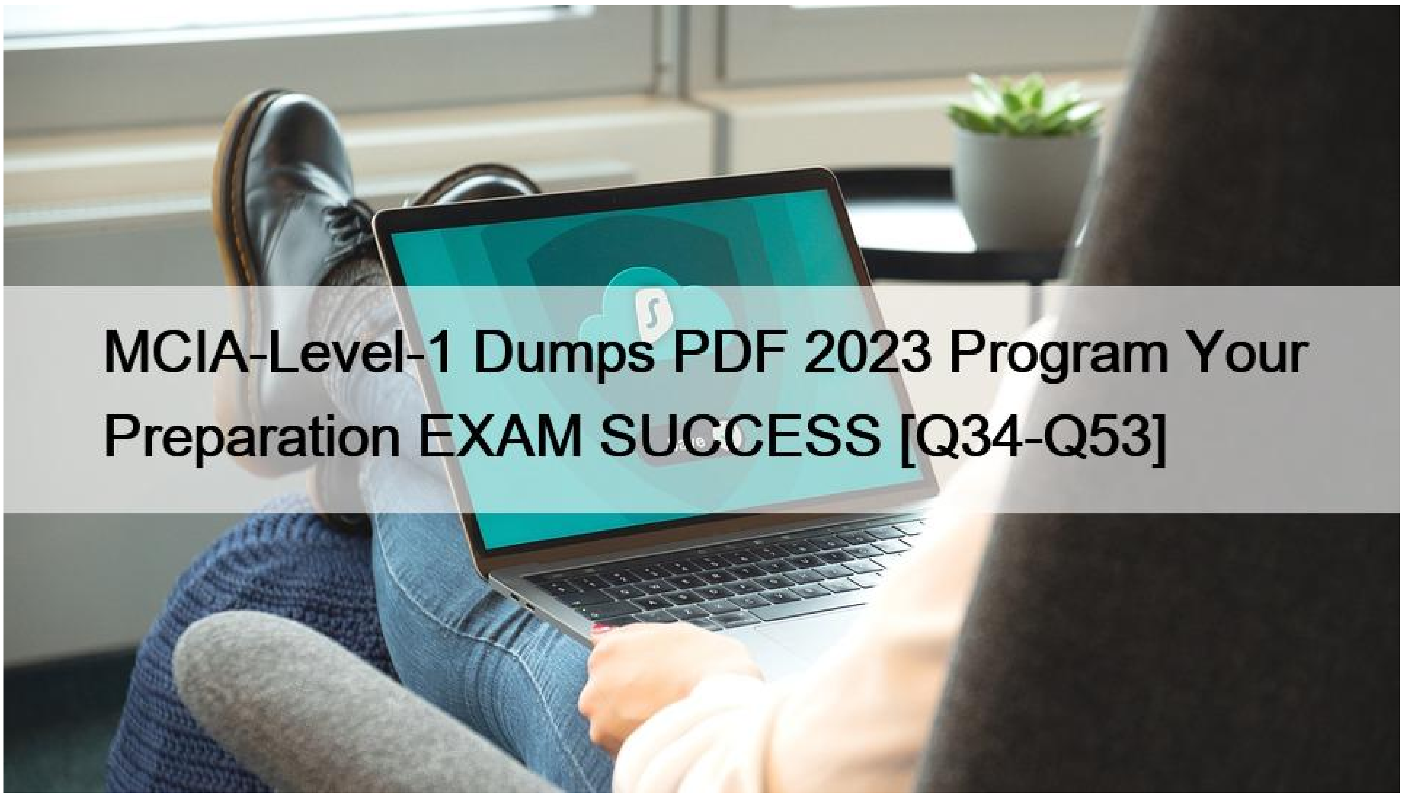
MCIA-Level-1 Dumps PDF 2023 Program Your Preparation EXAM SUCCESS [Q34-Q53



MCIA-Level-1 Dumps PDF 2023 Program Your Preparation EXAM SUCCESS Get Perfect Results with Premium MCIA-Level-1 Dumps Updated 246 Questions Q34. A mule application uses an HTTP request operation to involve an external API.

The external API follows the HTTP specification for proper status code usage.

What is possible cause when a 3xx status code is returned to the HTTP Request operation from the external API?
* The request was not accepted by the external API
* The request was Redirected to a different URL by the external API
* The request was NOT RECEIVED by the external API
* The request was ACCEPTED by the external API

Q35. An organization is sizing an Anypoint VPC to extend their internal network to Cloudhub.

For this sizing calculation, the organization assumes 150 Mule applications will be deployed among three(3) production environments and will use Cloudhub&#8217;s default zero-downtime feature. Each Mule application is expected to be configured with two(2) Cloudhub workers.This is expected to result in several Mule application deployments per hour.
* 10.0.0.0/21(2048 IPs)
* 10.0.0.0/22(1024IPs)
* 10.0.0.0/23(512 IPs)
* 10.0.0.0/24(256 IPs)

* When you create an Anypoint VPC, the range of IP addresses for the network must be specified in the form of a Classless Inter-Domain Routing (CIDR) block, using CIDR notation.

* This address space is reserved for Mule workers, so it cannot overlap with any address space used in your data center if you want to peer it with your VPC.

