DUMPSBOSS.

MuleSoft Certified Integration Architect - Level 1

Mulesoft MCIA-Level-1

Version Demo

Total Demo Questions: 39

Total Premium Questions: 409

Buy Premium PDF

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

Topic Break Down

Topic	No. of Questions
Topic 1, Designing Application Networks	18
Topic 2, Designing APIs	40
Topic 3, Designing Integration Solutions	159
Topic 4, Anypoint Platform Architecture	192
Total	409

QUESTION NO: 1

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 is valuable when an integration landscape needs a shared, consistent representation of business data across processes. “To have consistent data structure aligned in processes” is correct because a canonical representation defines common business entities and attributes, such as customer, order, or product, independently of the individual formats used by participating applications. Mule applications can then use that common contract when exchanging information, reducing duplicated mapping decisions and making integrations easier to understand and govern.

“Because the model isolates the back-end systems and supports Mule applications from change” is also correct. A canonical model creates an abstraction layer between a consuming application and a system-specific data model. When a back-end application changes its field names, schema, or interface format, the transformation at the boundary can be updated while the shared canonical contract and dependent integration flows remain stable. This supports loose coupling, reuse, and safer evolution of an application network. MuleSoft’s API-led connectivity approach similarly emphasizes separating system-specific complexity behind reusable interfaces, while DataWeave provides the transformation capability needed to map between system formats and a common model. See MuleSoft’s API-led connectivity overview and the DataWeave documentation.

QUESTION NO: 2

An organization is creating an Anypoint Platform business-group model for two independent lines of business. Each line of business has its own development, test, and production environments, and teams must not be able to deploy or view resources belonging to the other line of business.

What are two appropriate ways to support this organizational access-control requirement? (Choose two.)

A. Create separate business groups for the two lines of business

B. Assign roles and permissions within each business group and environment

C. Use one shared administrator account for all deployment teams

D. Use API Autodiscovery to restrict user access to environments

E. Store each line of business in a separate Mule domain project

ANSWER: A B

Explanation:

Creating separate business groups for the independent lines of business provides administrative and resource separation within the Anypoint Platform organization. Each business group can contain its own environments, deployed applications, APIs, and assets according to the organization's operating model.

Assigning roles and permissions within each business group limits users to the business groups and environments where they have authorized responsibilities. For example, a deployment role can be granted for a production environment in one business group without granting visibility or deployment permission in another business group. Identity-provider

authentication establishes who users are, while Anypoint Platform roles and permissions control what those users can access. See Business Groups documentation and Roles and permissions documentation.

QUESTION NO: 3

An organization's IT team follows an API-led connectivity approach and must use Anypoint Platform to implement a System AP\ that securely accesses customer data. The organization uses Salesforce as the system of record for all customer data, and its most important objective is to reduce the overall development time to release the System API.

The team's integration architect has identified four different approaches to access the customer data from within the implementation of the System API by using different Anypoint Connectors that all meet the technical requirements of the project.

A. Use the Anypoint Connector for Database to connect to a MySQL database to access a copy of the customer data

B. Use the Anypoint Connector for HTTP to connect to the Salesforce APIs to directly access the customer data

C. Use the Anypoint Connector for Salesforce to connect to the Salesforce APIs to directly access the customer data

D. Use the Anypoint Connector tor FTP to download a file containing a recent near-real time extract of the customer data

ANSWER: C

Explanation:

Use the Anypoint Connector for Salesforce to connect to the Salesforce APIs to directly access the customer data because it is the purpose-built Mule connector for integrating with Salesforce, the authoritative system of record in this scenario. The connector provides Salesforce-specific operations for common data-access tasks, including querying, creating, updating, deleting, and retrieving Salesforce records. It also encapsulates Salesforce connectivity and authentication configuration, so developers do not need to manually construct HTTP requests, manage Salesforce endpoint details, or implement the Salesforce API interaction patterns themselves. This lets the team implement the System API more quickly while still directly accessing the current customer data held in Salesforce. Using a dedicated connector also makes the integration easier to configure, maintain, and standardize because its operations and metadata are designed around Salesforce objects and APIs. MuleSoft documents the connector's supported Salesforce operations, connection configurations, and authentication choices in its Salesforce Connector documentation. This choice also aligns with API-led connectivity: the System API encapsulates the Salesforce system of record behind a reusable, secure interface for other APIs and applications. MuleSoft's Anypoint Connectors overview describes how connectors provide prebuilt integration capabilities that reduce implementation effort.

QUESTION NO: 4

An organization uses a four (4) node customer-hosted Mule runtime cluster to host one (1) stateless API implementation. The API is accessed over HTTPS through a load balancer that uses round-robin for load distribution. Each node in the cluster has been sized to be able to accept four (4) times the current number of requests.

Two (2) nodes in the cluster experience a power outage and are no longer available. The load balancer detects the outage and blocks the two unavailable nodes from receiving further HTTP requests.

What performance-related consequence is guaranteed to happen, on average, assuming the remaining cluster nodes are fully operational?

A. 100% increase in the number of requests received by each remaining node

B. 100% increase in the average response time of the API

C. 50% reduction in the throughput of the API

D. 50% increase in the JVM heap memory consumed by each remaining node

ANSWER: A

Explanation:

100% increase in the number of requests received by each remaining node is guaranteed. Before the outage, round-robin distribution across four healthy nodes gives each node, on average, one quarter of the incoming request volume. Once the load balancer removes the two unavailable nodes from its eligible backend pool, it distributes the same incoming traffic across only two healthy nodes. Each remaining node therefore receives one half of the total request volume. Moving from one quarter to one half doubles the requests handled per surviving node, which is a 100% increase.

The API is explicitly stateless, so no request-affinity or session-replication behavior changes this basic distribution calculation. The stated node capacity also establishes that the remaining nodes can accept the increased request volume. Mule runtime clustering supports high availability through multiple runtime instances, while the load balancer is responsible for directing incoming HTTP traffic to available instances. See MuleSoft's high-availability cluster documentation and the HTTP Listener documentation for relevant runtime and HTTP endpoint concepts.

QUESTION NO: 5

A Mule application is being designed to receive nightly a CSV file containing millions of records from an external vendor over SFTP. The records from the file need to be validated, transformed, and then written to a database. Records can be inserted into the database in any order.

In this use case, what combination of Mule components provides the most effective, performant, and idiomatic (used for its intended purpose) way to write these records to the database?

A. Use a Batch Job scope to bulk insert records into the database

B. Use a Scatter-Gather to bulk insert records into the database

C. Use a Parallel For Each scope to insert records in-parallel into the database

D. Use a DataWeave map function and an Async scope to insert records in-parallel into the database

ANSWER: A

Explanation:

Use a Batch Job scope to bulk insert records into the database is the appropriate approach because Mule batch processing is specifically intended for handling large collections of records efficiently. A batch job splits the CSV-derived collection into individual records for validation and transformation while managing processing in blocks, rather than requiring the entire file's records to be handled as one synchronous operation. This makes it well suited to nightly files containing millions of entries.

After records are prepared, a batch aggregator can group them into suitably sized collections and invoke the Database Connector bulk operation. Bulk database operations reduce per-record connector and database round trips, improving throughput substantially compared with issuing an individual insert for every record. The ability to process records independently and without preserving their original order aligns directly with the stated requirement. Batch processing also provides operational features useful for large loads, including configurable block size, record-level processing, and reporting on successful or failed records. MuleSoft documents batch processing as the mechanism for streaming and processing large messages efficiently, and documents bulk database operations for executing a statement against multiple input parameter sets. See Mule batch processing and Database Connector bulk operations.

QUESTION NO: 6

A supplier uses Anypoint Partner Manager to exchange EDI documents with several trading partners. The integration architect needs to define capabilities that support reliable B2B partner onboarding and operational visibility for document exchanges.

