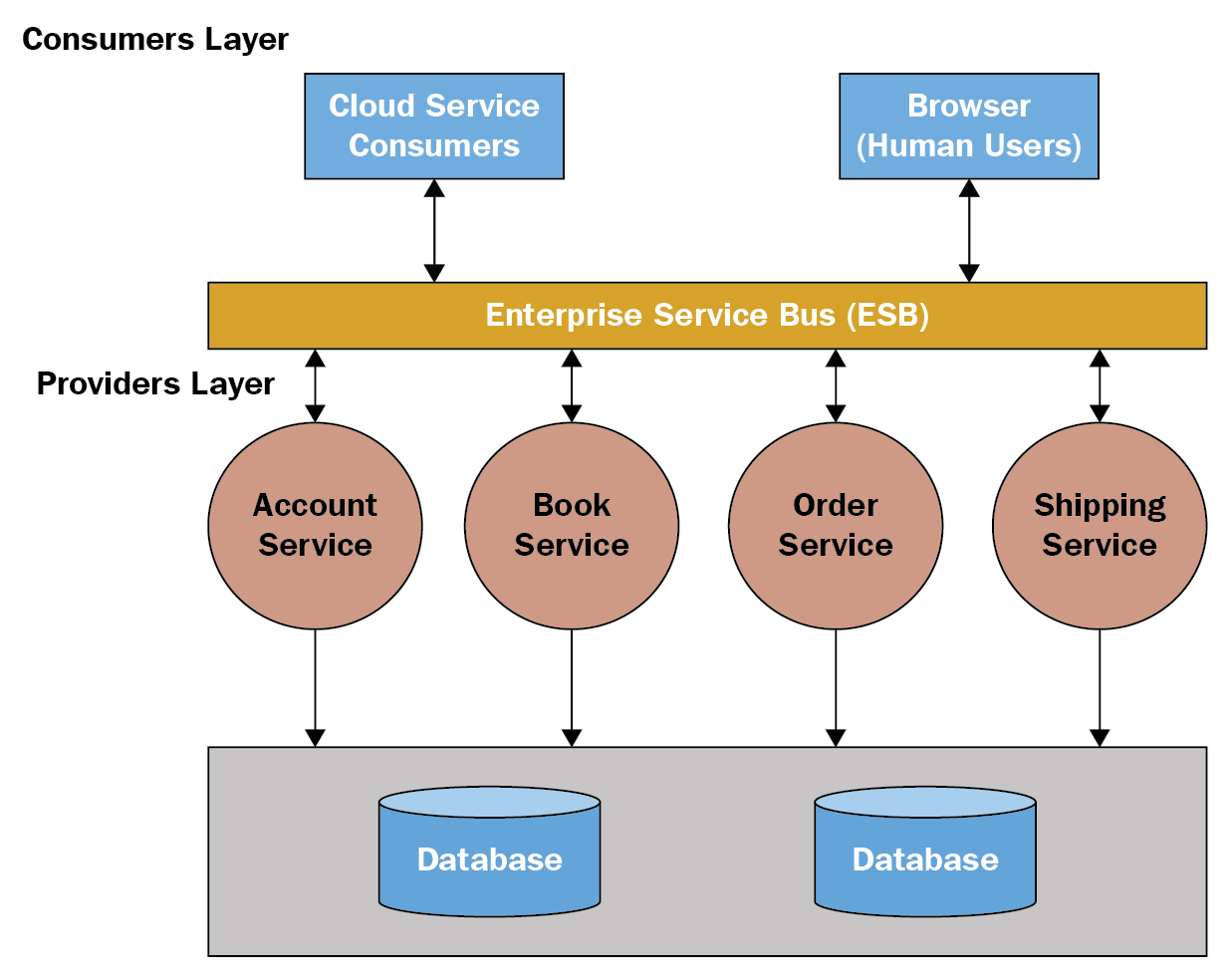


## Integration Architectures

Long term strategy and vision of the organization is achieved using enterprise architecture. This will usually contain multiple projects, like file transfer, ETL based, batch etc. This project specific architecture is Integration Architecture.