* To calculate the proper sizing for your Anypoint VPC, you first need to understand that the number of dedicated IP addresses is not the same as the number of workers you have deployed.

* For each worker deployed to CloudHub, the following IP assignation takes place: For better fault tolerance, the VPC subnet may be divided into up to four Availability Zones.

* A few IP addresses are reserved for infrastructure. At least two IP addresses per worker to perform at zero-downtime.

* Hence in this scenario 2048 IP&#8217;s are required to support the requirement.

Q36. An organization is designing the following two Mule applications that must share data via a common persistent object store instance:

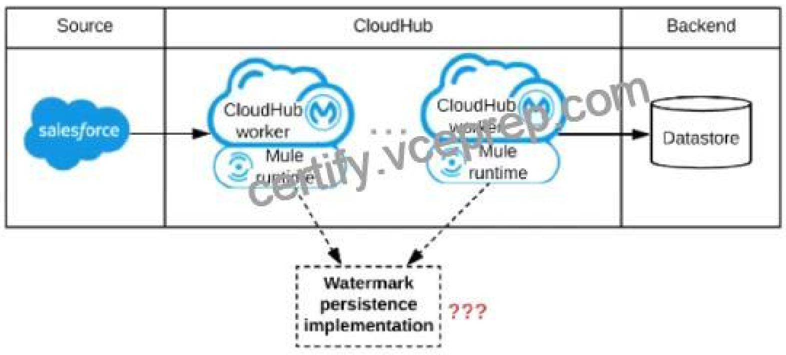
&#8211; Mule application P will be deployed within their on-premises datacenter.

&#8211; Mule application C will run on CloudHub in an Anypoint VPC.

The object store implementation used by CloudHub is the Anypoint Object Store v2 (OSv2).

what type of object store(s) should be used, and what design gives both Mule applications access to the same object store instance?
* Application P uses the Object Store connector to access a persistent object store Application C accesses this persistent object store via the Object Store REST API through an IPsec tunnel
* Application C and P both use the Object Store connector to access the Anypoint Object Store v2
* Application C uses the Object Store connector to access a persistent object Application P accesses the persistent object store via the Object Store REST API
* Application C and P both use the Object Store connector to access a persistent object store

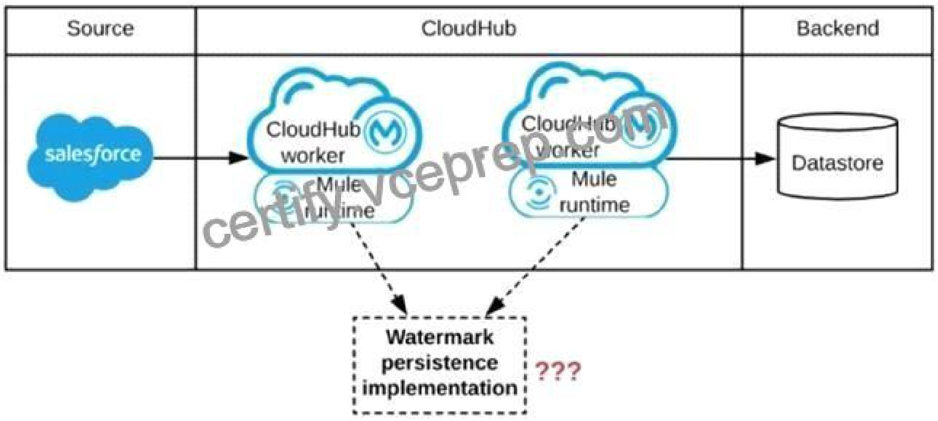
Q37. Refer to the exhibit.



A Mule application is being designed to be deployed to several CIoudHub workers. The Mule application&#8217;s integration logic is to replicate changed Accounts from Satesforce to a backend system every 5 minutes.

A watermark will be used to only retrieve those Satesforce Accounts that have been modified since the last time the integration logic ran.

What is the most appropriate way to implement persistence for the watermark in order to support the required data replication integration logic?



* Persistent Anypoint MQ Queue
* Persistent Object Store
* Persistent Cache Scope
* Persistent VM Queue
* An object store is a facility for storing objects in or across Mule applications. Mule uses object stores to persist data for eventual retrieval.

* Mule provides two types of object stores:

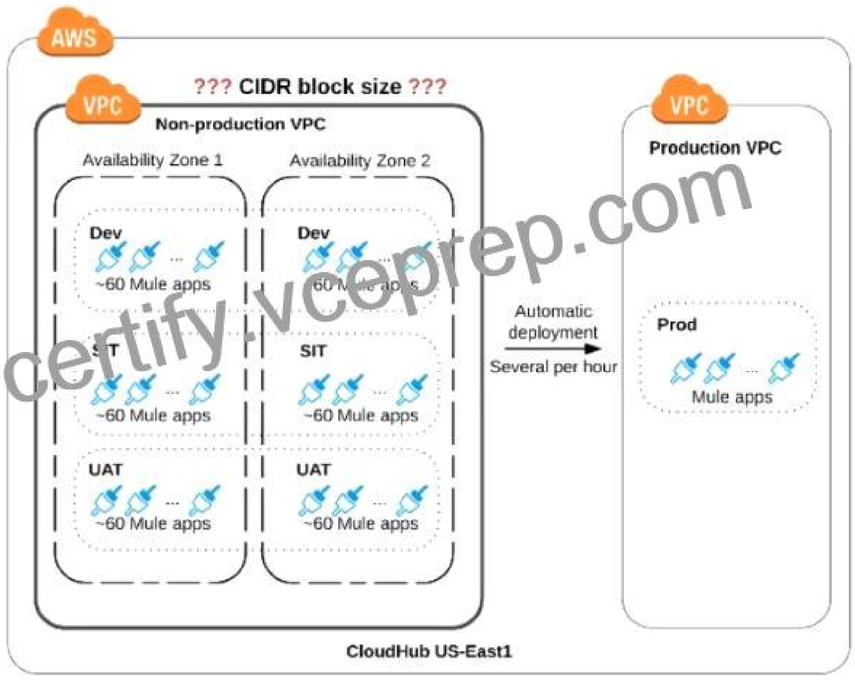
1) In-memory store &#8211; stores objects in local Mule runtime memory. Objects are lost on shutdown of the Mule runtime.

2) Persistent store &#8211; Mule persists data when an object store is explicitly configured to be persistent.

In a standalone Mule runtime, Mule creates a default persistent store in the file system. If you do not specify an object store, the default persistent object store is used.

MuleSoft Reference: https://docs.mulesoft.com/mule-runtime/3.9/mule-object-stores

Q38. 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&#8217;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&#8217;s normal zero-downtime deployment feature.

What is a CIDR block for this VPC that results in the smallest usable private IP address range?
* 10.0.0.0/26 (64 IPS)
* 10.0.0.0/25 (128 IPs)
* 10.0.0.0/24 (256 IPs)
* 10.0.0.0/22 (1024 IPs)
Mule applications are deployed in CloudHub workers and each worker is assigned with a dedicated IP * For zero downtime deployment, each worker in CloudHub needs additional IP addresses * A few IPs in a VPC are reserved for infrastructure (generally 2 IPs) * The IP addresses are usually in a private range with a subnet block specifier, such as 10.0.0.1/24 * The smallest CIDR network subnet block you can assign for your VPC is /24 (256 IP addresses) (60*3 env * 2 worker per application) + 50% of (total) for zero downtime = 540 In this case correct answer is 10.0.0.0/22 as this provided 1024 IP&#8217;s .

Other IP&#8217;s are insufficient.

Q39. What aspects of a CI/CD pipeline for Mule applications can be automated using MuleSoft-provided Maven plugins?
* Compile, package, unit test, validate unit test coverage, deploy
* Compile, package, unit test, deploy, integration test (Incorrect)
* Compile, package, unit test, deploy, create associated API instances in API Manager
* Import from API designer, compile, package, unit test, deploy, publish to Anypoint Exchange
Correct answer is &#8220;Compile, package, unit test, validate unit test coverage, deploy&#8221; : Anypoint Platform supports continuous integration and continuous delivery using industry standard tools Mule Maven Plugin The Mule Maven plugin can

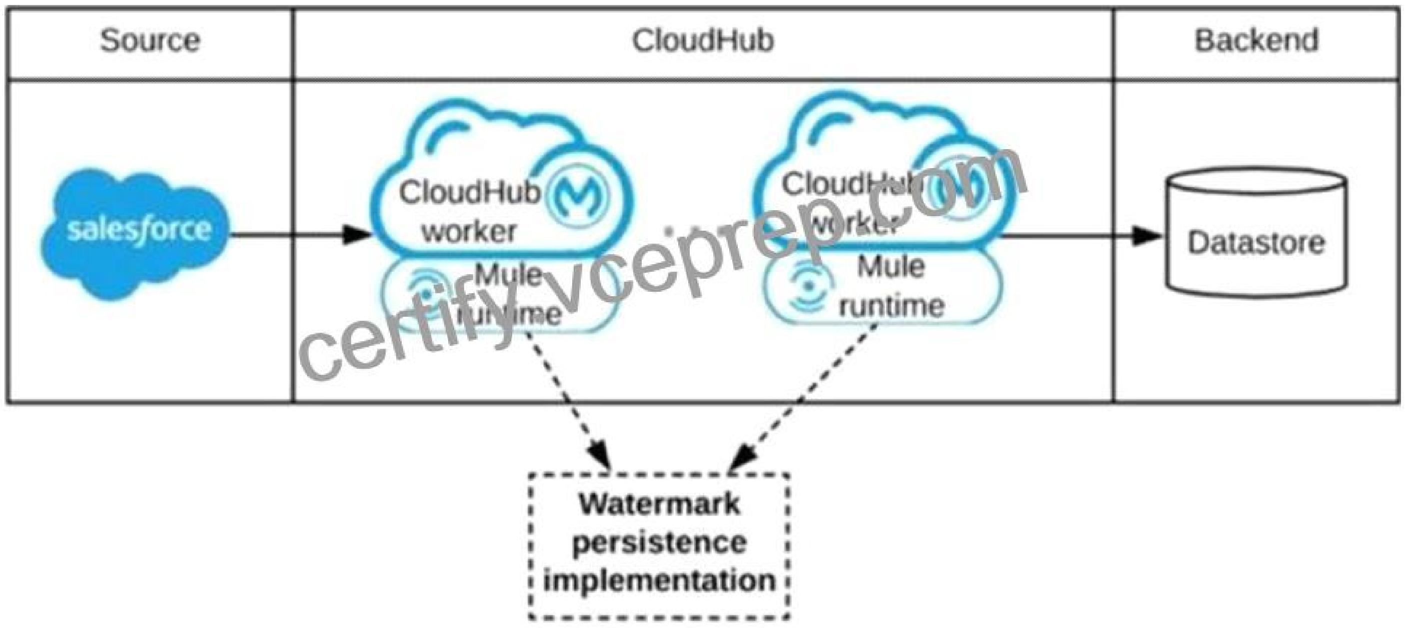
automate building, packaging and deployment of Mule applications from source projects Using the Mule Maven plugin, you can automate your Mule application deployment to CloudHub, to Anypoint Runtime Fabric, or on-premises, using any of the following deployment strategies * CloudHub deployment * Runtime Fabric deployment * Runtime Manager REST API deployment * Runtime Manager agent deployment MUnit Maven Plugin The MUnit Maven plugin can automate test execution, and ties in with the Mule Maven plugin. It provides a full suite of integration and unit test capabilities, and is fully integrated with Maven and Surefire for integration with your continuous deployment environment. Since MUnit 2.x, the coverage report goal is integrated with the maven reporting section. Coverage Reports are generated during Maven&#8217;s site lifecycle, during the coverage-report goal. One of the features of MUnit Coverage is to fail the build if a certain coverage level is not reached. MUnit is not used for integration testing Also publishing to Anypoint Exchange or to create associated API instances in API Manager is not a part of CICD pipeline which can ne achieved using mulesoft provided maven plugin Architecture mentioned in the question can be diagrammatically put as below. Persistent Object Store is the correct answer .