What are two capabilities that Anypoint Partner Manager can provide for this B2B integration scenario? (Choose two.)

A. Onboard and manage trading-partner configurations

B. Monitor B2B message transmissions and processing status

C. Host Mule application source code branches and pull requests

D. Apply API Manager policies to HTTP API endpoints

E. Replace the Mule runtime used to execute integration applications

ANSWER: A B

Explanation:

Anypoint Partner Manager supports onboarding and managing trading partners, including the partner-specific configuration needed to exchange B2B messages. This enables an organization to maintain partner profiles and configure the agreements and connection information required for each trading relationship.

Anypoint Partner Manager also provides visibility into B2B message transmissions and processing status. This assists operations teams in monitoring document exchanges, identifying failed transmissions, and investigating the processing history of partner messages.

Partner Manager is designed for B2B integration workflows and does not replace source control, application runtime deployment, or API policy enforcement. See the Anypoint Partner Manager documentation.

QUESTION NO: 7

An organization is defining an API governance strategy. The organization wants API design standards to be consistently checked before API specifications are published for reuse by other project teams.

What are two capabilities provided by Anypoint API Governance? (Choose two.)

A. Create and share rulesets for API specifications

B. View conformance results for governed API specifications

C. Deploy Mule applications to CloudHub

D. Replace API Manager runtime policies

E. Store message payloads for failed API requests

ANSWER: A B

Explanation:

API Governance enables organizations to define and apply rulesets that assess API specifications against established standards. A ruleset can enforce conventions for areas such as naming, required documentation, security schemes, versioning, and response definitions. Publishing and sharing the ruleset enables teams to use a common set of API design controls across projects.

API Governance also provides conformance results that identify which rules pass or fail for a governed API specification. Teams can use these results to correct design issues before an API is released, improving consistency and reducing the risk that consumers encounter incomplete or nonstandard API contracts. API Governance evaluates API specifications; it does not deploy Mule applications or replace runtime API policies. See API Governance documentation.

QUESTION NO: 8

An Organization has previously provisioned its own AWS VPC hosting various servers. The organization now needs to use Cloudhub to host a Mule application that will implement a REST API once deployed to Cloudhub, this Mule application must be able to communicate securely with the customer-provisioned AWS VPC resources within the same region, without being interceptable on the public internet.

What Anypoint Platform features should be used to meet these network communication requirements between Cloudhub and the existing customer-provisioned AWS VPC?

A. Add a Mulesoft hosted Anypoint VPC configured and with VPC Peering to the AWS VPC

B. Configure an external identity provider (IDP) in Anypoint Platform with certificates from the customer provisioned AWS VPC

C. Add a default API Whitelisting policy to API Manager to automatically whitelist the customer provisioned AWS VPC IP ranges needed by the Mule applicaton

D. Use VM queues in the Mule application to allow any non-mule assets within the customer provisioned AWS VPC to subscribed to and receive messages

ANSWER: A

Explanation:

Add a Mulesoft hosted Anypoint VPC configured and with VPC Peering to the AWS VPC is the appropriate solution because it establishes private network connectivity between CloudHub workers and resources in the organization's AWS VPC. Anypoint VPC provides an isolated networking environment for CloudHub deployments, including private IP addressing and controlled network routes. AWS VPC peering then connects that Anypoint VPC to the customer-managed VPC within the same AWS Region.

After peering is established, the required route tables and security controls can be configured so that the Mule application reaches private IP addresses of the AWS-hosted servers. Traffic follows AWS private networking rather than traversing the public internet, satisfying the requirement for communication that is not publicly interceptable. This design is specifically intended for CloudHub applications that must integrate with private infrastructure hosted in AWS, while retaining network segmentation and customer-controlled access rules.

MuleSoft documents Anypoint VPC as the CloudHub networking capability used to extend private connectivity to enterprise resources and supports AWS VPC peering as a connectivity method. See Anypoint VPC documentation and VPC connectivity methods.

QUESTION NO: 9

An organization is implementing a CI/CD pipeline for Mule applications. The pipeline must ensure that application artifacts are tested and that environment-specific credentials are not committed in source control.

What are two recommended practices for this pipeline? (Choose two.)

A. Run MUnit tests as part of the Maven build

B. Externalize credentials from application source code

C. Commit production passwords in the application's property files

D. Manually modify the deployed application JAR file after each deployment

E. Skip tests when deploying a new application version

ANSWER: A B

Explanation:

Running MUnit tests as part of the Maven build provides automated validation of Mule application behavior before an artifact is deployed. MUnit integrates with Maven-based builds, allowing test failures to stop the pipeline before a deployment stage is reached. This helps ensure that regressions are detected consistently and early.

Credentials and other sensitive configuration values should be externalized rather than committed as plain-text values in application source files. Mule applications can use secure configuration properties, while deployment automation can supply environment-specific values through controlled deployment configuration and secret-management processes. This separates deployable code from confidential environment data. See MUnit Maven support and Secure Configuration Properties.

QUESTION NO: 10

Which Mulesoft feature helps users to delegate their access without sharing sensitive credentials or giving full control of accounts to 3rd parties?

A. Secure Scheme

B. client id enforcement policy

C. Connected apps

D. Certificates

ANSWER: C

Explanation:

Connected apps provide the Anypoint Platform mechanism for delegating access to an external application without disclosing a user's sensitive credentials or granting unrestricted control of the user account. A connected app is registered in Anypoint Access Management and uses standards-based authorization, including OAuth 2.0 and OpenID Connect, to request the access it needs. The user can authorize the application to act within the approved scope, rather than sharing an Anypoint username and password with a third party.

This model supports least-privilege access because permissions can be scoped to the intended organization, environment, and actions. It also provides lifecycle control: administrators or users can revoke the app's access when it is no longer needed, and activity performed through the app can be audited. This makes connected apps appropriate for integrations, automation tools, and partner applications that need programmatic access to Anypoint Platform while preserving accountability and protecting account credentials. MuleSoft documents connected apps as the secure framework for external applications to integrate with Anypoint Platform APIs using delegated authorization. See Connected Apps Overview and Connected Apps for Developers.

QUESTION NO: 11

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: B D

Explanation:

Adding the dependency jar file to the Java classpath through JVM parameters makes its classes available from a parent class loader that Mule connector class loaders can resolve. This is appropriate when the library is managed at the runtime level and is intended to be available to deployed applications or extensions that need it. Mule's class-loading model delegates class lookup through class-loader relationships, so a library made available on the runtime/JVM classpath can be found by the Java Invoke connector when it loads the target class.

Configuring the dependency as a shared library in the project POM is the application-level way to make a dependency visible to Mule plugins and connectors. Shared libraries are specifically designed for dependencies that must be accessed both by the application and by extension class loaders. Declaring the artifact under the Mule Maven Plugin's shared-library configuration ensures that the dependency is packaged and exposed according to Mule's shared-library class-loading behavior, allowing Java Invoke to resolve and call the static method in the dependency JAR.

See MuleSoft's Shared Libraries documentation and its Mule Classloader documentation for the relevant class-loading model and configuration guidance.

QUESTION NO: 12

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 backend systems and supporting Mule applications from change

ANSWER: A E

Explanation:

A canonical data model provides a shared, stable representation of business data—such as customers, orders, and products—for integrations across the enterprise. "To have consistent data structure aligned in processes" is correct because this common representation establishes a consistent data contract between APIs and consuming applications. Each integration can map its local application-specific format to or from the canonical structure, allowing business processes to use data with predictable fields, meanings, and relationships. This improves reuse and reduces the number of point-to-point transformations required as more systems are connected.