## SOA

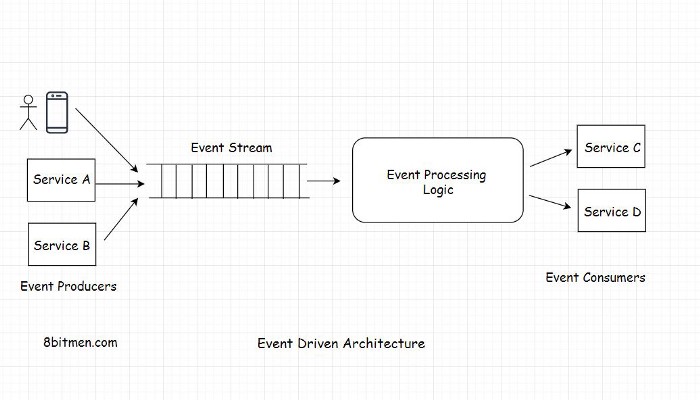
Service-oriented architecture (SOA) is an enterprise-wide approach to software development that takes advantage of reusable software components, or services. Each service is comprised of the code and data integrations required to execute a specific business function.



## EDA

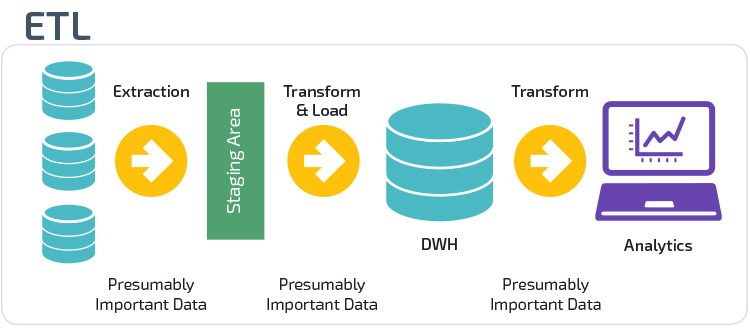
Non-blocking architecture is also known as reactive or event-driven architecture. Event-driven architectures are pretty popular in the modern web application development.

They’re capable of handling a big number of concurrent connections with minimal resource consumption. Modern applications need a fully asynchronous model to scale. These modern web frameworks provide more reliable behavior in a distributed environment.



## ETL

* ETL entails synchronizing data via background processes on a predefined schedule. This data can be pushed or pulled. Only backend ETL processes need database access. It is asynchronous, meaning services can execute without waiting for a “callback.”
* This integration pattern also hides implementation details nicely. It provides reasonable decoupling because the services are not dependent upon one another’s uptimes. Live users don’t get affected by the uptimes or the processing time.
* The ETL processes have to change with the source and destination databases. With ETL integrations, data consistency depends on the schedule and duration. Figuring out the change delta could get too complicated. In these situations, the teams fall back on pushing the entire dataset out. That makes processes very long-running, significantly undermining their usefulness.
* Reporting services are a natural fit for this type of integration. These processes have their place but usually get very involved with time. They should be used only when the stale data is acceptable in the system.



## Delivery Approach

## Catalyst Program

Catalyst program is a unique delivery method proposed by MuleSoft to make sure customer projects are successful. There are 3 focus areas in the catalyst program: Business Outcomes, Technology Delivery and Organizational Enablement.