* Mule Object Stores: An object store is a facility for storing objects in or across Mule applications. Mule uses object stores to persist data for eventual retrieval.

Mule provides two types of object stores:

1) In-memory store &#8211; stores objects in local Mule runtime memory. Objects are lost on shutdown of the Mule runtime. So we cant use in memory store in our scenario as we want to share watermark within all cloudhub workers

2) Persistent store &#8211; Mule persists data when an object store is explicitly configured to be persistent. Hence this watermark will be available even any of the worker goes down



Q40. An organization is migrating all its Mule applications to Runtime Fabric (RTF). None of the Mule applications use Mule domain projects.

Currently, all the Mule applications have been manually deployed to a server group among several customer- hosted Mule runtimes. Port conflicts between these Mule application deployments are currently managed by the DevOps team who carefully manage Mule application properties files.

When the Mule applications are migrated from the current customer-hosted server group to Runtime Fabric (RTF), do the Mule applications need to be rewritten, and what DevOps port configuration responsibilities change or stay the same?
* NO, the Mule applications do NOT need to be rewritten

DevOps MUST STILL manage port conflicts
* NO, the Mule applications do NOT need to be rewritten

DevOps NO LONGER needs to manage port conflicts between the Mule applications
* YES, the Mule applications MUST be rewritten

DevOps NO LONGER needs to manage port conflicts between the Mule applications
* YES, the Mule applications MUST be rewritten

DevOps MUST STILL manage port conflicts

Q41. A Mule application currently writes to two separate SQL Server database instances across the internet using a single XA transaction. It is 58. proposed to split this one transaction into two separate non-XA transactions with no other changes to the Mule application.

What non-functional requirement can be expected to be negatively affected when implementing this change?
* Throughput
* Consistency
* Response time
* Availability
Correct answer is Consistency as XA transactions are implemented to achieve this. XA transactions are added in the implementation to achieve goal of ACID properties. In the context of transaction processing, the acronym ACID refers to the four key properties of a transaction: atomicity, consistency, isolation, and durability. Atomicity : All changes to data are performed as if they are a single operation. That is, all the changes are performed, or none of them are. For example, in an application that transfers funds from one account to another, the atomicity property ensures that, if a debit is made successfully from one account, the corresponding credit is made to the other account. Consistency : Data is in a consistent state when a transaction starts and when it ends.For example, in an application that transfers funds from one account to another, the consistency property ensures that the total value of funds in both the accounts is the same at the start and end of each transaction. Isolation : The intermediate state of a transaction is invisible to other transactions. As a result, transactions that run concurrently appear to be serialized. For example, in an application that transfers funds from one account to another, the isolation property ensures that another transaction sees the transferred funds in one account or the other, but not in both, nor in neither. Durability : After a transaction successfully completes, changes to data persist and are not undone, even in the event of a system failure. For example, in an application that transfers funds from one account to another, the durability property ensures that the changes made to each account will not be reversed. MuleSoft reference: https://docs.mulesoft.com/mule-runtime/4.3/xa-transactions

Q42. An external REST client periodically sends an array of records in a single POST request to a Mule application API endpoint.

The Mule application must validate each record of the request against a JSON schema before sending it to a downstream system in the same order that it was received in the array Record processing will take place inside a router or scope that calls a child flow. The child flow has its own error handling defined. Any validation or communication failures should not prevent further processing of the remaining records.

