DUMPSARENA

Salesforce Certified MuleSoft Integration Architect 1

Salesforce MuleSoft-Integration-Architect-I

Version Demo

Total Demo Questions: 29

Total Premium Questions: 299

Buy Premium PDF

https://dumpsarena.co

sales@dumpsarena.co

sales@dumpsarena.co
dumpsarena.co

Topic Break Down

Topic	No. of Questions
Topic 1, Anypoint Platform Architecture	57
Topic 2, Application Network Design	26
Topic 3, Integration Architecture	109
Topic 4, Security	34
Topic 5, Operations	72
Topic 6, Mix Questions	1
Total	299

QUESTION NO: 1

An organization is establishing a Center for Enablement (C4E) to increase reuse and allow delivery teams to build integrations with greater autonomy.

Which two activities are appropriate responsibilities of the C4E? (Choose two answers)

A. Publish reusable assets, standards, and reference implementations

B. Provide training, guidance, and governance guardrails to delivery teams

C. Build and maintain every integration for all lines of business

D. Prevent delivery teams from directly consuming assets in Exchange

E. Take ownership of all business priorities and project funding decisions

ANSWER: A B

Explanation:

Publishing reusable assets, standards, and reference implementations helps teams discover proven integration capabilities instead of rebuilding common patterns. Providing enablement through training, guidance, and governance guardrails allows teams to deliver independently while maintaining agreed security, quality, and API design practices.

A C4E is not intended to become the sole delivery team for every integration initiative. Preventing teams from consuming Exchange assets works against self-service and reuse. Replacing business ownership of priorities with a centralized technical team would also conflict with the C4E's enablement role.

QUESTION NO: 2

An API has been published to external consumers and has many active client applications. The API team wants to release a new minor version without requiring consumers to change existing requests or response parsing.

Which two changes are generally backward-compatible for this API? (Choose two answers)

A. Add an optional property to an existing response body

B. Add an optional query parameter to an existing operation

C. Rename an existing property in a response body

D. Change an optional request property to required

E. Change the business meaning of an existing success status code

ANSWER: A B

Explanation:

Adding an optional property to an existing response and adding an optional query parameter are generally backward-compatible changes. Existing clients can continue sending the same requests, and clients that do not need the new capability can ignore the new response property or omit the new query parameter.

Renaming or removing an existing response property breaks consumers that depend on that contract. Changing an optional request property to required breaks consumers that do not send it. Changing the meaning of an existing status code is also a contract-breaking change because consumers can rely on its established semantics.

QUESTION NO: 3

Refer to the exhibit.

An organization is sizing an Anypoint VPC for the non-production deployments of those Mule applications that connect to the organization's on-premises systems. This applies to approx. 60 Mule applications. Each application is deployed to two CloudHub i workers. The organization currently has three non-production environments (DEV, SIT and UAT) that share this VPC. The AWS region of the VPC has two AZs.

The organization has a very mature DevOps approach which automatically progresses each application through all non-production environments before automatically deploying to production. This process results in several Mule application deployments per hour, using CloudHub's normal zero-downtime deployment feature.

What is a CIDR block for this VPC that results in the smallest usable private IP address range?

A. 10.0.0.0/26 (64 IPS)

B. 10.0.0.0/25 (128 IPs)

C. 10.0.0.0/24 (256 IPs)

D. 10.0.0.0/22 (1024 IPs)

ANSWER: D

Explanation:

10.0.0.0/22 (1024 IPs) is the smallest listed CIDR range that provides sufficient address capacity for the shared non-production VPC. The steady-state deployment requirement is 60 applications × 3 environments × 2 workers, or 360 worker IP addresses. CloudHub workers deployed into an Anypoint VPC consume private addresses from the VPC CIDR range. Because the delivery process uses zero-downtime deployments and performs several deployments each hour, the VPC must also have headroom for replacement workers that run temporarily while the existing worker instances continue to serve traffic. Capacity planning must therefore account for both the normally running workers and concurrent deployment overlap, rather than sizing only for the 360 steady-state workers.

A /22 CIDR contains 1,024 total addresses, giving enough capacity for the worker fleet, temporary zero-downtime deployment instances, and platform/network-reserved addresses. This provides practical operational headroom while remaining the smallest range among those presented that can support the expected peak allocation. MuleSoft documents that applications in a CloudHub VPC use addresses from the VPC's assigned CIDR block and that the range must be sized for the applications and workers deployed to it. See the CloudHub networking guide and MuleSoft's CloudHub deployment documentation.

QUESTION NO: 4

A Mule application consumes payment-status messages from an Anypoint MQ queue. Some messages fail repeatedly because they contain malformed business data and cannot be corrected by retrying the consumer flow.

The solution must prevent such messages from blocking normal queue processing while retaining them for investigation and possible correction.

Which two actions should the integration architect recommend? (Choose two.)

A. Configure a dead-letter queue and a maximum number of delivery attempts for the source queue

B. Use manual acknowledgement and acknowledge each message only after successful processing

C. Use immediate acknowledgement so that malformed messages are removed before processing begins

D. Increase the acknowledgement timeout so that malformed messages remain in processing indefinitely

E. Configure every consumer to retry the malformed message until it succeeds

ANSWER: A B

Explanation:

Configure a dead-letter queue and a maximum delivery-attempt value for the source queue. When a message cannot be successfully consumed within the configured number of attempts, Anypoint MQ moves it to the dead-letter queue. This keeps repeatedly failing messages available for investigation without allowing them to be delivered indefinitely on the primary queue.

