

# MuleSoft

## Exam Questions MCPA-Level-1

MuleSoft Certified Platform Architect - Level 1



#### NEW QUESTION 1

True or False. We should always make sure that the APIs being designed and developed are self-servable even if it needs more man-day effort and resources.

- A. FALSE
- B. TRUE

**Answer: B**

#### Explanation:

Correct Answer  
TRUE

\*\*\*\*\*

>> As per MuleSoft proposed IT Operating Model, designing APIs and making sure that they are discoverable and self-servable is VERY VERY IMPORTANT and decides the success of an API and its application network.

#### NEW QUESTION 2

What best describes the Fully Qualified Domain Names (FQDNs), also known as DNS entries, created when a Mule application is deployed to the CloudHub Shared Worker Cloud?

- A. A fixed number of FQDNs are created, IRRESPECTIVE of the environment and VPC design
- B. The FQDNs are determined by the application name chosen, IRRESPECTIVE of the region
- C. The FQDNs are determined by the application name, but can be modified by an administrator after deployment
- D. The FQDNs are determined by both the application name and the Anypoint Platform organization

**Answer: B**

#### Explanation:

Correct Answer  
The FQDNs are determined by the application name chosen, IRRESPECTIVE of the region

\*\*\*\*\*

>> When deploying applications to Shared Worker Cloud, the FQDN are always determined by application name chosen.

>> It does NOT matter what region the app is being deployed to.

>> Although it is fact and true that the generated FQDN will have the region included in it (Ex:

exp-salesorder-api.au-s1.cloudhub.io), it does NOT mean that the same name can be used when deploying to another CloudHub region.

>> Application name should be universally unique irrespective of Region and Organization and solely determines the FQDN for Shared Load Balancers.

#### NEW QUESTION 3

A retail company is using an Order API to accept new orders. The Order API uses a JMS queue to submit orders to a backend order management service. The normal load for orders is being handled using two (2) CloudHub workers, each configured with 0.2 vCore. The CPU load of each CloudHub worker normally runs well below 70%. However, several times during the year the Order API gets four times (4x) the average number of orders. This causes the CloudHub worker CPU load to exceed 90% and the order submission time to exceed 30 seconds. The cause, however, is NOT the backend order management service, which still responds fast enough to meet the response SLA for the Order API. What is the MOST resource-efficient way to configure the Mule application's CloudHub deployment to help the company cope with this performance challenge?

- A. Permanently increase the size of each of the two (2) CloudHub workers by at least four times (4x) to one(1) vCore
- B. Use a vertical CloudHub autoscaling policy that triggers on CPU utilization greater than 70%
- C. Permanently increase the number of CloudHub workers by four times (4x) to eight (8) CloudHub workers
- D. Use a horizontal CloudHub autoscaling policy that triggers on CPU utilization greater than 70%

**Answer: D**

#### Explanation:

Correct Answer  
Use a horizontal CloudHub autoscaling policy that triggers on CPU utilization greater than 70%

\*\*\*\*\*

The scenario in the question is very clearly stating that the usual traffic in the year is pretty well handled by the existing worker configuration with CPU running well below 70%. The problem occurs only "sometimes" occasionally when there is spike in the number of orders coming in.

So, based on above, We neither need to permanently increase the size of each worker nor need to permanently increase the number of workers. This is unnecessary as other than those "occasional" times the resources are idle and wasted.

We have two options left now. Either to use horizontal Cloudhub autoscaling policy to automatically increase the number of workers or to use vertical Cloudhub autoscaling policy to automatically increase the vCore size of each worker.

Here, we need to take two things into consideration:

\* 1. CPU

\* 2. Order Submission Rate to JMS Queue

>> From CPU perspective, both the options (horizontal and vertical scaling) solves the issue. Both helps to bring down the usage below 90%.

>> However, If we go with Vertical Scaling, then from Order Submission Rate perspective, as the application is still being load balanced with two workers only, there may not be much improvement in the incoming request processing rate and order submission rate to JMS queue. The throughput would be same as before. Only CPU utilization comes down.