To best address these requirements what is the most idiomatic(used for it intended purpose) router or scope to used in the parent flow, and what type of error handler should be used in the child flow?
* First Successful router in the parent flow

On Error Continue error handler in the child flow
* For Each scope in the parent flow

On Error Continue error handler in the child flow
* Parallel For Each scope in the parent flow

On Error Propagate error handler in the child flow
* Until Successful router in the parent flow

On Error Propagate error handler in the child flow

Q43. Mule application A receives a request Anypoint MQ message REQU with a payload containing a variable- length list of request objects. Application A uses the For Each scope to split the list into individual objects and sends each object as a message to an Anypoint MQ queue.

Service S listens on that queue, processes each message independently of all other messages, and sends a response message to a response queue.

Application A listens on that response queue and must in turn create and publish a response Anypoint MQ message RESP with a payload containing the list of responses sent by service S in the same order as the request objects originally sent in REQU.

Assume successful response messages are returned by service S for all request messages.

What is required so that application A can ensure that the length and order of the list of objects in RESP and REQU match, while at the same time maximizing message throughput?
* Perform all communication involving service S synchronously from within the For Each scope, so objects in RESP are in the exact same order as request objects in REQU
* Use a Scatter-Gather within the For Each scope to ensure response message order Configure the Scatter-Gather with a persistent object store
* Keep track of the list length and all object indices in REQU, both in the For Each scope and in all communication involving service. Use persistent storage when creating RESP
* Use an Async scope within the For Each scope and collect response messages in a second For Each scope in the order in which they arrive, then send RESP using this list of responses

Q44. An organization&#8217;s governance process requires project teams to get formal approval from all key stakeholders for all new Integration design specifications. An integration Mule application Is being designed that interacts with various backend systems. The Mule application will be created using Anypoint Design Center or Anypoint Studio and will then be deployed to a customer-hosted runtime.

What key elements should be included in the integration design specification when requesting approval for this Mule application?
* SLAs and non-functional requirements to access the backend systems
* Snapshots of the Mule application&#8217;s flows, including their error handling
* A list of current and future consumers of the Mule application and their contact details
* The credentials to access the backend systems and contact details for the administrator of each system
SLAs and non-functional requirements to access the backend systems. Only this option actually speaks to design parameters and reqs. * Below two are technical implementations and not the part of design: &#8211; Snapshots of the Mule application&#8217;s flows, including their error handling &#8211; The credentials to access the backend systems and contact details for the administrator of each system * List of consumers is not relevant to the design

Q45. An organization is migrating all its Mule applications to Runtime Fabric (RTF). None of the Mule applications use Mule domain projects.

Currently, all the Mule applications have been manually deployed to a server group among several customer hosted Mule runtimes.

Port conflicts between these Mule application deployments are currently managed by the DevOps team who carefully manage Mule application properties files.

When the Mule applications are migrated from the current customer-hosted server group to Runtime Fabric (RTF), fo the Mule applications need to be rewritten and what DevOps port configuration responsibilities change or stay the same?
* Yes, the Mule applications Must be rewritten

DevOps No Longer needs to manage port conflicts between the Mule applications
* Yes, the Mule applications Must be rewritten

DevOps Must Still Manage port conflicts.
* NO, The Mule applications do NOT need to be rewritten

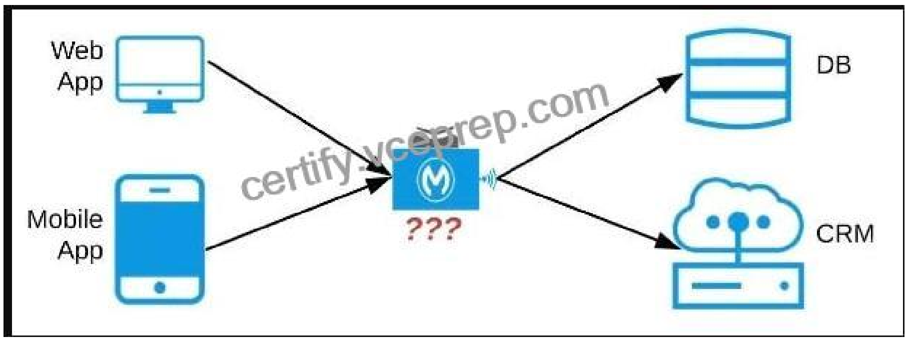
DevOps MUST STILL manage port conflicts
* NO, the Mule applications do NO need to be rewritten

DevOps NO LONGER needs to manage port conflicts between the Mule applications.

Q46. 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?
* Create a Mule plugin project with all the dependencies and add it as a dependency in each application&#8217;s POM.xml file
* Create a mule domain project with all the dependencies define in its POM.xml file and add each application to the domain Project
* Add all dependencies in each application&#8217;s POM.xml file
* Create a parent POM of all the required dependencies and reference each in each application&#8217;s POM.xml file

Q47. An organization needs to enable access to their customer data from both a mobile app and a web application, which each need access to common fields as well as certain unique fields. The data is available partially in a database and partially in a 3rd-party CRM system. What APIs should be created to best fit these design requirements?