The consumer should also use manual acknowledgement and acknowledge a message only after successful processing. A malformed message that causes the flow to fail before acknowledgement remains eligible for redelivery and can eventually reach the dead-letter queue. Immediate acknowledgement would remove the message before processing completes and could result in message loss. Retrying malformed business data indefinitely does not resolve the underlying data issue.

QUESTION NO: 5

An organization is building a test suite for their applications using m-unit. The integration architect has recommended using test recorder in studio to record the processing flows and then configure unit tests based on the capture events

What are the two considerations that must be kept in mind while using test recorder

(Choose two answers)

A. Tests for flows cannot be created with Mule errors raised insidethe flow or already existing in the incoming event

B. Recorder supports smoking a message before or inside a ForEach processor

C. The recorder support loops where the structure of the data been testedchanges inside the iteration

D. A recorded flow execution ends successfully but the result doesnot reach its destination because the application is killed

E. Mocking values resulting from parallel processes are possible and will not affect the execution of the processes that follow in the test

ANSWER: A D

Explanation:

The Test Recorder is designed to observe a successful Mule flow execution and use the captured event data to generate an MUnit test skeleton with appropriate mocks and assertions. Consequently, *Tests for flows cannot be created with Mule errors raised insidethe flow or already existing in the incoming event* is a key consideration. A recording session needs an event that can be processed normally so the recorder can capture the processors, inputs, and outputs needed for generated test behavior. Teams should therefore use a successful representative execution when recording and add explicit error-path tests separately.

A recorded flow execution ends successfully but the result doesnot reach its destination because the application is killed is also an important operational consideration. The recorder controls the application execution for capture purposes. A flow can complete from the recorder's perspective, while externally visible or asynchronous downstream work has not fully reached its ultimate destination before the application is stopped. Recordings should therefore be performed in an isolated environment, and any important external or asynchronous behavior should be validated through purpose-built MUnit tests and assertions after generation.

For implementation details and recorder behavior, consult the MUnit Test Recorder documentation and the MUnit testing concepts documentation.

QUESTION NO: 6

An organization exposes a public API through API Manager. Browser-based applications hosted on two approved web domains must be able to invoke the API. In addition, requests must be accepted only when they originate from the organization's approved corporate network ranges.

Which two API Manager policies are most appropriate? (Choose two answers)

A. Apply a CORS policy

B. Apply an IP Allowlist policy

C. Apply a Client ID Enforcement policy

D. Apply a Rate Limiting policy

E. Apply a Header Removal policy

ANSWER: A B

Explanation:

A CORS policy is appropriate because browser clients require the API to return the required cross-origin response headers for approved origins. An IP Allowlist policy is appropriate because it restricts inbound requests according to the source IP addresses or network ranges defined by the organization.

Client ID enforcement identifies an application that supplies valid client credentials, but it does not by itself permit browser cross-origin requests or restrict traffic by source network. Rate limiting controls request volume rather than network origin. Header removal can modify HTTP headers but is not an access-control policy for approved browser origins or source ranges.

QUESTION NO: 7

Which Anypoint Platform component should a MuleSoft developer use to create an API specification prior to building the API implementation?

A. MUnit

B. API Designer

C. API Manager

D. Runtime Manager

ANSWER: B

Explanation:

API Designer is the appropriate Anypoint Platform component for creating an API specification before implementation begins. It supports a design-first API development approach, in which the API contract is defined, reviewed, and shared independently of the Mule application that will later implement it. In API Designer, developers can author and edit API definitions using RAML or OpenAPI Specification, defining resources, methods, request and response payloads, parameters, security schemes, examples, and reusable data types. The resulting specification provides a clear, machine-readable contract that consumers and implementation teams can use to align on expected API behavior. API Designer also provides interactive documentation and validation assistance, helping teams refine the contract before writing integration flows. After the specification is ready, it can be published to Anypoint Exchange and used as the basis for implementation, governance, and consumer discovery. This workflow helps reduce ambiguity and supports consistent API-led delivery across teams. MuleSoft documents API Designer as the web-based tool for designing, documenting, and testing APIs and API fragments. See Design an API Specification and Anypoint Design Center Documentation.

QUESTION NO: 8

An IT integration delivery team begins a project by gathering all of the requirements, and proceeds to execute the remaining project activities as sequential, non-repeating phases.

Which IT project delivery methodology is this team following?

A. Kanban

B. Scrum

C. Waterfall

D. Agile

ANSWER: C

Explanation:

Waterfall is the appropriate methodology because it organizes delivery into a linear sequence of distinct phases. In this approach, the team defines and documents requirements at the beginning, then progresses through activities such as design, implementation, testing, and deployment in order. Completion of one phase provides the basis for beginning the next, rather than continuously revisiting requirements and reprioritizing work throughout delivery. This model is particularly applicable when the required scope, interfaces, compliance constraints, and expected deliverables can be understood early and are unlikely to change significantly. For an integration initiative, that can include projects with stable source and target systems, established data mappings, and fixed business processes. The Project Management Institute describes predictive approaches as planning most work in advance and executing it through a defined sequence, which aligns with the scenario's upfront requirements gathering and non-repeating phases. Salesforce architecture work can use either predictive or adaptive delivery depending on uncertainty and change, but the explicit linear progression in this scenario identifies Waterfall. See the Project Management Institute's overview of predictive, agile, and hybrid approaches and the Atlassian Waterfall methodology guide.