>> But, if we go with Horizontal Scaling, it will spawn new workers and adds extra hand to increase the throughput as more workers are being load balanced now. This way we can address both CPU and Order Submission rate.

Hence, Horizontal CloudHub Autoscaling policy is the right and best answer.

#### NEW QUESTION 4

What best explains the use of auto-discovery in API implementations?

- A. It makes API Manager aware of API implementations and hence enables it to enforce policies
- B. It enables Anypoint Studio to discover API definitions configured in Anypoint Platform
- C. It enables Anypoint Exchange to discover assets and makes them available for reuse
- D. It enables Anypoint Analytics to gain insight into the usage of APIs

**Answer:** A

**Explanation:**

Correct Answer

It makes API Manager aware of API implementations and hence enables it to enforce policies.

\*\*\*\*\*

>> API Autodiscovery is a mechanism that manages an API from API Manager by pairing the deployed application to an API created on the platform.

>> API Management includes tracking, enforcing policies if you apply any, and reporting API analytics.

>> Critical to the Autodiscovery process is identifying the API by providing the API name and version. References:

<https://docs.mulesoft.com/api-manager/2.x/api-auto-discovery-new-concept> <https://docs.mulesoft.com/api-manager/1.x/api-auto-discovery>

<https://docs.mulesoft.com/api-manager/2.x/api-auto-discovery-new-concept>

#### NEW QUESTION 5

What Mule application can have API policies applied by

Anypoint Platform to the endpoint exposed by that Mule application?

A) A Mule application that accepts requests over HTTP/1.x

B) A Mule application that accepts JSON requests over TCP but is NOT required to provide a response

C) A Mule application that accepts JSON requests over WebSocket

D) A Mule application that accepts gRPC requests over HTTP/2

A. Option A

B. Option B

C. Option C

D. Option D

**Answer:** A

**Explanation:**

Correct Answer

Option A

\*\*\*\*\*

>> Anypoint API Manager and API policies are applicable to all types of HTTP/1.x APIs.

>> They are not applicable to WebSocket APIs, HTTP/2 APIs and gRPC APIs

#### NEW QUESTION 6

What is the most performant out-of-the-box solution in Anypoint Platform to track transaction state in an asynchronously executing long-running process implemented as a Mule application deployed to multiple CloudHub workers?

A. Redis distributed cache

B. java.util.WeakHashMap

C. Persistent Object Store

D. File-based storage

**Answer:** C

**Explanation:**

Correct Answer

Persistent Object Store

\*\*\*\*\*

>> Redis distributed cache is performant but NOT out-of-the-box solution in Anypoint Platform

>> File-storage is neither performant nor out-of-the-box solution in Anypoint Platform

>> java.util.WeakHashMap needs a completely custom implementation of cache from scratch using Java code and is limited to the JVM where it is running. Which means the state in the cache is not worker aware when running on multiple workers. This type of cache is local to the worker. So, this is neither out-of-the-box nor

worker-aware among multiple workers on cloudhub. <https://www.baeldung.com/java-weakhashmap>  
>> Persistent Object Store is an out-of-the-box solution provided by Anypoint Platform which is performant as well as worker aware among multiple workers running on CloudHub. <https://docs.mulesoft.com/object-store/>  
So, Persistent Object Store is the right answer.

#### NEW QUESTION 7

What is most likely NOT a characteristic of an integration test for a REST API implementation?

- A. The test needs all source and/or target systems configured and accessible
- B. The test runs immediately after the Mule application has been compiled and packaged
- C. The test is triggered by an external HTTP request
- D. The test prepares a known request payload and validates the response payload

**Answer: B**

#### Explanation:

Correct Answer

The test runs immediately after the Mule application has been compiled and packaged

\*\*\*\*\*

>> Integration tests are the last layer of tests we need to add to be fully covered.  
>> These tests actually run against Mule running with your full configuration in place and are tested from external source as they work in PROD.  
>> These tests exercise the application as a whole with actual transports enabled. So, external systems are affected when these tests run.  
So, these tests do NOT run immediately after the Mule application has been compiled and packaged.  
FYI... Unit Tests are the one that run immediately after the Mule application has been compiled and packaged.