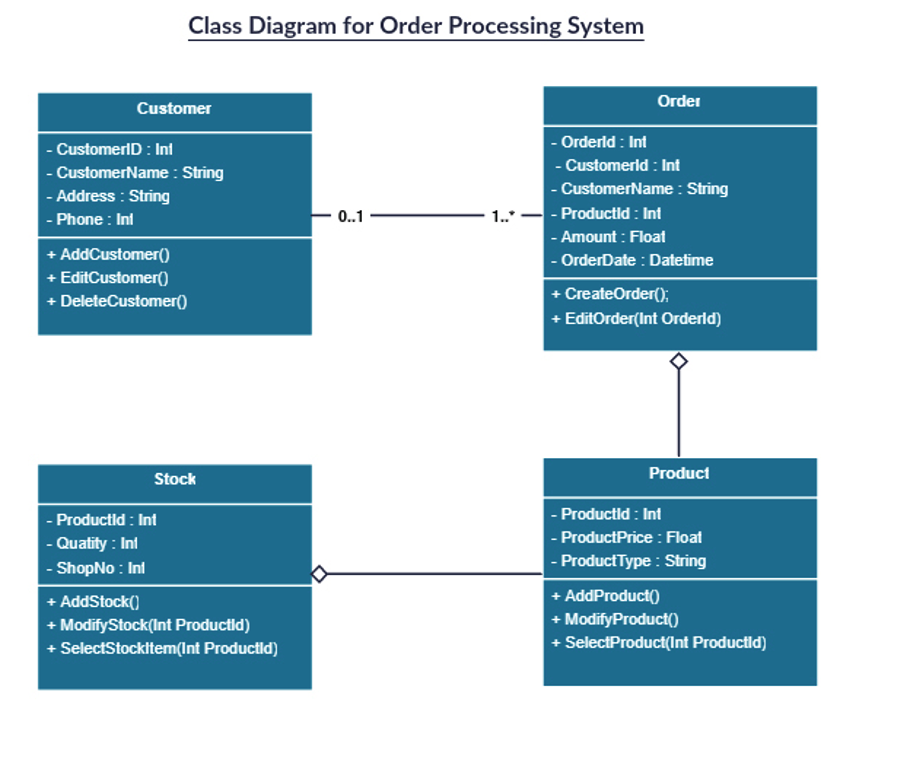
## Diagramming Tools

## Archimate

The Open Group created ArchiMate as an open and independent modelling language for enterprise architecture (EA). The standard provides a notation to enable enterprise architects to describe, analyze, and visualize the relationships among business domains in an unambiguous way. ArchiMate enables the creation of fully integrated models of the organization’s EA, the motivation for it, and the programs, projects and migration paths to implement it. ArchiMate consists of the core layers, and as well as the extensions for modeling the motivations for the architecture, and its implementation and migration planning.

## UML

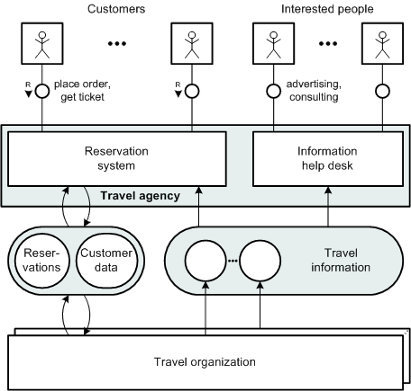
Unified Modeling Language, is a standardized modeling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing object oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software



## FMC

High level / Big picture / context view diagram

The Fundamental Modeling Concepts (FMC) primarily provide a framework for the comprehensive description of software-intensive systems. It is based on a precise terminology and supported by a graphical notation which can be easily understood. Modeling we call the intellectual activity of creating a model of some system with the goal to capture its essential structures necessary to understand its (existing or planned) behavior (internal and to its environment) and to describe these structures in a comprehensive way.



## Exercise

## Requirement Analysis

Go through the requirements and come up with user stories of my trains project.

User Story 1:

As an End user, I need to provide username, password, phone, email so that I can register for MyTrain notification service and I should be able to get authenticated for all further API calls.

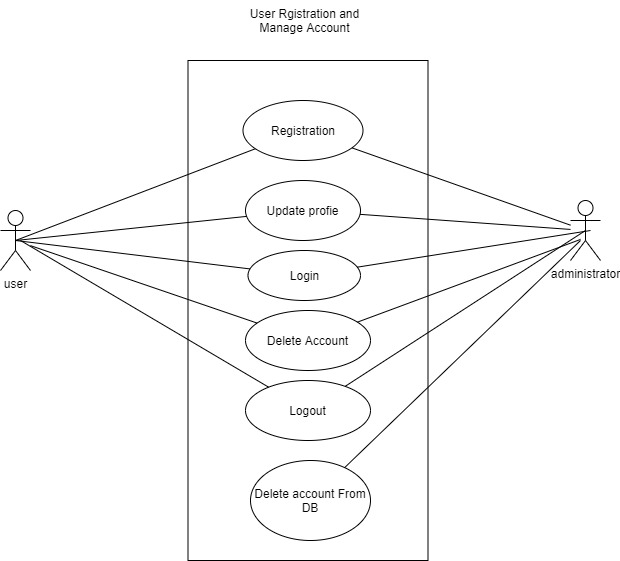
User Story 2:

As an End user, I need to get authenticated by providing username and password and I should be able to obtain a token so that I can make all other API calls