QUESTION NO: 9

A Mule application is synchronizing customer data between two different database systems.

What is the main benefit of using XA transaction over local transactions to synchronize these two database system?

A. Reduce latency

B. Increase throughput

C. Simplifies communication

D. Ensure consistency

ANSWER: D

Explanation:

Ensure consistency is correct because an XA transaction coordinates work across multiple XA-capable resource managers, such as two separate database systems. Mule uses the XA transaction protocol to manage a distributed transaction through a two-phase commit process. Before the final commit, each participating database indicates whether it can successfully commit its portion of the work. The transaction manager then directs all participants to commit only when every participant is prepared; otherwise, it rolls back the transaction across the participating resources.

This all-or-nothing behavior is the key benefit when synchronizing customer data. For example, if a customer update succeeds in one database but the second database cannot commit because of a constraint or connectivity failure, XA coordination prevents the integration from leaving the systems in a partially updated state. This preserves the atomicity and consistency expected for a business operation that spans both databases. Mule local transactions apply only to a single transactional resource and therefore cannot provide this coordinated outcome across independent database systems. See MuleSoft's transaction management documentation and Oracle's overview of the XA distributed transaction model.

QUESTION NO: 10

As a part of project , existing java implementation is being migrated to Mulesoft. Business is very tight on the budget and wish to complete the project in most economical way possible.

Canonical object model using java is already a part of existing implementation. Same object model is required by mule application for a business use case. What is the best way to achieve this?

A. Make use of Java module

B. Create similar model for Mule applications

C. Create a custom application to read Java code and make it available for Mule application

D. Use Anypoint exchange

ANSWER: A

Explanation:

Make use of Java module is the best approach because it enables the Mule application to reuse the established Java canonical object model rather than rebuilding and maintaining an equivalent model. The existing model should be packaged as a reusable JAR and added as a Maven dependency of the Mule application. Its classes can then be used on the application classpath, while the Java module can invoke compatible Java methods and constructors where interaction with the model is required. This preserves the existing domain definitions, validation behavior, and serialization-related logic that may already be embedded in the Java implementation. Reusing the shared artifact also reduces implementation effort, testing cost, and the risk that independently maintained Java and Mule models drift apart over time. For a budget-constrained migration, this is a practical incremental path: retain stable business-domain assets in Java and implement integration orchestration in Mule. MuleSoft documents how the Java module invokes Java methods and creates Java objects, and Maven dependency management provides the standard mechanism for including shared libraries in a Mule project. See the Mule Java Module documentation and Mule Maven reference.

QUESTION NO: 11

An organization has decided on a cloudhub migration strategy that aims to minimize the organizations own IT resources. Currently, the organizational has all of its Mule applications running on its own premises and uses an premises load balancer that exposes all APIs under the base URL https://api.acme.com

As part of the migration strategy, the organization plans to migrate all of its Mule applications and load balancer to cloudhub

What is the most straight-forward and cost effective approach to the Mule applications deployment and load balancing that preserves the public URLs?

A. Deploy the Mule applications to CloudhubUpdate the CNAME record for an api.acme.com in the organizations DNS server pointing to the A record of a cloudhub dedicated load balancer(DLB)Apply mapping rules in the DLB to map URLs to their corresponding Mule applications

B. For each migrated Mule application, deploy an API proxy Mule application to Cloudhub with all applications under the control of a dedicated load balancer(CLB)Update the CNAME record for api.acme.com in the organization DNS server pointing to the A record of a cloudhub dedicated load balancer(DLB)Apply mapping rules in the DLB to map each API proxy application to its corresponding Mule applications

C. Deploy the Mule applications to CloudhubCreate CNAME record for api.acme.com in the Cloudhub Shared load balancer (SLB) pointing to the A record of the on-premise load balancerApply mapping rules in the SLB to map URLs to their corresponding Mule applications

D. Deploy the Mule applications to CloudhubUpdate the CNAME record for api.acme.com in the organization DNS server pointing to the A record of the cloudhub shared load balancer(SLB)Apply mapping rules in the SLB to map URLs to their corresponding Mule applications.

ANSWER: A

Explanation:

Deploy the Mule applications to Cloudhub, update the DNS entry for `api.acme.com` to direct traffic to a CloudHub dedicated load balancer, and configure mapping rules is the appropriate approach. A dedicated load balancer provides a managed CloudHub ingress layer with a stable, organization-specific endpoint and supports custom domains. This allows the organization to retain the existing public base URL while moving both application hosting and traffic distribution off premises.

Mapping rules on the dedicated load balancer route requests based on host, path, or both to the intended deployed Mule application. For example, requests for `https://api.acme.com/orders` and `https://api.acme.com/customers`

can be sent to different CloudHub applications without changing the public API addresses consumed by clients. The organization therefore needs neither its former on-premises load balancer nor additional proxy applications solely to perform path-based routing. CloudHub manages the load-balancing infrastructure, while DNS directs the established API hostname to it. MuleSoft documents dedicated load balancers as the CloudHub capability for custom-domain routing and mapping rules; see Dedicated Load Balancer and Dedicated Load Balancer Mapping Rules.