"Because the model isolates backend systems and supporting Mule applications from change" is also correct. The canonical model acts as an abstraction layer between the varying schemas of systems of record and the APIs or applications that consume their data. When a backend system changes its internal data structure, the impact can typically be contained in the mapping at that system boundary rather than forcing changes throughout all downstream integrations. This supports the API-led connectivity goal of decoupling systems and managing change through well-defined interfaces. MuleSoft describes this approach to abstraction and reuse in its API-led connectivity overview and discusses reusable integration assets in the Anypoint Platform API-led connectivity documentation.

QUESTION NO: 13

An organization is using Mulesoft cloudhub and develops API's in the latest version. As a part of requirements for one of the API's, third party API needs to be called. The security team has made it clear that calling any external API needs to have include listing

As an integration architect please suggest the best way to accomplish the design plan to support these requirements?

A. Implement includelist IP on the cloudhub VPC firewall to allow the traffic

B. Implement the validation of includelisted IP operation

C. Implement the Any point filter processor to implement the include list IP

D. Implement a proxy for the third party API and enforce the IPinclude list policy and call this proxy from the flow of the API

ANSWER: D

Explanation:

Implementing a proxy for the third-party API and enforcing an IP allowlist policy centralizes the security control at an API-management boundary. The consuming API calls the managed proxy rather than directly integrating with the external endpoint. This gives the organization a consistent, reusable place to govern access to the third-party capability, apply an IP allowlist, and maintain visibility into requests through Anypoint API Manager. It also decouples consuming implementations

from the third party's endpoint details, so changes to routing, credentials, policies, or the downstream service can be managed at the proxy layer without changing every caller.

MuleSoft API Manager supports applying policies to API instances, including IP allowlisting controls that restrict which client source addresses may access a protected API. Using a proxy/API instance for the external service therefore aligns the design with API-led governance: access restrictions are declared and enforced through the platform rather than embedded ad hoc in individual application flows. This pattern provides a clear policy enforcement point and makes the security requirement auditable and consistently administered. See MuleSoft's IP Allowlist policy documentation and proxy application documentation.

QUESTION NO: 14

An organization exposes APIs to external consumers. A downstream customer system is occasionally slow or unavailable, causing API request threads to remain occupied while waiting for responses.

What are two benefits of implementing a circuit-breaker pattern for calls to the downstream system? (Choose two.)

A. Fail requests quickly when the downstream system is known to be unhealthy

B. Reduce the risk of cascading failures caused by repeated dependency calls

C. Guarantee successful completion of every downstream request

D. Eliminate the need to configure HTTP request timeouts

E. Convert synchronous downstream operations into XA transactions

ANSWER: A B

Explanation:

A circuit breaker can fail requests quickly after a configured threshold of downstream failures is reached. Instead of continuing to wait for a dependency that is known to be unhealthy, the API can return a controlled error or invoke an appropriate fallback behavior. This protects request-processing resources and gives consumers more predictable failure behavior.

A circuit breaker can also reduce cascading failures. By limiting calls to an unhealthy dependency, it prevents repeated requests from increasing load on the downstream system while it is recovering and helps preserve capacity in the calling Mule application for other operations.

A circuit breaker does not guarantee that the downstream system will recover, nor does it replace the need for appropriate timeouts, error handling, monitoring, and retry strategies. MuleSoft provides a Custom Circuit Breaker policy for applicable managed API scenarios.

QUESTION NO: 15

An organization plans to expose a set of APIs to mobile applications and browser-based clients. The APIs are accessed through a common public domain and the organization needs to protect its backend services from excessive request volumes.

What are two benefits of applying rate limiting at the API boundary? (Choose two.)

A. Protect backend services from excessive request volumes

B. Enforce approved consumption limits for API clients

C. Encrypt request payloads between the client and API

D. Replace authentication of API clients

E. Transform JSON payloads into the backend data model

ANSWER: A B

Explanation:

Rate limiting restricts the number of requests that a client can make during a configured period. This helps protect backend systems and Mule application resources from accidental or intentional request bursts that could otherwise consume available capacity and degrade service for other clients.

Rate limiting can also establish enforceable consumption limits for distinct API clients when used with an identity mechanism such as client ID enforcement. This enables the organization to provide predictable access levels and apply different limits according to the client application's approved service tier. It does not encrypt payloads or replace client authentication and authorization controls. See Rate Limiting and Throttling policies and Client ID-based policies.

QUESTION NO: 16

A Kubernetes controller automatically adds another pod replica to the resource pool in response to increased application load.

Which scalability option is the controller implementing?

A. Down

B. Diagonal

C. Vertical

D. Horizontal

ANSWER: D

Explanation:

Horizontal is correct because the controller responds to increased demand by adding another pod replica, increasing the number of running application instances. This is horizontal scaling (also called scaling out): capacity grows by distributing work across additional, typically equivalent, compute instances rather than increasing the CPU, memory, or other resources assigned to one existing instance. In Kubernetes, a HorizontalPodAutoscaler can automatically adjust the number of Pods in a workload such as a Deployment based on observed metrics, including CPU utilization, memory utilization, or custom metrics. The controller continually evaluates the configured target metric and changes the desired replica count within its allowed limits, enabling the application to handle higher load while maintaining availability and throughput. This behavior aligns with cloud-native architecture practices because independently replicated, stateless workloads can be scaled across nodes and can share incoming traffic through a Service or ingress layer. MuleSoft integration applications deployed as containerized services can use this same approach when their processing demand varies. See the Kubernetes Horizontal Pod Autoscaling documentation and its workload autoscaling concepts.

QUESTION NO: 17

A Mule application is deployed to a cluster of two (2) customer-hosted Mule runtimes. Currently, the node named Alice is the primary node and the node named Bob is the secondary node. The Mule application has a flow that polls a directory on a file system for new files.

The primary node Alice fails for an hour and then is restarted.

After the Alice node completely restarts, from what node are the files polled, and what node is now the primary node for the cluster?

A. Files are polled from the Bob node Bob is now the primary node

B. Files are polled from the Bob node Alice is now the primary node

C. Files are polled from the Alice node Alice is now the primary node

D. Files are polled from the Alice node Bob is now the primary node

ANSWER: A

Explanation:

Files are polled from the Bob node Bob is now the primary node. In a customer-hosted Mule high-availability cluster, only the primary node runs polling message sources, such as a File connector listener that checks a directory for new files. This primary-only behavior prevents multiple nodes from simultaneously consuming the same input and producing duplicate processing.

When Alice, the original primary node, fails, the surviving node Bob is promoted to primary so that polling and message processing can continue. Bob retains the primary role after Alice has restarted and successfully rejoined the cluster. The returning node joins as a secondary node rather than automatically reclaiming its former primary status. Therefore, the File connector polling activity continues on Bob after Alice's recovery, and Bob remains responsible for the cluster's primary-node duties unless Bob later fails or another cluster transition occurs.

MuleSoft documents the primary-node role and failover behavior for customer-hosted high-availability deployments in its Mule High Availability documentation. File polling behavior is documented in the File Connector documentation.

QUESTION NO: 18

An organization if struggling frequent plugin version upgrades and external plugin project dependencies. The team wants to minimize the impact on applications by creating best practices that will define a set of default dependencies across all new and in progress projects.

How can these best practices be achieved with the applications having the least amount of responsibility?

A. Create a Mule plugin project with all the dependencies and add it as a dependency in each application's POM.xml file

B. Create a mule domain project with all the dependencies define in its POM.xml file and add each application to the domain Project

C. Add all dependencies in each application's POM.xml file

D. Create a parent POM of all the required dependencies and reference each in each application's POM.xml file

ANSWER: D

Explanation:

Create a parent POM of all the required dependencies and reference each in each application's POM.xml file is the appropriate approach because Maven inheritance centralizes common build configuration, dependency versions, plugin versions, repositories, and dependency management in one reusable parent project. Each Mule application then declares that parent in its `<parent>` section, rather than independently maintaining the shared dependency configuration. This gives the organization one governed location for defining approved default dependencies and for coordinating plugin or library version upgrades across both current and future applications.