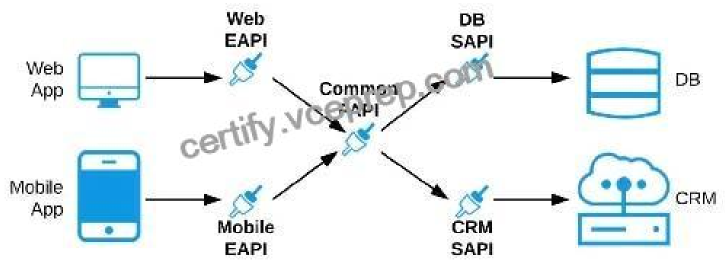
* A Process API that contains the data required by both the web and mobile apps, allowing these applications to invoke it directly and access the data they need thereby providing the flexibility to add more fields in the future without needing API changes.

* One set of APIs (Experience API, Process API, and System API) for the web app, and another set for the mobile app.
* Separate Experience APIs for the mobile and web app, but a common Process API that invokes separate System APIs created for the database and CRM system
* A common Experience API used by both the web and mobile apps, but separate Process APIs for the web and mobile apps that interact with the database and the CRM System.

Lets analyze the situation in regards to the different options available Option : A common Experience API but separate Process APIs Analysis : This solution will not work because having common experience layer will not help the purpose as mobile and web applications will have different set of requirements which cannot be fulfilled by single experience layer API Option : Common Process API Analysis : This solution will not work because creating a common process API will impose limitations in terms of flexibility to customize API;s as per the requirements of different applications. It is not a recommended approach.

Option : Separate set of API&#8217;s for both the applications Analysis : This goes against the principle of Anypoint API-led connectivity approach which promotes creating reusable assets. This solution may work but this is not efficient solution and creates duplicity of code.

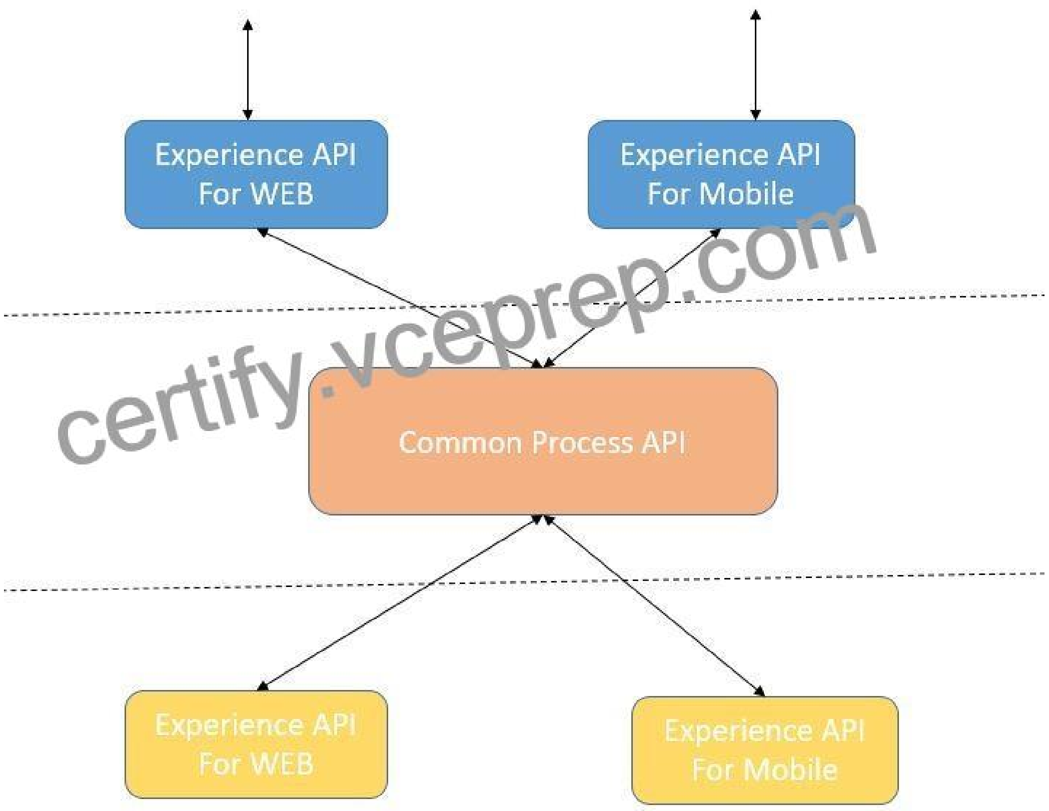
Hence the correct answer is: Separate Experience APIs for the mobile and web app, but a common Process API that invokes separate System APIs created for the database and CRM system



Lets analyze the situation in regards to the different options available Option : A common Experience API but separate Process APIs Analysis : This solution will not work because having common experience layer will not help the purpose as mobile and web applications will have different set of requirements which cannot be fulfilled by single experience layer API Option : Common Process API Analysis : This solution will not work because creating a common process API will impose limitations in terms of flexibility to customize API;s as per the requirements of different applications. It is not a recommended approach.

Option : Separate set of API&#8217;s for both the applications Analysis : This goes against the principle of Anypoint API-led connectivity approach which promotes creating reusable assets. This solution may work but this is not efficient solution and creates duplicity of code.