QUESTION NO: 12

What are two reasons why a typical MuleSoft customer favors a MuleSoft-hosted Anypoint Platform runtime plane over a customer-hosted runtime for its Mule application deployments? (Choose two.)

A. Reduced application latency

B. Increased application isolation

C. Reduced time-to-market for the first application

D. Increased application throughput

E. Reduced IT operations effort

ANSWER: C E

Explanation:

Reduced time-to-market for the first application and **Reduced IT operations effort** are the primary advantages of selecting a MuleSoft-hosted runtime plane, such as CloudHub. MuleSoft provides and operates the underlying runtime infrastructure, so a customer does not need to first procure, install, configure, secure, and maintain its own compute environment before deploying an initial Mule application. This allows development teams to concentrate on implementing integrations and releasing them sooner.

A MuleSoft-hosted deployment also transfers substantial platform-level operational work to MuleSoft. MuleSoft manages the hosted service infrastructure and provides platform capabilities for deploying, monitoring, scaling, and operating applications. Customer teams still own their Mule applications, configurations, integrations, and operational governance, but they avoid much of the effort associated with administering runtime hosts and their supporting infrastructure. This is especially valuable when a team has limited infrastructure operations capacity or needs to begin delivering APIs and integrations quickly.

MuleSoft documents CloudHub as a fully managed cloud service for deploying Mule applications and describes its operational and deployment capabilities in the CloudHub documentation. Additional deployment and operational details are available in the Anypoint Runtime Manager documentation.

QUESTION NO: 13

An organization has chosen Mulesoft for their integration and API platform.

According to the Mulesoft catalyst framework, what would an integration architect do to create achievement goals as part of their business outcomes?

A. Measure the impact of the centre for enablement

B. build and publish foundational assets

C. Agree upon KPIs and help develop an overall success plan

D. evangelize API ' s

ANSWER: C

Explanation:

“Agree upon KPIs and help develop an overall success plan” is correct because MuleSoft Catalyst connects technical delivery activities to measurable business outcomes. In this approach, the integration architect collaborates with business and technology stakeholders to define the intended outcomes of the integration program, identify meaningful key performance indicators, and establish how success will be evaluated over time. Achievement goals must be concrete enough to track progress and demonstrate value, such as improved delivery speed, increased reuse, reduced operational effort, or better customer experiences. A documented success plan provides the shared baseline for measuring those results, assigning ownership, reviewing progress, and making adjustments as the organization advances its integration strategy. This responsibility helps ensure that architecture decisions and API-led connectivity initiatives are guided by business priorities rather than solely by implementation tasks. MuleSoft describes Catalyst as a framework for aligning people, processes, and technology to achieve business outcomes, with measurable value and an operating model that supports sustained adoption. See the MuleSoft Catalyst overview and the MuleSoft Catalyst documentation for further guidance on outcome-driven transformation and success planning.

QUESTION NO: 14

An organization deploys a stateful Mule application to Runtime Fabric. The application must remain available when a single Runtime Fabric node fails, and business state must survive a pod restart or node failure.

Which two design decisions best address these requirements? (Choose two.)

A. Deploy multiple application replicas with sufficient capacity to serve traffic after one node fails

B. Store required business state in a persistent external store or persistent object store

C. Store required business state only in application variables on each replica

D. Use a non-persistent object store because Runtime Fabric automatically replicates all in-memory entries

E. Deploy one application replica with a larger JVM heap

ANSWER: A B

Explanation:

Deploying more than one replica allows the application to continue serving requests when a node hosting one replica becomes unavailable, provided the remaining replicas and supporting infrastructure have sufficient capacity. A single application replica creates a single runtime instance that cannot provide application-level availability after its hosting node fails.

State that must survive a restart or node failure must be stored in a persistent external service or persistent object store suitable for the required recovery behavior. Non-persistent object-store entries are memory-based and can be lost when the runtime instance stops. Keeping state only in application variables has the same limitation.

Increasing JVM heap size does not make in-memory state durable, and relying on a single replica cannot meet the stated node-failure availability requirement.

QUESTION NO: 15

As part of a growth strategy, a supplier signs a trading agreement with a large customer. The customer sends purchase orders to the supplier according to the ANSI X12 EDI standard, and the supplier creates the orders in its ERP system using the information in the EDI document.

The agreement also requires that the supplier provide a new RESTful API to process request from the customer for current product inventory level from the supplier’ s ERP system.

Which two fundamental integration use cases does the supplier need to deliver to provide an end-to-end solution for this business scenario? (Choose two.)

A. Synchronized data transfer

B. Sharing data with external partners

C. User interface integration

D. Streaming data ingestion

E. Data mashups

ANSWER: A B

Explanation:

Synchronized data transfer is required because incoming ANSI X12 purchase-order documents must be received, interpreted, transformed into the ERP's required order format, and used to create corresponding orders. This is a system-to-system exchange in which the business data supplied by the customer must be reliably propagated into the supplier's system of record. MuleSoft B2B Integration capabilities are intended to support partner onboarding and the processing of B2B messages, including EDI formats and trading-partner transactions.