A parent POM can use Maven's `<dependencyManagement>` section to control versions without necessarily forcing every child project to include every dependency. Applications retain only the minimal responsibility of inheriting the organizational parent and declaring any dependencies they actually use. When a centrally managed version needs to change, the organization updates and publishes the parent POM, then applications can adopt that controlled parent version through their normal release process. This structure supports consistent, repeatable Mule project builds and reduces configuration drift among teams. MuleSoft documents Maven-based project and dependency management in its build tooling guidance, and Maven documents the parent-child inheritance model at Maven: Introduction to the POM and Mule Maven Plugin documentation.

QUESTION NO: 19

An organization is in the process of building automated deployments using a CI/CD process. As a part of automated deployments, it wants to apply policies to API Instances.

What tool can the organization use to promote and deploy API Manager policies?

A. Anypoint CLI

B. MUnit Maven plugin

C. Mule Maven plugin

D. Runtime Manager agent

ANSWER: A

Explanation:

Anypoint CLI is the appropriate tool because it enables API-management tasks to be executed noninteractively from scripts and CI/CD pipeline jobs. In particular, its API Manager commands can manage API instances and automate policy-related operations, allowing a team to apply a consistent policy configuration as APIs are promoted through development, test, staging, and production environments. A pipeline can authenticate to Anypoint Platform using suitable credentials, target the required organization and environment, identify the relevant API instance, and run the required commands as a repeatable deployment step. This removes the need for a release engineer to manually configure policies in the API Manager user interface and helps ensure that environment promotions follow the same governed process each time. Policy automation is particularly useful for consistently enforcing controls such as client access requirements, rate limiting, headers, and other gateway behaviors alongside API releases. MuleSoft documents Anypoint CLI as a command-line interface for interacting with Anypoint Platform and provides API Manager command documentation for automating API management operations. See the Anypoint CLI documentation and the API Manager commands reference.

QUESTION NO: 20

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 cannot be created for flows that raise Mule errors internally or receive an incoming event with an existing Mule error.

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

C. The recorder support loops where the structure of the data been tested

D. A recorded flow execution can end successfully while its result does not 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:

“Tests cannot be created for flows that raise Mule errors internally or receive an incoming event with an existing Mule error” is a key Test Recorder limitation. The recorder creates a test by executing the selected flow and collecting the processor interactions and events needed to generate MUnit behavior. Consequently, recording requires a successful execution path from a valid starting event. Before recording, the team should therefore use an input that does not already carry an error and select a scenario that does not raise a Mule error during the capture.

“A recorded flow execution can end successfully while its result does not reach its destination because the application is killed” is also important when the flow contains asynchronous work. The recorder’s capture lifecycle can finish once the main flow execution completes, while work dispatched independently may still be in progress. If Studio stops or restarts the application as part of ending the recording session, that work can be terminated before it reaches an external target. Architects should account for this behavior when recording flows with asynchronous processing and verify the generated test represents the intended observable behavior. See the official MUnit Test Recorder documentation and Mule’s Async Scope reference.

QUESTION NO: 21

An organization maintains a REST API that is consumed by several internal applications. A new business requirement introduces a change that removes a required request field and changes the meaning of an existing response field.

What are two appropriate approaches for managing this change while minimizing disruption to existing API clients? (Choose two.)

A. Create and publish a new major version of the API contract

B. Continue operating the existing API version during a defined deprecation period

C. Update the existing API contract in place without changing its version

D. Require all clients to use the new response model immediately after deployment

E. Replace the existing endpoint with an HTTP 301 response

ANSWER: A B

Explanation:

Creating and publishing a new major version of the API is appropriate because removing a required request field or changing the meaning of a response field is a breaking contract change. Existing clients can continue to consume the prior version while they plan, test, and complete their migration to the new interface.

Continuing to operate the existing API version during a defined deprecation period is also appropriate. This provides consumers with a stable migration window and allows the API provider to communicate a retirement date, usage guidance, and migration instructions. An API provider should avoid silently changing an existing contract in a way that causes deployed clients to fail or behave incorrectly.

MuleSoft documents API versioning and lifecycle management in the API versioning documentation and provides guidance for publishing API assets through Anypoint Exchange.

QUESTION NO: 22

Which of the below requirements prevent the usage of Anypoint MQ in a company's network? (Choose two answers)

A. single message payload can be up to 15 MB

B. payloads must be encrypted

C. the message broker must be hosted on premises

D. support for point-to-point messaging

E. ability for a third party outside the company's network to consume events from the queue

ANSWER: A C

Explanation:

Anypoint MQ cannot meet a requirement that a single message payload be up to 15 MB because its documented maximum message size is 10 MB. This limit applies to the complete message sent to Anypoint MQ, so an integration handling individual payloads larger than that threshold must use a different design. Common alternatives include storing the large object in external storage and placing a reference, identifier, or URL in the queue message, or choosing a messaging technology that supports the required message size.

The requirement that the message broker must be hosted on premises also prevents using Anypoint MQ. Anypoint MQ is a fully managed, cloud-based MuleSoft service hosted as part of the Anypoint Platform; customers use its queues and exchanges through the service rather than installing and operating an Anypoint MQ broker in their own data center or private network. Therefore, an architecture with a strict on-premises broker-hosting constraint needs an on-premises messaging product or a separately managed broker deployment.

MuleSoft documents the message-size limit and the managed-service model in the Anypoint MQ documentation and its Anypoint MQ FAQ.

QUESTION NO: 23

What is true about the network connections when a Mule application uses a JMS connector to interact with a JMS provider (message broker)?

A. To complete sending a JMS message, the JMS connector must establish a network connection with the JMS message recipient

B. To receive messages into the Mule application, the JMS provider initiates a network connection to the JMS connector and pushes messages along this connection

C. The JMS connector supports both sending and receiving of JMS messages over the protocol determined by the JMS provider

D. The AMQP protocol can be used by the JMS connector to portably establish connections to various types of JMS providers

ANSWER: C

Explanation:

The JMS connector supports both sending and receiving of JMS messages over the protocol determined by the JMS provider. JMS is an API specification rather than a wire-level network protocol. A Mule application uses the JMS connector together with a provider-specific JMS client library and a configured JMS `ConnectionFactory`. That provider client is responsible for establishing and managing the connection to the message broker using the broker's supported transport and protocol implementation. For example, the actual connection behavior is supplied by the selected JMS provider's client, not by a direct Mule-to-recipient connection.

Within Mule, JMS operations can publish messages to a destination, while JMS listeners consume messages from queues or topics. Both activities use the configured provider connection factory and its connection settings. This abstraction is the principal value of JMS: Mule code and flow configuration interact through standard JMS concepts such as connections, sessions, destinations, producers, and consumers, while the vendor-specific client handles the underlying broker communication. MuleSoft documents that the connector requires a JMS provider and connection factory configuration, and its listener and publish capabilities operate through that configuration. See the MuleSoft JMS Connector documentation and the Jakarta Messaging specification.