#### NEW QUESTION 8

What is a best practice when building System APIs?

- A. Document the API using an easily consumable asset like a RAML definition
- B. Model all API resources and methods to closely mimic the operations of the backend system
- C. Build an Enterprise Data Model (Canonical Data Model) for each backend system and apply it to System APIs
- D. Expose to API clients all technical details of the API implementation's interaction with the backend system

**Answer: B**

#### Explanation:

Correct Answer

Model all API resources and methods to closely mimic the operations of the backend system.

\*\*\*\*\*

>> There are NO fixed and straight best practices while opting data models for APIs. They are completely contextual and depends on number of factors. Based upon those factors, an enterprise can choose if they have to go with Enterprise Canonical Data Model or Bounded Context Model etc.  
>> One should NEVER expose the technical details of API implementation to their API clients. Only the API interface/ RAML is exposed to API clients.  
>> It is true that the RAML definitions of APIs should be as detailed as possible and should reflect most of the documentation. However, just that is NOT enough to call your API as best documented API. There should be even more documentation on Anypoint Exchange with API Notebooks etc. to make and create a developer friendly API and repository..  
>> The best practice always when creating System APIs is to create their API interfaces by modeling their resources and methods to closely reflect the operations and functionalities of that backend system.

#### NEW QUESTION 9

An API implementation is deployed on a single worker on CloudHub and invoked by external API clients (outside of CloudHub). How can an alert be set up that is guaranteed to trigger AS SOON AS that API implementation stops responding to API invocations?

- A. Implement a heartbeat/health check within the API and invoke it from outside the Anypoint Platform and alert when the heartbeat does not respond
- B. Configure a "worker not responding" alert in Anypoint Runtime Manager
- C. Handle API invocation exceptions within the calling API client and raise an alert from that API client when the API is unavailable
- D. Create an alert for when the API receives no requests within a specified time period

**Answer: B**

#### Explanation:

Correct Answer

Configure a "Worker not responding" alert in Anypoint Runtime Manager.

\*\*\*\*\*

>> All the options eventually helps to generate the alert required when the application stops responding.  
>> However, handling exceptions within calling API and then raising alert from API client is inappropriate and silly. There could be many API clients invoking the API implementation and it is not ideal to have this setup consistently in all of them. Not a realistic way to do.  
>> Implementing a health check/ heartbeat with in the API and calling from outside to determine the health sounds OK but needs extra setup for it and same time there are very good chances of generating false alarms when there are any intermittent network issues between external tool calling the health check API on API implementation. The API implementation itself may not have any issues but due to some other factors some false alarms may go out.  
>> Creating an alert in API Manager when the API receives no requests within a specified time period would actually generate realistic alerts but even here some false alarms may go out when there are genuinely no requests from API clients.  
The best and right way to achieve this requirement is to setup an alert on Runtime Manager with a condition "Worker not responding". This would generate an alert AS SOON AS the workers become unresponsive.

Bottom of Form Top of Form

#### NEW QUESTION 10

When designing an upstream API and its implementation, the development team has been advised to NOT set timeouts when invoking a downstream API, because that downstream API has no SLA that can be relied upon. This is the only downstream API dependency of that upstream API. Assume the downstream API runs uninterrupted without crashing. What is the impact of this advice?

- A. An SLA for the upstream API CANNOT be provided
- B. The invocation of the downstream API will run to completion without timing out
- C. A default timeout of 500 ms will automatically be applied by the Mule runtime in which the upstream API implementation executes
- D. A load-dependent timeout of less than 1000 ms will be applied by the Mule runtime in which the downstream API implementation executes

**Answer: A**

#### Explanation:

Correct Answer

An SLA for the upstream API CANNOT be provided.