Sharing data with external partners is also required because the supplier must expose current inventory information to its customer through a RESTful API. The API provides a controlled, reusable interface through which an authorized external party can request data held in the supplier's ERP, without requiring direct access to that system. This supports the agreement's need for timely inventory visibility while allowing the supplier to manage security, access, and the API contract independently of the ERP implementation.

Together, these use cases address both directions of the partner relationship: inbound operational order data and outbound, on-demand inventory data. See MuleSoft's B2B Integration documentation and its overview of APIs.

QUESTION NO: 16

A Mule application name Pub uses a persistence object store. The Pub Mule application is deployed to Cloudhub and it configured to use Object Store v2.

Another Mule application name sub is being developed to retrieve values from the Pub Mule application persistence object Store and will also be deployed to cloudhub.

What is the most direct way for the Sub Mule application to retrieve values from the Pub Mule application persistence object store with the least latency?

A. Use an object store connector configured to access the Pub Mule application persistence object store

B. Use a VM connector configured to directly access the persistence queue of the Pub Mule application persistence object store.

C. Use an Anypoint MQ connector configured to directly access the Pub Mule application persistence object store

D. Use the Object store v2 REST API configured to access the Pub Mule application persistence object store.

ANSWER: D

Explanation:

Use the Object store v2 REST API configured to access the Pub Mule application persistence object store. Object Store v2 exposes a REST API specifically for working with object stores hosted in Anypoint Platform. Its endpoints support reading a value by its key from an identified application object store, in addition to listing stores and keys and creating, updating, or deleting values. Therefore, the Sub application can make an authenticated HTTPS request directly to the Object Store v2 service and retrieve the required persisted value from Pub's store without introducing an intermediate messaging or application layer. This is the direct supported mechanism when a client needs programmatic access to an Object Store v2 store associated with another deployed application. The calling application must use appropriate Anypoint Platform authentication and have permissions for the relevant organization, environment, and application object store. Object Store v2 is primarily intended for application state and persistence rather than general-purpose interapplication messaging, but where the requirement is explicitly to retrieve an existing value from Pub's Object Store v2 persistence store, the REST API is the applicable access interface. See the Object Store v2 API documentation and the Object Store documentation for the available operations and access model.

QUESTION NO: 17

A retailer is exposing inventory availability from an ERP system to a mobile application and to a warehouse application. The mobile application needs a simplified customer-facing response, while the warehouse application needs operational inventory details.

The ERP system is expected to change its interface over time.

Which two API-led connectivity design decisions are most appropriate? (Choose two answers)

A. Create a System API that abstracts the ERP inventory interface

B. Create separate Experience APIs for the mobile and warehouse applications

C. Allow both client applications to directly invoke the ERP interface

D. Create one client-facing API that contains all channel-specific response fields

E. Publish the ERP schemas to Exchange for the client applications to consume directly

ANSWER: A B

Explanation:

A System API should encapsulate the ERP system and expose inventory capabilities independently of the ERP's proprietary interface. This isolates downstream consumers from ERP-specific protocols, data structures, and future implementation changes.

Separate Experience APIs are appropriate because the mobile and warehouse applications require different representations of inventory data. Each Experience API can provide a channel-specific contract while consuming reusable inventory capabilities exposed through the API-led layer.

Allowing both client applications to directly invoke the ERP creates tight coupling to the system of record. A single shared client-facing API with every channel-specific field mixes distinct consumer concerns into one contract. Publishing raw ERP schemas to Exchange does not isolate consumers from ERP changes.

QUESTION NO: 18

As a part of project requirement, Java Invoke static connector in a mule 4 application needs to invoke a static method in a dependency jar file. What are two ways to add the dependency to be visible by the connectors class loader?

(Choose two answers)

A. In the Java Invoke static connector configuration, configure a path and name of the dependency jar file

B. Add the dependency jar file to the java classpath by setting the JVM parameters

C. Use Maven command to include the dependency jar file when packaging the application

D. Configure the dependency as a shared library in the project POM

E. Update mule-artefact.json to export the Java package

ANSWER: C D

Explanation:

Use Maven command to include the dependency jar file when packaging the application is correct because Mule applications are Maven projects and external Java libraries must be resolved and packaged as part of the deployable application artifact. Declaring the JAR as a Maven dependency provides repeatable dependency resolution and ensures its classes are available at runtime as part of the application build. The Java module can then resolve the target class and invoke its public static method when the library is made available in the application's packaged dependencies.

Configure the dependency as a shared library in the project POM is also correct because Mule 4 uses separate class loaders for the application and its plugins/connectors. A shared library configuration in the Mule Maven Plugin explicitly exposes a dependency to both the application and plugin class loaders. This is the appropriate mechanism when a connector, such as the Java module, must load classes contained in an application dependency. The dependency should be declared in the project and identified as a shared library during packaging so that the connector can access the class at runtime. MuleSoft documents the application/plugin class-loading model and shared-library configuration at Mule Classloading and Maven dependency handling at Maven Dependency Mechanism.

QUESTION NO: 19

An organization wants its API delivery teams to reuse common API definitions and enforce consistent API design standards. The teams use RAML specifications and develop APIs independently.

Which two practices best support these requirements? (Choose two.)