QUESTION NO: 24

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,

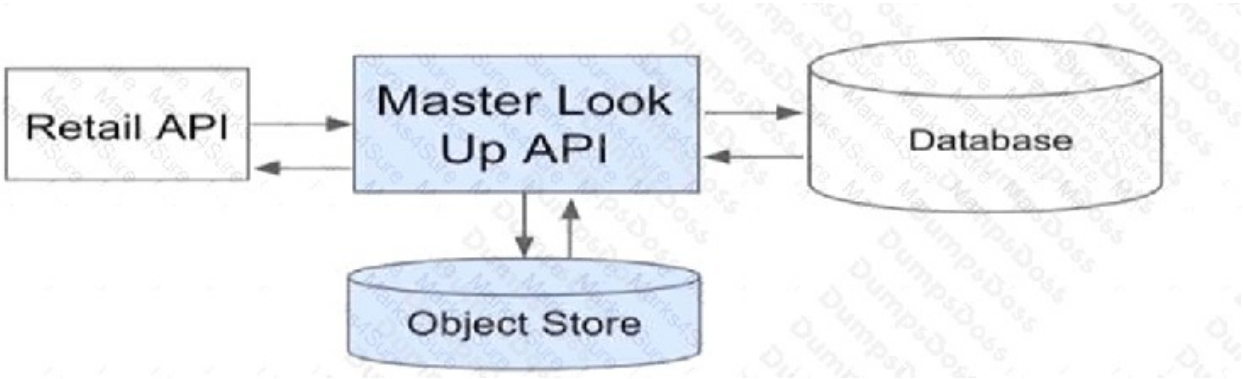
The API uses a persistent object-store. This API will be used by almost all other APIs to provide master lookup data.

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 GET endpoints allows the platform to retain cacheable HTTP responses and serve subsequent equivalent lookup requests without invoking the Master Look Up API implementation. Since this API is heavily reused and returns master lookup data, response caching reduces repeated cache-miss processing and, most importantly, avoids database calls for requests whose representations are already cached. The policy is especially appropriate for GET operations because these operations are intended to retrieve data and can use HTTP cache controls such as expiration and validation semantics. MuleSoft documents the HTTP Caching policy and its applicability to GET responses in HTTP Caching policy documentation.

Upgrading from 0.1 vCore to 0.2 vCore provides each CloudHub worker with additional compute and memory resources. This improves the API's capacity to process concurrent requests, serialize and deserialize payloads, manage local runtime activity, and maintain cache-related processing with less resource contention. It complements HTTP response caching: caching reduces demand on the database, while additional worker capacity helps the application remain responsive for cache misses, cache refreshes, and traffic that cannot be served from the HTTP cache. CloudHub worker sizing and available vCore options are described in the CloudHub architecture documentation.

QUESTION NO: 25

An organization's IT team must secure all of the internal APIs within an integration solution by using an API proxy to apply required authentication and authorization policies.

Which integration technology, when used for its intended purpose, should the team choose to meet these requirements if all other relevant factors are equal?

A. API Management (APIM)

B. Robotic Process Automation (RPA)

C. Electronic Data Interchange (EDI)

D. Integration Platform-as-a-service (PaaS)

ANSWER: A

Explanation:

API Management (APIM) is the appropriate technology because its intended purpose is to govern, secure, and manage APIs throughout their lifecycle. In MuleSoft, API Manager can apply policies at the API proxy or gateway layer, separating cross-cutting controls from the backend implementation. This allows the organization to consistently enforce required authentication and authorization for internal APIs without requiring each underlying API implementation to duplicate the same security logic. Policies can be configured centrally and applied to managed API instances, supporting controls such as client application authentication, OAuth-based access-token validation, and authorization-related enforcement. An API proxy is especially useful when an existing API must be managed and protected without modifying its original implementation, because the proxy receives requests, applies the configured policies, and forwards permitted traffic to the upstream service. This architecture gives IT teams centralized visibility and control over API access while preserving a clear separation between API security governance and integration or business-process logic. MuleSoft documents that API Manager is used to manage API instances and apply policies, including through proxy deployments. See the MuleSoft API Manager documentation and the API proxy deployment documentation.

QUESTION NO: 26

A Mule application accepts order requests through an HTTP Listener and invokes a downstream payment service. The integration team is writing MUnit tests for the application.

What are two appropriate uses of MUnit mocking in these tests? (Choose two.)

A. Return a controlled response from the outbound HTTP Request operation

B. Simulate an error from the downstream payment service

C. Create a real payment-service account for every test execution

D. Deploy the application automatically to production

E. Generate API analytics in API Manager

ANSWER: A B

Explanation:

MUnit mocking is appropriate for replacing the outbound HTTP Request operation with a controlled response. This isolates the Mule application's own transformation and orchestration logic from the availability, latency, and changing data of the payment service. The test can return a representative successful response, a timeout, or an error response as needed to validate the relevant application behavior.

Mocking is also appropriate for simulating an error from the downstream payment service in order to validate the application's error-handling path. The test can verify that the application returns the expected error response, logs or records the failure as required, and does not execute unintended downstream behavior. MUnit mocks message processors; they do not provision external services or execute production traffic. See MUnit Mocking documentation and MUnit testing concepts.

QUESTION NO: 27

An organization is implementing event-driven integration between order management and fulfillment applications. The order management application must publish order-created events without waiting for fulfillment processing to complete. Fulfillment processing must continue if the order management application is temporarily unavailable after publishing an event.

What are two characteristics of an asynchronous messaging design that support these requirements? (Choose two.)

A. A durable message broker decouples the event publisher from the event consumer

B. The publisher can continue without waiting for fulfillment processing to complete

C. The consumer must be online when the publisher sends every event

D. The publisher and consumer must use the same runtime deployment

E. The broker guarantees that consumers never receive duplicate events

ANSWER: A B

Explanation:

A durable message broker decouples the publishing application from the consuming application. After an order-created event is accepted by the broker, the publisher does not need to remain available while the fulfillment application processes the event. The consumer can retrieve and process the message later.

Asynchronous messaging also allows the publisher to avoid waiting for fulfillment processing to complete before continuing its own work. This improves temporal decoupling between the systems and is appropriate when fulfillment does not need to return an immediate result to the order-management process.

The architecture must still address message delivery, error handling, duplicate processing, monitoring, and retention requirements. MuleSoft documents queues and exchanges in the Anypoint MQ documentation.

QUESTION NO: 28

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 inside
the 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 tested
changes inside the iteration

D. A recorded flow execution ends successfully but the result does
not 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:

MUnit Test Recorder is intended to generate an initial unit-test structure from a successful, observable flow execution. Consequently, *Tests for flows cannot be created with Mule errors raised inside the flow or already existing in the incoming event* is a key constraint: recorder-generated tests require a normal execution path and cannot use an execution that starts with, or reaches, a Mule error as its recording basis. Error scenarios should instead be covered by manually authored MUnit tests that invoke the relevant flow and explicitly validate the expected error behavior.

A recorded flow execution ends successfully but the result does not reach its destination because the application is killed is also an important consideration. A recorder capture reflects what it observed while the application was running; it is not a durable, end-to-end delivery guarantee. If the application is terminated after processing appears successful but before an outbound operation completes, the capture can still look successful even though no result is delivered. Architects should therefore use recorder output as a starting point, then add deterministic mocks and assertions for outbound behavior and failure handling. MuleSoft documents recorder usage and its constraints in the MUnit Test Recorder documentation; use the MUnit testing concepts documentation when completing the generated tests.

QUESTION NO: 29

An organization wants Mule applications to provide useful operational information to a centralized monitoring team. The team must be able to trace a business request across multiple Mule applications and investigate failed downstream calls.

What are two recommended practices for supporting these operational requirements? (Choose two.)

A. Include the Mule event correlation ID in application log messages

B. Log relevant error context without exposing sensitive data

C. Log client secrets and authorization tokens to simplify troubleshooting

D. Disable error logging to reduce worker resource consumption

E. Use a different correlation ID for every processor in a flow

ANSWER: A B

Explanation:

Including the Mule event correlation ID in application log messages is recommended because Mule assigns a correlation ID to an event and propagates it as the event moves through flows. Logging this identifier enables operations teams to find related log entries for the same request, including entries emitted by different flows and applications when the correlation information is propagated.