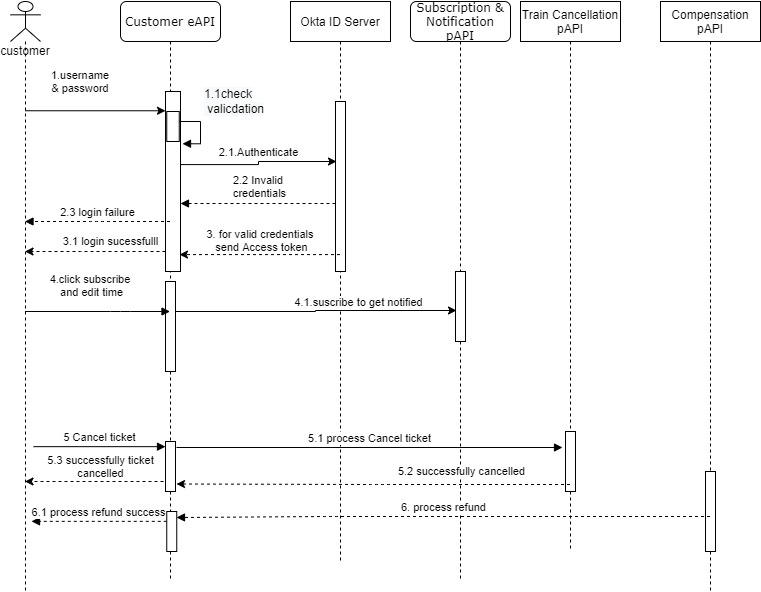
## Diagramming - Use case

Create a use case diagram to represent different use cases my trains project



## Diagramming - Sequence Diagram - UML

Create a sequence diagram to represent one of the use cases of my trains project



## API Led Connectivity Architecture - Archimate

Create an API led connectivity architecture diagram for the use cases of my trains project and use archimate to draw the diagram.

## Module 2: Design Center

## RESTFul APIs vs Non RESTFul APIs

## RAML

## Definition

## OAS (Swagger)

## Definition

## GraphQL

facebook story and advantage of GraphQL. It reduces the number of calls and improves user experience.

## RPC

Slack API to show what's RPC API, RPC uses header parameters to add operation names. Used in process centric APIs

## SOAP

Demonstrate a WSDL file

## Richardson’s Maturity Model and Hypermedia

Shows the different levels of maturity of APIs starting from Level 0 to Level 3 which uses hypermedia.

## Visual editor & Design Center

Demonstrate the use of Visual Editor to design APIs and how it can be edited in Text Editor

## API Fragments

## Data Type

## Traits

## Library

## Example

## Data modelling

## EDM/CDM

Discuss about enterprise wide data model or common / canonical data model. This is a time taking process to define a data model across enterprise. if there is enterprise data model which is successful then use it otherwise go for bounded context

## Bounded Context

Define what's Bounded Contexts. Identify a domain and draw a boundary to define a data model. You could start with business domain or subdomain or even a project. At the worst case every API could have its own data model. You also need to look at what’s power relationships.

## Mirror a Backend System

You could define a data model mirroring a back end system. This could be done if there are no enterprise data models or bounded context data models.

## Exercise

## Data modelling - Bounded Context Data modelling

Create a bounded context based data model for my trains project

## RAML

Create a RAML based API specification using data type, library for a API in my trains project

## XSD

Create a xsd for input string and output string data type.

## SOAP

Identify the sections of a wsdl file and create a new wsdl for helloWorld Soap API using the XSD of above exercise

## Module 3: Exchange

## Connector Types

## Community

## Premium

## Certified

## Select

## Custom Connector

## Common Connectors

## HTTP

## DB

## SFTP/FTP/FILE

## Scheduler

## REST Connector

## Differences in features when you compare with HTTP

## Exercise

## Custom Connector

Create a simple custom connector using Mule SDK

## REST Connector

Consume a REST connector in anypoint studio for the already published API in exchange and compare the differences of features from http connector.

## Module 4: Developer Portal

## Developer Portal

## API Notebook

## New Pages

## Request for access

## Share Resources to public portal

## External Access

## API Community Manager

## Exercise

## Share API to public portal

Share an API published in exchange to a public portal and use the API console.

## 

## Module 5: Build Mule Application

## Mule Projects & Domain Projects

## Naming Conventions

## Logging

## Log4j

## Correlation Id

CorrelationId could be of use to uniquely identify a single request end to end from client app to the downstream systems or apis.

## Externalisation of logs

Externalisation of logs to log management systems like splunk, Elk etc.

## Versioning

## Semantic Versioning

major . minor. patch

## Control Flows

## Choice

## First Successful

## Round Robin

## Scatter Gather

## Dataweave

## Exception handling

## Exercise

## Mule Projects & Domain project

Create a domain project and deploy them in a standalone server so that multiple applications can use the shared configs like http listner from domain project.

## Exception Handling

Use error mapping, raise error, error continue & error propagate to implement exception handling in various use cases of my trains project.

## Advanced Dataweave

Use a complex scenario and implement dataweave to do the data transformation from source to destination. .

## Logging

Use a mule project to demonstrate the working of json logger and externalise with splunk, use correlationId in the logs.

## Versioning

Implement a mule project with the support of multiple versions APIs

## Module 6: Runtime Manager

## Control plane vs Runtime Plane

## Deployment options

## CloudHub

## Hybrid

## RTF

## PCE

## PCF

## Runtime Behavior / thread pool

## JVM properties

## Exercise

## Customer hosted deployment

setup a customer hosted cluster and deploy two applications in a cluster using runtime manager. This application should also use a domain project to share http listener.