A. Publish reusable API fragments and libraries to Anypoint Exchange

B. Apply API Governance rules to validate API specifications against organizational standards

C. Copy approved RAML data types into each API project so that teams can modify them independently

D. Apply runtime rate-limiting policies to ensure that API specifications use common data types

E. Use MUnit tests as the primary mechanism for sharing RAML security schemes between APIs

ANSWER: A B

Explanation:

Publishing reusable API fragments, such as data types, security schemes, traits, and libraries, to Anypoint Exchange supports consistent API definitions and avoids teams recreating common contract elements. API designers can import approved fragments into their specifications, allowing a centrally governed asset to be reused across APIs.

Applying API Governance rules to API specifications provides automated validation of the organization's design standards. Governance can identify nonconforming specifications before implementation and promotion, which is more effective than relying only on manual reviews after APIs have been built.

Duplicating common types in every project creates drift, while applying runtime policies does not validate the design of an API specification. MUnit validates implementation behavior rather than acting as a reusable API-definition mechanism.

QUESTION NO: 20

As a part of business requirement , old CRM system needs to be integrated using Mule application. CRM system is capable of exchanging data only via SOAP/HTTP protocol. As an integration architect who follows API led approach , what is the the below step you will perform so that you can share document with CRM team?

A. Create RAML specification using Design Center

B. Create SOAP API specification using Design Center

C. Create WSDL specification using text editor

D. Create WSDL specification using Design Center

ANSWER: C

Explanation:

Create WSDL specification using text editor is correct because a WSDL is the contract document conventionally used to define a SOAP web service. It gives the CRM team a technology-neutral, shareable description of the integration interface: the available service operations, request and response message structures, XML schema types, SOAP bindings, transport

details, and service endpoint information. This lets both teams agree on the contract before or while implementing the Mule application, which is consistent with an API-led approach centered on a clearly defined interface.

The WSDL document is XML-based and can be authored and maintained as source-controlled text. Mule applications can subsequently use the defined contract when implementing or consuming SOAP services. MuleSoft's Web Service Consumer documentation shows that SOAP service consumption is driven by a WSDL location and its service, port, and operation definitions. The WSDL specification also defines WSDL as the language for describing web-service interfaces and their bindings. See MuleSoft Web Service Consumer Connector Reference and W3C Web Services Description Language.

QUESTION NO: 21

A Mule application receives an order request, invokes an external shipping API, and publishes a shipment request to a messaging destination. The integration team must create automated MUnit tests that are deterministic and do not invoke the external shipping provider or a real messaging broker.

Which two MUnit practices should the team use? (Choose two answers)

A. Mock the HTTP Request operation that invokes the shipping API

B. Use Verify Call to confirm that the messaging publish operation was invoked

C. Deploy the application to CloudHub before executing each MUnit test

D. Invoke the real shipping provider with production-like test orders

E. Use Test Recorder instead of configuring mocks and assertions

ANSWER: A B

Explanation:

Mocking the HTTP Request operation allows the test to return controlled shipping-provider responses without making a network call. Verifying that the publish operation was called lets the test confirm that the Mule flow attempted to send the expected shipment request without requiring a real messaging broker.

Deploying the application to CloudHub does not make a unit test deterministic and still requires access to external dependencies. Calling the real shipping provider creates dependency on its availability, test data, and response behavior. The Test Recorder can assist with generating a starting test structure, but it is not required to isolate the test from external systems.

QUESTION NO: 22

A high-volume eCommerce retailer receives thousands of orders per hour and requires notification of its order management, warehouse, and billing system for subsequent processing within 15 minutes of order submission through its website.

Which integration technology, when used for its typical and intended purpose, meets the retailer's requirements for this use case?

A. Managed File Transfer (MFT)

B. Publish/Subscriber Messaging Bus (Pub/Sub)

C. Enterprise Data Warehouse (EDW)

D. Extract Transform Load (ETL)

ANSWER: B

Explanation:

Publish/Subscriber Messaging Bus (Pub/Sub) is the appropriate integration technology because a single order event must be distributed reliably and asynchronously to several independent downstream systems. When an order is submitted, the eCommerce application publishes an event to a topic or exchange. The order management, warehouse, and billing systems each subscribe to that event stream and can process their own copy independently. This supports the required one-to-many notification pattern without requiring the website to make synchronous calls to every receiving system.

The messaging layer also decouples order submission from downstream processing capacity and availability. The retailer can continue accepting a high volume of orders while subscribers consume messages at their own rates, with queues, acknowledgments, retries, and durable delivery configured to help meet the fifteen-minute processing-notification objective. This architecture is especially suitable where new consumers may be added later without changing the order-submission application. MuleSoft Anypoint MQ supports publish-subscribe messaging through exchanges and queues, enabling a published message to be routed to multiple subscriber queues. See the Anypoint MQ exchanges documentation and MuleSoft's Anypoint MQ overview.

QUESTION NO: 23

An organization heeds to procure an enterprise software system to increase cross-selling opportunities and better rack prospect data.

Which category of enterprise software has these core capabilities, when used for its typical and intended purpose?