Logging relevant error context without exposing sensitive data is also recommended. Useful context can include the target system name, operation, error type, response status, and a business-safe identifier. Credentials, access tokens, full payment details, and other sensitive payload information should not be written to logs. This improves diagnosability while supporting security and compliance requirements.

See MuleSoft's logging in Mule applications documentation and error handling documentation.

QUESTION NO: 30

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 is a key benefit of using a MuleSoft-hosted runtime plane, such as CloudHub. MuleSoft provides and manages the underlying runtime infrastructure, so an organization can deploy Mule applications without first procuring servers, installing and configuring runtimes, establishing baseline platform operations, or building its own runtime-management environment. This lets teams focus sooner on implementing integrations and APIs that deliver business value.

Reduced IT operations effort is also a primary reason to choose a MuleSoft-hosted runtime plane. With CloudHub, MuleSoft operates the infrastructure and provides managed runtime capabilities, including application deployment and monitoring through Anypoint Platform, platform-level availability features, and runtime patching or upgrades within the service model. Customer teams still own application design, configuration, security responsibilities appropriate to their deployment, and operational observability, but they do not need to operate the full underlying runtime infrastructure themselves. This managed-service approach is especially useful when an organization wants to standardize deployment operations while

minimizing the platform administration burden. See MuleSoft's CloudHub 2.0 documentation and the Runtime Manager documentation.

QUESTION NO: 31

A new Mule application has been deployed through Runtime Manager to CloudHub 1.0 using a CI/CD pipeline with sensitive properties set as cleartext. The Runtime Manager Administrator opened a high priority incident ticket about this violation of their security requirements indicating

these sensitive properties values must not be stored or visible in Runtime Manager but should be changeable in Runtime Manager by Administrators with proper permissions.

How can the Mule application be deployed while safely hiding the sensitive properties?

A. the application

B. Define the sensitive values as secure application properties in Runtime Manager for the Mule application.

C. wrapper.conf file used by the CI/CD pipeline scripts

D. pipeline scripts

ANSWER: B

Explanation:

Define the sensitive values as secure application properties in Runtime Manager when deploying the Mule application. CloudHub securely stores the value and masks it in Runtime Manager after it is saved, so an administrator cannot retrieve or view the previously entered cleartext value. Authorized Runtime Manager users can still replace the value by entering a new one during an application-property update, satisfying the requirement for controlled operational changes without exposing the secret in the console.

The Mule application should reference the property normally through Mule property placeholders, such as `${database.password}`. The secure designation is a Runtime Manager deployment configuration concern: it protects the deployed property value in the CloudHub management experience while allowing the application to resolve the configured value at runtime. Access to modify application settings must be governed through Anypoint Platform roles and permissions, ensuring that only approved administrators can update the secure values.

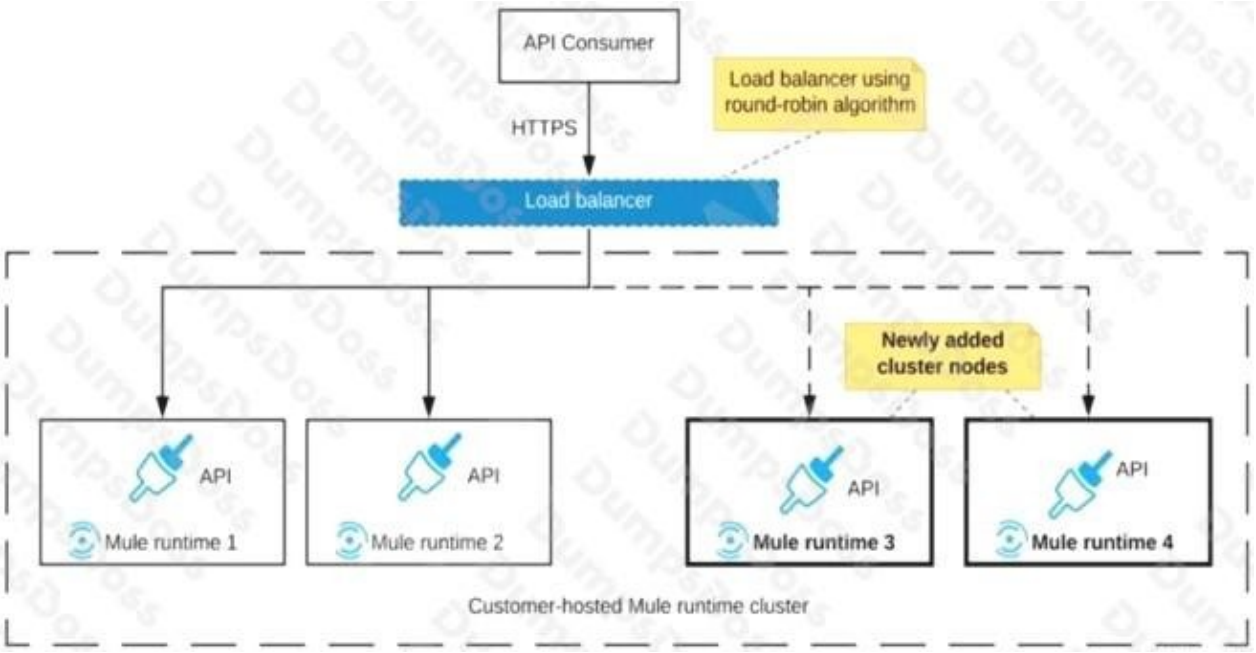
MuleSoft documents secure application properties and their masking behavior in the CloudHub application properties documentation. For the related Runtime Manager deployment and property-management workflow, see Deploying applications to CloudHub.

QUESTION NO: 32

Refer to the exhibit. An organization uses a 2-node Mule runtime cluster to host one stateless API implementation. The API is accessed over HTTPS through a load balancer that uses round-robin for load distribution.

Two additional nodes have been added to the cluster and the load balancer has been configured to recognize the new nodes with no other change to the load balancer.

What average performance change is guaranteed to happen, assuming all cluster nodes are fully operational?



A. 50% reduction in the response time of the API

B. 100% increase in the throughput of the API

C. 50% reduction in the JVM heap memory consumed by each node

D. 50% reduction in the number of requests being received by each node

ANSWER: D

Explanation:

With a fixed incoming request rate and a round-robin load-balancing policy, adding two fully operational nodes changes the target pool from two nodes to four nodes. Round robin distributes successive requests evenly across the eligible targets. Before the expansion, each node receives, on average, one-half of the requests. After it, each node receives, on average, one-quarter of the requests. Therefore, the average request volume received by an individual node falls from 50% to 25% of the total traffic, which is a 50% reduction per node.

This conclusion follows directly from the configured routing policy and does not depend on assumptions about API processing time, downstream-system capacity, JVM behavior, message size, or whether the application is CPU- or I/O-bound. Because the API implementation is stateless, requests can be handled by any cluster member without session-affinity requirements changing the distribution. MuleSoft describes clustering as a way to distribute application processing among multiple runtime instances, while a load balancer routes requests to the available instances. See MuleSoft's CloudHub architecture documentation and the Runtime Manager Agent documentation for Mule runtime deployment and operational context.

QUESTION NO: 33

What is an example of data confidentiality?