\*\*\*\*\*

>> First thing first, the default HTTP response timeout for HTTP connector is 10000 ms (10 seconds). NOT 500 ms.  
>> Mule runtime does NOT apply any such "load-dependent" timeouts. There is no such behavior currently in Mule.  
>> As there is default 10000 ms time out for HTTP connector, we CANNOT always guarantee that the invocation of the downstream API will run to completion without timing out due to its unreliable SLA times. If the response time crosses 10 seconds then the request may time out.  
The main impact due to this is that a proper SLA for the upstream API CANNOT be provided.

#### NEW QUESTION 10

What do the API invocation metrics provided by Anypoint Platform provide?

- A. ROI metrics from APIs that can be directly shared with business users
- B. Measurements of the effectiveness of the application network based on the level of reuse
- C. Data on past API invocations to help identify anomalies and usage patterns across various APIs
- D. Proactive identification of likely future policy violations that exceed a given threat threshold

**Answer: C**

#### Explanation:

Correct Answer

Data on past API invocations to help identify anomalies and usage patterns across various APIs

\*\*\*\*\*

API Invocation metrics provided by Anypoint Platform:

>> Does NOT provide any Return Of Investment (ROI) related information. So the option suggesting it is OUT.  
>> Does NOT provide any information w.r.t how APIs are reused, whether there is effective usage of APIs or not etc...  
>> Does NOT provide any prediction information as such to help us proactively identify any future policy violations.  
So, the kind of data/information we can get from such metrics is on past API invocations to help identify anomalies and usage patterns across various APIs.

#### NEW QUESTION 12

What correctly characterizes unit tests of Mule applications?

- A. They test the validity of input and output of source and target systems
- B. They must be run in a unit testing environment with dedicated Mule runtimes for the environment
- C. They must be triggered by an external client tool or event source
- D. They are typically written using MUnit to run in an embedded Mule runtime that does not require external connectivity

**Answer: D**

#### Explanation:

Correct Answer

They are typically written using MUnit to run in an embedded Mule runtime that does not require external connectivity.

\*\*\*\*\*

Below TWO are characteristics of Integration Tests but NOT unit tests:

>> They test the validity of input and output of source and target systems.  
>> They must be triggered by an external client tool or event source.

It is NOT TRUE that Unit Tests must be run in a unit testing environment with dedicated Mule runtimes for the environment.

MuleSoft offers MUnit for writing Unit Tests and they run in an embedded Mule Runtime without needing any separate/ dedicated Runtimes to execute them. They also do NOT need any external connectivity as MUnit supports mocking via stubs.

<https://dzone.com/articles/munit-framework>

#### NEW QUESTION 13

An API has been updated in Anypoint Exchange by its API producer from version 3.1.1 to 3.2.0 following accepted semantic versioning practices and the changes have been communicated via the API's public portal.

The API endpoint does NOT change in the new version.

How should the developer of an API client respond to this change?

- A. The update should be identified as a project risk and full regression testing of the functionality that uses this API should be run
- B. The API producer should be contacted to understand the change to existing functionality
- C. The API producer should be requested to run the old version in parallel with the new one
- D. The API client code ONLY needs to be changed if it needs to take advantage of new features

**Answer: D**



#### NEW QUESTION 15

What CANNOT be effectively enforced using an API policy in Anypoint Platform?

- A. Guarding against Denial of Service attacks
- B. Maintaining tamper-proof credentials between APIs
- C. Logging HTTP requests and responses
- D. Backend system overloading

**Answer:** A

#### Explanation:

Correct Answer

Guarding against Denial of Service attacks

\*\*\*\*\*

>> Backend system overloading can be handled by enforcing "Spike Control Policy"

>> Logging HTTP requests and responses can be done by enforcing "Message Logging Policy"

>> Credentials can be tamper-proofed using "Security" and "Compliance" Policies

However, unfortunately, there is no proper way currently on Anypoint Platform to guard against DOS attacks.

#### NEW QUESTION 16

What API policy would be LEAST LIKELY used when designing an Experience API that is intended to work with a consumer mobile phone or tablet application?