Hence the correct answer is: Separate Experience APIs for the mobile and web app, but a common Process API that invokes separate System APIs created for the database and CRM system



Q48. What requires configuration of both a key store and a trust store for an HTTP Listener?
* Support for TLS mutual (two-way) authentication with HTTP clients
* Encryption of requests to both subdomains and API resource endpoints fhttPs://aDi.customer.com/ and

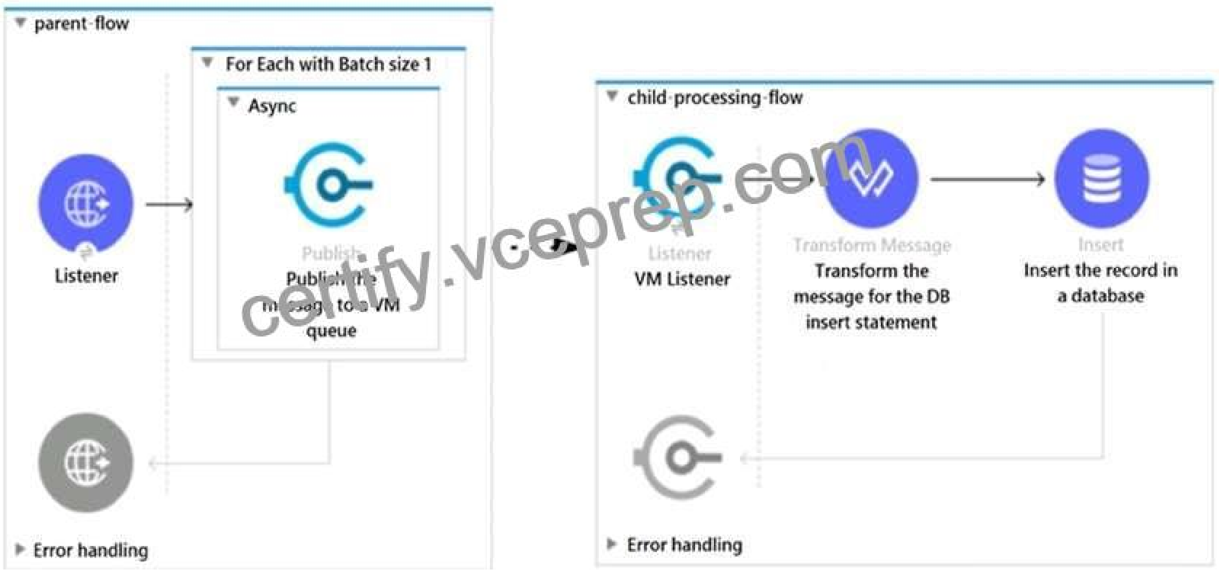
https://customer.com/api)
* Encryption of both HTTP request and HTTP response bodies for all HTTP clients
* Encryption of both HTTP request header and HTTP request body for all HTTP clients

Q49. Refer to the exhibit. A Mule 4 application has a parent flow that breaks up a JSON array payload into 200 separate items, then sends each item one at a time inside an Async scope to a VM queue.

A second flow to process orders has a VM Listener on the same VM queue. The rest of this flow processes each received item by writing the item to a database.

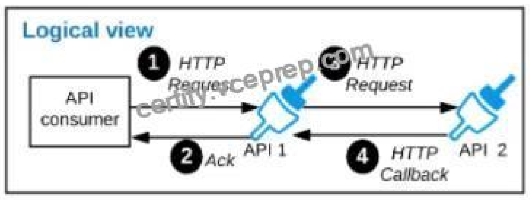
This Mule application is deployed to four CloudHub workers with persistent queues enabled.

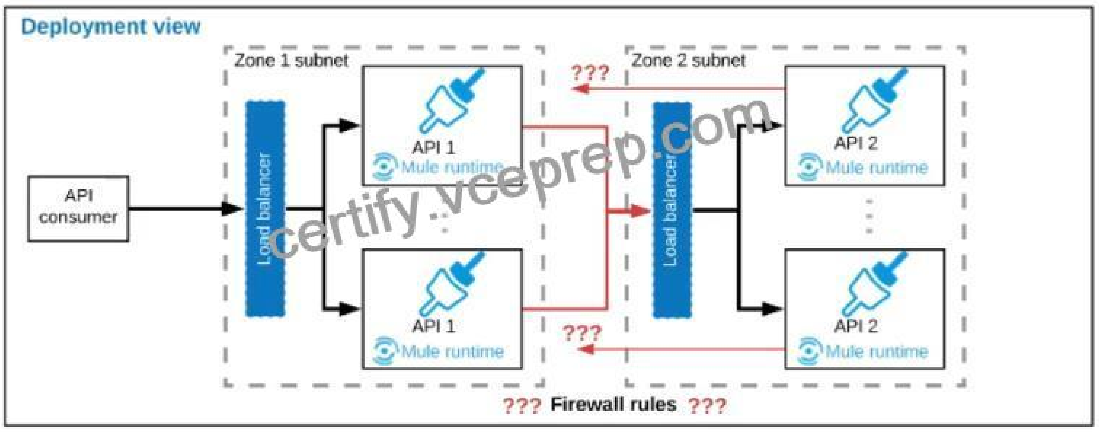
What message processing guarantees are provided by the VM queue and the CloudHub workers, and how are VM messages routed among the CloudHub workers for each invocation of the parent flow under normal operating conditions where all the CloudHub workers remain online?