A. Signing a file digitally and sending it using a file transfer mechanism

B. Encrypting a file containing personally identifiable information (PII)

C. Providing a server's private key to a client for secure decryption of data during a two-way SSL handshake

D. De-masking a person's Social Security number while inserting it into a database

ANSWER: B

Explanation:

Encrypting a file containing personally identifiable information (PII) is an example of data confidentiality because encryption transforms readable sensitive data into ciphertext that cannot be understood without the appropriate decryption key. This protects the information from unauthorized disclosure if the file is intercepted, copied, accessed by an unauthorized party, or stored in an insecure location. PII, such as names, government identifiers, addresses, or account information, requires particular protection because exposure can cause privacy harm, identity theft, and regulatory noncompliance. In an integration architecture, encryption should be applied to sensitive data both in transit and at rest as appropriate to the data flow and threat model. MuleSoft applications can support this protection through the Cryptography Module, which provides encryption and decryption operations, while platform and deployment controls can help protect keys and access to encrypted content. Confidentiality is specifically concerned with ensuring that information is available only to authorized users and systems; encrypting PII directly serves that goal. See MuleSoft's Cryptography Module documentation and the NIST definition of confidentiality.

QUESTION NO: 34

An XA transaction Is being configured that involves a JMS connector listening for Incoming JMS messages. What is the meaning of the timeout attribute of the XA transaction, and what happens after the timeout expires?

A. The time that is allowed to pass between committing the transaction and the completion of the Mule flow After the timeout, flow processing triggers an error

B. The time that Is allowed to pass between receiving JMS messages on the same JMS connection After the timeout, a new JMS connection Is established

C. The time that Is allowed to pass without the transaction being ended explicitly After the timeout, the transaction Is forcefully rolled-back

D. The time that Is allowed to pass for state JMS consumer threads to be destroyed After the timeout, a new JMS consumer thread is created

ANSWER: C

Explanation:

The time that Is allowed to pass without the transaction being ended explicitly After the timeout, the transaction Is forcefully rolled-back is correct. An XA transaction is coordinated as a single unit of work across participating transactional resources, such as the JMS resource and any other resource enlisted during processing. The transaction timeout establishes the maximum amount of time that this unit of work may remain active without successfully reaching its explicit completion boundary. It is therefore a transaction-lifetime limit, not a setting governing JMS connection reuse, consumer-thread lifecycle, or time after a commit.

If processing does not complete the XA transaction within the configured timeout, the transaction manager treats it as timed out and rolls it back. This protects consistency: JMS message consumption and the work performed by the flow remain part of the same atomic outcome. In a listener-based JMS flow, rollback allows the broker's redelivery and failure-handling behavior to apply according to the JMS configuration and broker policy. Mule documentation describes transaction management, including XA transactions and their coordinated commit or rollback behavior, at Mule Runtime Transaction Management. The XA model itself is defined by the Java Transaction API transaction-manager contract, including timeout behavior, at Jakarta TransactionManager API.

QUESTION NO: 35

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

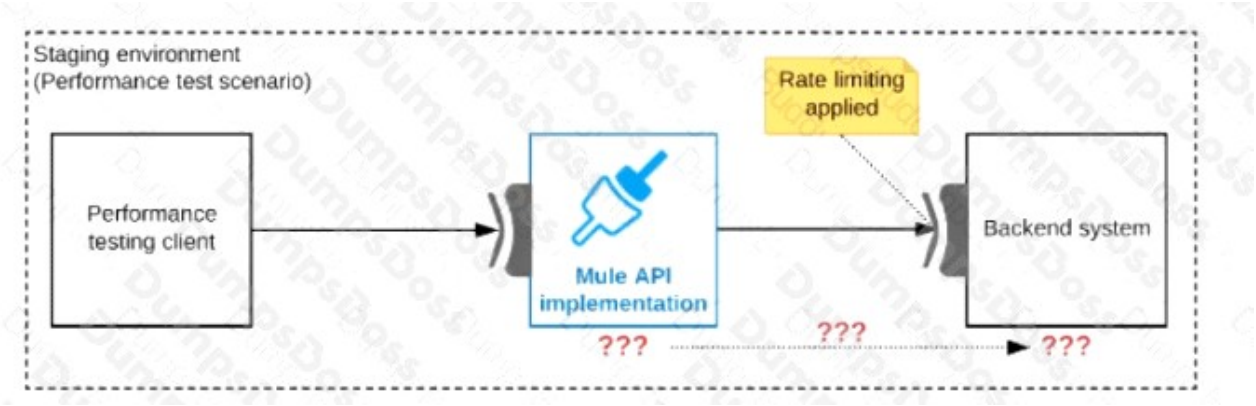
Explanation:

Synchronized data transfer is required because each incoming ANSI X12 purchase order must be transformed from its EDI representation into the supplier's ERP order model and then used to create the corresponding ERP order. This is a system-to-system exchange in which business data received in one application is reliably propagated to another application in a coordinated flow. The integration must map the purchase-order content, apply the relevant validation and processing logic, and ensure the ERP is updated with the order information.

Sharing data with external partners is also required because the supplier is exposing current inventory information to a separate organization under a trading agreement. The RESTful API provides the customer with a governed, reusable interface for requesting inventory data held by the supplier's ERP system. This partner-facing capability requires the supplier to make selected business data available externally while retaining control over access, security, and the API contract. MuleSoft supports X12 EDI message processing for partner exchanges and enables APIs to expose enterprise-system data in a managed way. See the Anypoint Partner Manager documentation and MuleSoft's API-led connectivity documentation.

QUESTION NO: 36

Refer to the exhibit.



One of the backend systems invoked by an API implementation enforces rate limits on the number of requests a particular client can make. Both the backend system and the API implementation are deployed to several non-production environments in addition to production.

Rate limiting of the backend system applies to all non-production environments. The production environment, however, does NOT have any rate limiting.

What is the most effective approach to conduct performance tests of the API implementation in a staging (non-production) environment?

A. Create a mocking service that replicates the backend system's production performance characteristics. Then configure the API implementation to use the mocking service and conduct the performance tests

B. Use MUnit to simulate standard responses from the backend system then conduct performance tests to identify other bottlenecks in the system

C. Include logic within the API implementation that bypasses invocations of the backend system in a performance test situation. Instead invoking local stubs that replicate typical backend system responses then conduct performance tests using this API Implementation

D. Conduct scaled-down performance tests in the staging environment against the rate limited backend system then upscale performance results to full production scale

ANSWER: A

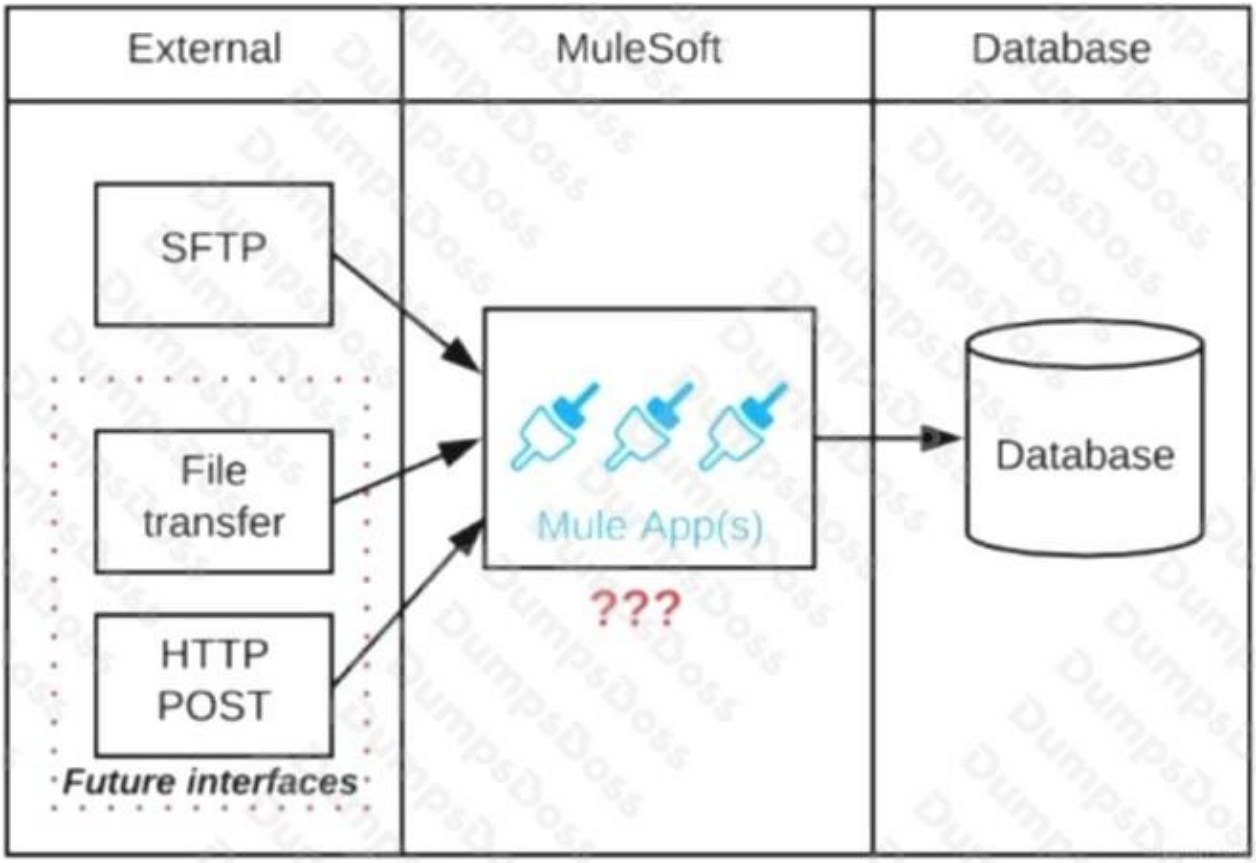
Explanation:

Create a mocking service that replicates the backend system's production performance characteristics. Then configure the API implementation to use the mocking service and conduct the performance tests. This approach lets the staging performance test exercise the API implementation's real integration path—request processing, transformations, error handling, connection behavior, and response handling—without having test throughput artificially constrained by a non-production backend rate limit. The mock must model more than a representative payload: it should reproduce production-relevant response latency, expected response codes, payload sizes, and, where applicable, realistic failure behavior. That enables load-test results to reflect how the API implementation behaves when its dependency performs as it will in production, while avoiding excessive calls to an environment that intentionally has different capacity controls. MuleSoft's mocking capabilities are intended to simulate API or service behavior for consumers when the actual implementation is unavailable or unsuitable for a given test scenario. A separately deployed mock service also permits environment-specific configuration, so the staging API can target the mock only during the controlled test while preserving the normal backend endpoint configuration elsewhere. See MuleSoft's Mocking Service documentation and its API deployment and environment configuration guidance.

QUESTION NO: 37

Refer to the exhibit. A business process involves the receipt of a file from an external vendor over SFTP. The file needs to be parsed and its content processed, validated, and ultimately persisted to a database. The delivery mechanism is expected to change in the future as more vendors send similar files using other mechanisms such as file transfer or HTTP POST.

What is the most effective way to design for these requirements in order to minimize the impact of future change?



A. Use a composite data source so files can be retrieved from various sources and delivered to a MuleSoft Batch Job for processing

B. Use a MuleSoft Scatter-Gather and a MuleSoft Batch Job to handle the different files coming from different sources

C. Create a Process API to receive the file and process it using a MuleSoft Batch Job while delegating the data save process to a System API

D. Create an API that receives the file and invokes a Process API with the data contained in the file, then have the Process API process the data using a MuleSoft Batch Job and other System APIs as needed

ANSWER: D

Explanation:

Create an API that receives the file and invokes a Process API with the data contained in the file, then have the Process API process the data using a MuleSoft Batch Job and other System APIs as needed is the most effective design because it separates the inbound transport concern from the reusable business-processing concern. The receiving API acts as an adapter at the integration boundary: it can initially be implemented with an SFTP listener and later be supplemented or replaced with file, HTTP, or other inbound mechanisms without requiring changes to validation, transformation, orchestration, or persistence logic.

The Process API owns the canonical business workflow for the file contents. It can parse records, validate them, apply business rules, and use Mule batch processing for scalable, reliable handling of large record sets. Database persistence is delegated to a System API, which encapsulates the database-specific interface and enables that system to evolve independently. This follows MuleSoft API-led connectivity principles by organizing capabilities into clear layers with well-defined contracts, maximizing reuse and reducing the scope of future changes. Mule batch processing is specifically intended to process messages as collections of records, making it appropriate for file-content processing. See MuleSoft API-led connectivity and Mule batch processing.

QUESTION NO: 38

A stock broking company makes use of CloudHub VPC to deploy Mule applications. Mule application needs to connect to a database application in the customers on-premises corporate data center and also to a Kafka cluster running in AWS VPC.

How is access enabled for the API to connect to the database application and Kafka cluster securely?

A. Set up a transit gateway to the customers on-premises corporate datacenter to AWS VPC

B. Setup AnyPoint VPN to the customer's on-premise corporate data
center and VPC peering with AWS VPC

C. Setup VPC peering with AWS VPC and the customers devices corporate data center

D. Setup VPC peering with the customers onto my service corporate data center and Anypoint VPN to AWS VPC

ANSWER: B

Explanation:

Setup AnyPoint VPN to the customer's on-premise corporate data
center and VPC peering with AWS VPC is correct because the Mule applications are deployed within an Anypoint/CloudHub VPC and need private, secure network paths to two different environments. Anypoint VPN provides an IPsec-based site-to-site VPN connection between the Anypoint VPC and the customer’s on-premises network. This enables workloads in the CloudHub VPC to reach the internal database using private address ranges, without exposing the database to the public internet.

For the Kafka cluster hosted in a separate AWS VPC, VPC peering provides private network connectivity between the Anypoint VPC and that AWS VPC. After establishing the peering connection, the required CIDR routes, security-group or firewall rules, and DNS configuration can be set so that Mule applications can reach Kafka brokers through private IP addresses. This combination matches the connectivity model for a hybrid environment: VPN for the corporate data center and VPC peering for the AWS-hosted private network. MuleSoft documents both Anypoint VPN connectivity and VPC

peering as network-configuration capabilities for Anypoint VPCs. See Anypoint VPC documentation and Anypoint VPC peering documentation.

QUESTION NO: 39

During a planning session with the executive leadership, the development team director presents plans for a new API to expose the data in the company's order database. An earlier effort to build an API on top of this data failed, so the director is recommending a design-first approach.

Which characteristics of a design-first approach will help make this API successful?

A. Building MUnit tests so administrators can confirm code coverage percentage during deployment

B. Publishing the fully implemented API to Exchange so all developers can reuse the API

C. Adding global policies to the API so all developers automatically secure the implementation before coding anything

D. Developing a specification so consumers can test before the implementation is built

ANSWER: D

Explanation:

Developing a specification so consumers can test before the implementation is built is the central benefit of a design-first approach. The team first defines the API contract, typically using RAML or OAS, including resources, methods, request and response structures, examples, error behavior, and security requirements. This shared, reviewable contract enables stakeholders and prospective consumers to validate that the API will meet real business and integration needs before implementation investment is made. In MuleSoft, the API specification can be published to Anypoint Exchange and used to generate a mock service. Consumers can then invoke the mock endpoint and begin developing and testing their integrations against the agreed interface while the implementation team builds the underlying API. This parallelizes work, exposes contract misunderstandings early, and reduces the risk of repeating a failed implementation effort caused by unclear or unvalidated requirements. The specification also becomes the authoritative artifact for governance, documentation, and later implementation validation. MuleSoft describes API design as defining the API contract before building the implementation, and its mocking capability supports testing that contract without a live backend. See MuleSoft API specification design documentation and MuleSoft mocking service documentation.