A. Supply Chain Management (SCM)

B. IT Service Management (ITSM)

C. Business-to-Business (A2B)

D. Customer Relationship Management (CRM)

ANSWER: D

Explanation:

Customer Relationship Management (CRM) is the enterprise software category intended to centralize and manage an organization's relationships and interactions with prospects and customers. CRM platforms maintain prospect and account records, capture sales activities and communications, manage leads and opportunities through the sales pipeline, and make customer information available to sales and marketing teams. This unified view of relationship data enables teams to identify relevant products or services for existing customers, supporting informed cross-selling and upselling efforts. CRM also helps organizations qualify and nurture prospects by recording engagement history, contact details, campaign responses, and next steps, rather than leaving critical customer context in disconnected systems or individual spreadsheets. In Salesforce, lead management supports tracking prospects from initial capture through qualification, while opportunity and account management provide the information needed to manage customer relationships and identify revenue opportunities. Salesforce describes CRM as technology for managing a company's relationships and interactions with customers and potential customers. See Salesforce: What Is CRM? and Salesforce Help: Leads.

QUESTION NO: 24

An integration Mule application is deployed to a customer-hosted multi-node Mule 4 runtime duster. The Mule application uses a Listener operation of a JMS connector to receive incoming messages from a JMS queue.

How are the messages consumed by the Mule application?

A. Depending on the JMS provider ' s configuration, either all messages are consumed by ONLY the primary cluster node or else ALL messages are consumed by ALL cluster nodes

B. Regardless of the Listener operation configuration, all messages are consumed by ALL cluster nodes

C. Depending on the Listener operation configuration, either all messages are consumed by ONLY the primary cluster node or else EACH message is consumed by ANY ONE cluster node

D. Regardless of the Listener operation configuration, all messages are consumed by ONLY the primary cluster node

ANSWER: C

Explanation:

Depending on the Listener operation configuration, either all messages are consumed by ONLY the primary cluster node or else EACH message is consumed by ANY ONE cluster node is correct. Mule runtime clusters designate a primary node, and message sources such as the JMS Listener can be configured to run only on that primary node. This is the default behavior of the listener's `primaryNodeOnly` setting, which prevents the same source from being activated independently across every cluster member. Consequently, when this setting remains enabled, the primary node receives the queue's messages.

For a JMS queue workload that should be distributed among cluster members, the listener can instead set `primaryNodeOnly="false"`. Each node then establishes an eligible consumer for the queue. Queue semantics ensure competing consumers receive messages independently: a particular queued message is delivered to one consumer/node, rather than broadcast to every node. This enables horizontal processing while retaining the single-consumer delivery behavior expected for a JMS queue. MuleSoft's JMS connector documentation describes the Listener source and its cluster-related primary-node setting, while the JMS specification defines queues as point-to-point destinations. See MuleSoft JMS Listener documentation and Jakarta Messaging specification.

QUESTION NO: 25

A manufacturing company is developing a new set of APIs for its retail business. One of the APIs is a Master Look Up API, which is a System API,

The API uses a persistent object-store. This API will be used by almost all other APIs to provide master lookup data.

The Master Look Up API is deployed on two CloudHub workers of 0.1 vCore each because there is a lot of master data to be cached. Most of the master

lookup data is stored as a key-value pair. The cache gets refreshed if the key is not found in the cache.

During performance testing, it was determined that the Master Look Up API has a high response time due to the latency of database queries executed to fetch the master lookup data.

What two methods can be used to resolve these performance issues?

Choose 2 answers

A. Implement the HTTP caching policy for all GET endpoints for the Master Look Up API

B. Implement an HTTP caching policy for all GET endpoints in the Master Look Up API

C. Implement locking to synchronize access to the Object Store

D. Upgrade the vCore size from 0.1 vCore to 0.2 vCore

ANSWER: A D

Explanation:

Implementing the HTTP caching policy for all GET endpoints for the Master Look Up API adds response caching in front of the API's lookup operations. Master data is generally read frequently and changes relatively infrequently, making GET responses strong candidates for HTTP caching. When a valid cached response is available, callers can receive the lookup result without invoking the application flow and, importantly, without waiting for the underlying database query. This reduces database load and lowers observed response times for repeated requests. MuleSoft documents that the HTTP Caching policy caches responses for GET requests and returns cached responses while they remain valid: HTTP Caching policy.

Upgrading the vCore size from 0.1 vCore to 0.2 vCore gives each CloudHub worker additional CPU and memory resources. The API must still process cache misses, serialize responses, manage Object Store interactions, and serve a high number of concurrent consumers. More worker resources reduce resource contention and improve throughput, helping the service handle cache-miss processing and concurrent lookup traffic more effectively. CloudHub worker sizing determines the compute and memory capacity available to a deployed application, as described in CloudHub architecture and worker sizing.

QUESTION NO: 26

As an enterprise architect, what are the two reasons for which you would use a canonical data model in the new integration project using Mulesoft Anypoint platform (choose two answers)

A. To have consistent data structure aligned in processes

B. To isolate areas within a bounded context

C. To incorporate industry standard data formats

D. There are multiple canonical definitions of each data type

E. Because the model isolates the back-end systems and supports Mule applications from change

ANSWER: A E

Explanation:

A canonical data model establishes a shared, consistent representation of business data—such as customers, orders, products, and addresses—for integration flows. “To have consistent data structure aligned in processes” is therefore a core reason to use it: systems and APIs can exchange a common business-oriented structure instead of requiring every consumer to understand each source application’s proprietary schema. This improves semantic consistency across processes and makes integrations easier to understand and govern.