## Module 7: API Manager

## Auto Discovery

## Gatekeeper feature

## Enforcing Policies

## Proxy

## Basic end point

## Anypoint servicemesh

## API Manager Policies

## Policies Depends on external identity vs internal identity

## Exercise

## Configure Oauth2 policy using okta

Create oauth2 policy for an experience api of my-trains project .

## Configure client-id / secret policy for an api

Create clientId enforcement api policy to secure APIs .

## Different Auto Discovery behaviours

Auto discovery in basic end point approach is manual whereas AutoDiscovery is automatic in end point with proxy option.

## Policies using client-id

Client Id enforcement /SLA based rate limit, External Auth2 etc will need client management will ask your to add client-id

## Module 8: Anypoint security

## Features

## Tokenization

## Masking

## Encryption

## Edge Policies

## Exercise

## Explore anypoint security policies

Write an article about the features of anypoint security

## Module 9: Design patterns & integration pattern

## Enterprise Integration Patterns

## Scatter Gather

## Enrichment

## Message Based Routing

## Choice Router

## Transformer

## Dataweave

## XSLT

## Scripting

## Java

## Object Oriented Design Patterns

## Observer pattern

## CQRS / Eventual Consistency

## Iterator Pattern

## Reliability Pattern

## Circuit Breaker

## Exercise

## Circuit Breaker

Use a mule application to demonstrate Circuit Breaker scenario.

## Reliability pattern

Implement a reliability pattern for a use case in which huge files (million records) are synchronized with salesforce. .

## Module 10: Object Store & Use cases

## Object Store

## Persistence & Non persistence behaviour in different deployment options

## State storage use cases

## Caching

## Watermark

## Circuit Breaker

## Token Storage

## External systems for storing state

## Exercise

## Circuit Breaker implementation

Implement Circuit breaker scenario to stop processing messages from a JMS when the downstream systems like DB / API is down. Messages which are failed will be forwarded to DLQ and the system should start processing messages when the downstream systems are online. Use my trains project

## Module 11: Scheduler & Use cases

## Near real time processing

## Scheduler

## Exercise

## Scheduler in cluster

Build an application to see how the scheduler works in the cluster..

## Scheduling strategy in File Connector

Build an application to see how the scheduling strategy works in file connector

## Module 12: Messaging & Use cases

## Messaging / Asynchronous / Real Time Processing options

## JMS

## VM Queue

## Anypoint MQ

## Exercise

## Messaging with million messages

U

## Module 13: Streaming & Use cases

## Streaming Processing options

## Types of Streaming

## Connector Behaviour in streaming

## Streaming in Batch

## 

## Exercise

## adf

dsfasdfas

## Module 14: Asynchronous & Use cases

## Streaming Processing options

## Types of Streaming

## Connector Behaviour in streaming

## Streaming in Batch

## 

## Exercise

## adf

**dsfasdfas**

## Module 15: Testing Strategies

## Unit Testing

## MUnit

## Junit

## Integration Testing

## SoapUI

## Postman

## Rest Assured

## Test Coverage

## Resilience Testing

## Performance Testing

## Blazemeter / JMeter

## Exercise

## Munit Test Case

Create a Munit Test suite to perform unit testing of a use case in my trains project.

## Soap UI Testing

Create a SoapUI project to place a file in a file directory and see the output of mule processing from the database.

## 

## Module 16: High Availability

## High availability

## Cluster

## Multi Worker in CloudHub

## Multiple Standalone nodes

## Load balancing

## Primary Nodes

## Load balancing of VM Queues and Object store

## Behaviour of Connectors in Load balancing

## Exercise

## Module 17: Reliability

## Reliability

## Transaction

## Until Successful

## Reconnection Strategies

## Redelivery

## Exercise

## Module 18: Performance Tuning

## Performance tuning

## Dataweave

## Connector Configuration

## Component Configuration

## Profiling

## JvisualVM

## Exercise

## Module 19: Network Security

## Security

## Data at Rest

## Encryption

## Crypto Module

## Data in Transit

## TLS (one way / two way)

## VPC

## VPN

## Exercise

## Module 20: Application Security

## Security

## Data at Rest

## Encryption

## Crypto Module

## Data in Transit

## TLS (one way / two way)

## VPC

## VPN

## Exercise

## Module 21: Monitoring & Automation

## Visualizer

## Anypoint Monitoring

## Custom Dashboard

## Builtin Dashboard

## Anypoint Analytics

## Custom Dashboard

## Chart

## Reports

## Runtime Manager Dashboard

## Insights

## Alerts

## Runtime Manager

## API Manager

## CloudHub Connector

## Exercise