parent-flow
For Each with Batch size 1
Async
Listener
Publish
Publish the message to a VM queue
Error handling
child-processing-flow
Listener
VM Listener
Transform Message
Transform the message for the DB insert statement
Insert
Insert the record in a database
Error handling

* EACH item VM message is processed AT LEAST ONCE by ONE ARBITRARY CloudHub worker Each of the four CloudHub workers can be expected to process some item VM messages
* ALL item VM messages are processed AT MOST ONCE by ONE ARBITRARY CloudHub worker This one CloudHub worker processes ALL 200 item VM messages
* ALL item VM messages are processed AT LEAST ONCE by the SAME CloudHub worker where the parent flow was invoked This one CloudHub worker processes ALL 200 item VM messages
* EACH item VM message is processed AT MOST ONCE by ONE CloudHub worker, with workers chosen in a deterministic round-robin fashion Each of the four CloudHub workers can be expected to process 1/4 of the item VM messages (about 50 items)

Q50. Refer to the exhibit.



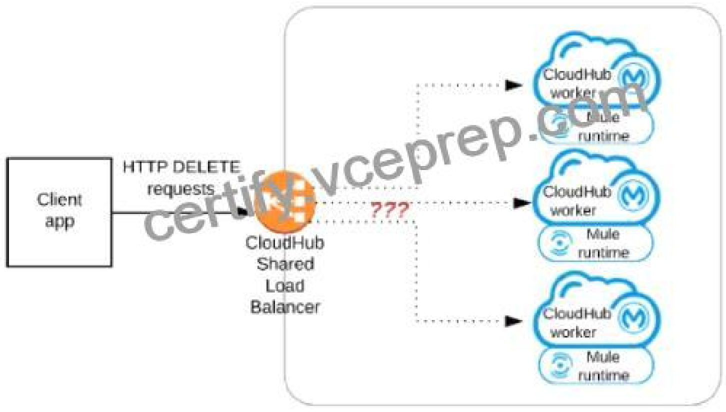


A business process involves two APIs that interact with each other asynchronously over HTTP. Each API is implemented as a Mule application. API 1 receives the initial HTTP request and invokes API 2 (in a fire and forget fashion) while API 2, upon completion of the processing, calls back into API l to notify about completion of the asynchronous process.

Each API Is deployed to multiple redundant Mule runtimes and a separate load balancer, and is deployed to a separate network zone.

In the network architecture, how must the firewall rules be configured to enable the above Interaction between API 1 and API 2?
* To allow communication between the load balancers used by each API
* To authorize the certificates used by both the apis
* To enable communication from each API&#8217;s Mule runtimes and network zone to the toad balancer of the other API
* To open direct two-way communication between the Mule runtimes of both APIs

Q51. Refer to the exhibit.



A Mule application has an HTTP Listener that accepts HTTP DELETE requests. This Mule application Is deployed to three CloudHub workers under the control of the CloudHub Shared Load Balancer.

A web client makes a sequence of requests to the Mule application&#8217;s public URL.

How is this sequence of web client requests distributed among the HTTP Listeners running in the three CloudHub workers?
* Each request is routed to the PRIMARY CloudHub worker in the PRIMARY Availability Zone (AZ)
* Each request is routed to ONE ARBiTRARY CloudHub worker in the PRIMARY Availability Zone (AZ)
* Each request Is routed to ONE ARBiTRARY CloudHub worker out of ALL three CloudHub workers
* Each request is routed (scattered) to ALL three CloudHub workers at the same time

Q52. An ABC Farms project team is planning to build a new API that is required to work with data from different domains across the organization.

The organization has a policy that all project teams should leverage existing investments by reusing existing APIs and related resources and documentation that other project teams have already developed and deployed.

To support reuse, where on Anypoint Platform should the project team go to discover and read existing APIs, discover related resources and documentation, and interact with mocked versions of those APIs?

* Design Center
* API Manager
* Runtime Manager
* Anypoint Exchange
:

The mocking service is a feature of Anypoint Platform and runs continuously. You can run the mocking service from the text editor, the visual editor, and from Anypoint Exchange. You can simulate calls to the API in API Designer before publishing the API specification to Exchange or in Exchange after publishing the API specification.

Q53. One of the backend systems involved by the API implementation enforces rate limits on the number of request a particle client can make.

Both the back-end system and API implementation are deployed to several non-production environments including the staging environment and to a particular production environment. Rate limiting of the back-end system applies to all non-production environments.

The production environment however does not have any rate limiting.

What is the cost-effective approach to conduct performance test of the API implementation in the non-production staging environment?
* Including logic within the API implementation that bypasses in locations of the back-end system in the staging environment and invoke a Mocking service that replicates typical back-end system responses Then conduct performance test using this API implementation
* Use MUnit to simulate standard responses from the back-end system.

Then conduct performance test to identify other bottlenecks in the system
* Create a Mocking service that replicates the back-end system&#8217;s

production performance characteristics

Then configure the API implementation to use the mocking

service and conduct the performance test
* Conduct scaled-down performance tests in the staging environment against rate-limiting back-end system. Then upscale performance results to full production scale