“Because the model isolates the back-end systems and supports Mule applications from change” is also correct. Mule applications can map each system-specific payload to or from the canonical model at the appropriate boundary. Consequently, a schema or interface change in a back-end application is localized to its mapping or adapter rather than propagating through every consuming integration. This decoupling reduces the number of point-to-point transformations, limits change impact, and supports reusable APIs and integration assets. MuleSoft’s API-led approach similarly emphasizes separating system-specific complexity from reusable integration capabilities, while DataWeave provides the transformation language used to map between source, canonical, and target formats. See MuleSoft API-led connectivity and MuleSoft DataWeave documentation.

QUESTION NO: 27

An integration team uses Anypoint Platform and follows MuleSoft ' s recommended approach to full lifecycle API development.

Which step should the team ' s API designer take before the API developers implement the AP! Specification?

A. Generate test cases using MUnit so the API developers can observe the results of running the API

B. Use the scaffolding capability of Anypoint Studio to create an API portal based on the API specification

C. Publish the API specification to Exchange and solicit feedback from the API ' s consumers

D. Use API Manager to version the API specification

ANSWER: C

Explanation:

Publish the API specification to Exchange and solicit feedback from the API ' s consumers is the appropriate step before implementation because MuleSoft promotes a design-first, API-led lifecycle in which an API contract is reviewed and validated before developers build the implementation. Publishing the specification as an Exchange asset makes the proposed contract discoverable to intended consumers, architects, and other stakeholders. These users can evaluate whether the resources, operations, request and response models, security requirements, and error behavior meet actual integration needs. Gathering this feedback while the asset is still a specification enables the team to refine the contract early, when changes are much less costly than after application logic and downstream integrations have been developed. Exchange also acts as the governed central catalog for API assets, supporting collaboration and reuse across the organization. After the contract has been agreed on, API developers can implement it with clearer requirements and a

stronger basis for testing, documentation, and consumer adoption. MuleSoft describes Exchange as the place to discover, share, and reuse assets, including APIs and API specifications. See Anypoint Exchange documentation and MuleSoft's guidance on designing APIs in Design Center.

QUESTION NO: 28

A stock trading company handles millions of trades a day and requires excellent performance and reliability within its stock trading system. The company operates a number of event-driven APIs Implemented as Mule applications that are hosted on various customer-hosted Mule clusters and needs to enable message exchanges between the APIs within their internal network using shared message queues.

What is an effective way to meet the cross-cluster messaging requirements of its event-driven APIs?

A. Non-transactional JMS operations with a reliability pattern and manual acknowledgements

B. Persistent VM queues with automatic acknowledgements

C. JMS transactions with automatic acknowledgements

D. extended Architecture (XA) transactions and XA connected components with manual acknowledgements

ANSWER: C

Explanation:

JMS transactions with automatic acknowledgements is the effective approach because a JMS broker provides the shared, durable messaging layer needed for Mule applications deployed across separate customer-hosted clusters. Producers in one cluster can publish to broker-managed destinations, while consumers in another cluster receive messages independently of the producer's availability. This decoupling supports event-driven processing at high volume and enables reliable recovery when an application instance or cluster is temporarily unavailable.

When a Mule flow processes a JMS message within a transaction, successful processing commits the transaction and completes message consumption. If processing fails, the transaction can roll back, allowing the broker to make the message available for redelivery according to its configured policy. Automatic acknowledgement aligns acknowledgement with successful flow completion, avoiding the need for application code to explicitly acknowledge each message while preserving reliable processing semantics. For trading workloads, this pattern supports durable delivery and controlled failure recovery, provided the shared JMS broker is appropriately sized, highly available, and configured with persistent destinations.

MuleSoft documents transactional operation support in the JMS Connector transactions documentation and describes listener acknowledgement behavior in the JMS Listener documentation.

QUESTION NO: 29

An organization currently uses a multi-node Mule runtime deployment model within their datacenter, so each Mule runtime hosts several Mule applications. The organization is planning to transition to a deployment model based on Docker containers in a Kubernetes cluster. The organization has already created a standard Docker image containing a Mule runtime and all required dependencies (including a JVM), but excluding the Mule application itself.

What is an expected outcome of this transition to container-based Mule application deployments?

A. Required redesign of Mule applications to follow microservice architecture principles

B. Required migration to the Docker and Kubernetes-based Anypoint Platform - Private Cloud Edition

C. Required change to the URL endpoints used by clients to send requests to the Mule applications

D. Guaranteed consistency of execution environments across all deployments of a Mule application

ANSWER: D

Explanation:

Guaranteed consistency of execution environments across all deployments of a Mule application is the expected outcome. The organization's standard Docker image defines the reusable runtime baseline: the Mule runtime, JVM, and every required operating-system-level dependency are packaged into a versioned image. Each Kubernetes workload that uses that image starts from the same immutable execution environment, rather than relying on separately installed and potentially divergent runtime software on individual datacenter servers.

The Mule application can be supplied separately during the container build or deployment process while retaining this standardized base image. This makes deployments repeatable across Kubernetes nodes and across lifecycle environments that use the same approved image version. It also simplifies promotion and troubleshooting because a deployed application is paired with a known Mule runtime and dependency set. Environment-specific values, such as credentials, endpoints, and replicas, should still be externalized through configuration and Kubernetes resources; those settings do not change the consistency of the packaged execution baseline.

MuleSoft documents containerized Mule runtime deployment and image customization in its Docker deployment documentation. Kubernetes provides the mechanism for scheduling and running these consistently defined container images, as described in the Kubernetes container images documentation.