- A. OAuth 2.0 access token enforcement
- B. Client ID enforcement
- C. JSON threat protection
- D. IPwhitelist

**Answer:** D

#### Explanation:

Correct Answer

IP whitelist

\*\*\*\*\*

>> OAuth 2.0 access token and Client ID enforcement policies are VERY common to apply on Experience APIs as API consumers need to register and access the APIs using one of these mechanisms

>> JSON threat protection is also VERY common policy to apply on Experience APIs to prevent bad or suspicious payloads hitting the API implementations.

>> IP whitelisting policy is usually very common in Process and System APIs to only whitelist the IP range inside the local VPC. But also applied occasionally on some experience APIs where the End User/ API Consumers are FIXED.

>> When we know the API consumers upfront who are going to access certain Experience APIs, then we can request for static IPs from such consumers and whitelist them to prevent anyone else hitting the API.

However, the experience API given in the question/ scenario is intended to work with a consumer mobile phone or tablet application. Which means, there is no way we can know all possible IPs that are to be whitelisted as mobile phones and tablets can so many in number and any device in the city/state/country/globe.

So, It is very LEAST LIKELY to apply IP Whitelisting on such Experience APIs whose consumers are typically Mobile Phones or Tablets.

#### NEW QUESTION 21

The responses to some HTTP requests can be cached depending on the HTTP verb used in the request. According to the HTTP specification, for what HTTP verbs is this safe to do?

- A. PUT, POST, DELETE
- B. GET, HEAD, POST
- C. GET, PUT, OPTIONS
- D. GET, OPTIONS, HEAD

**Answer:** D

#### Explanation:

Correct Answer

GET, OPTIONS, HEAD

<http://restcookbook.com/HTTP%20Methods/idempotency/>

#### NEW QUESTION 26

A new upstream API is being designed to offer an SLA of 500 ms median and 800 ms maximum (99th percentile) response time. The corresponding API implementation needs to sequentially invoke 3 downstream APIs of very similar complexity.

The first of these downstream APIs offers the following SLA for its response time: median: 100 ms, 80th percentile: 500 ms, 95th percentile: 1000 ms.

If possible, how can a timeout be set in the upstream API for the invocation of the first downstream API to meet the new upstream API's desired SLA?

- A. Set a timeout of 50 ms; this times out more invocations of that API but gives additional room for retries
- B. Set a timeout of 100 ms; that leaves 400 ms for the other two downstream APIs to complete
- C. No timeout is possible to meet the upstream API's desired SLA; a different SLA must be negotiated with the first downstream API or invoke an alternative API
- D. Do not set a timeout; the Invocation of this API is mandatory and so we must wait until it responds

**Answer:** B

#### Explanation:

Correct Answer

Set a timeout of 100ms; that leaves 400ms for other two downstream APIs to complete

\*\*\*\*\* Key details to take from the given scenario:

>> Upstream API's designed SLA is 500ms (median). Lets ignore maximum SLA response times.

>> This API calls 3 downstream APIs sequentially and all these are of similar complexity.

>> The first downstream API is offering median SLA of 100ms, 80th percentile: 500ms; 95th percentile: 1000ms.

Based on the above details:

>> We can rule out the option which is suggesting to set 50ms timeout. Because, if the median SLA itself being offered is 100ms then most of the calls are going to timeout and time gets wasted in retried them and eventually gets exhausted with all retries. Even if some retries gets successful, the remaining time wont leave enough room for 2nd and 3rd downstream APIs to respond within time.

>> The option suggesting to NOT set a timeout as the invocation of this API is mandatory and so we must wait until it responds is silly. As not setting time out would go against the good implementation pattern and moreover if the first API is not responding within its offered median SLA 100ms then most probably it would either respond in 500ms (80th percentile) or 1000ms (95th percentile). In BOTH cases, getting a successful response from 1st downstream API does NO GOOD because already by this time the Upstream API SLA of 500 ms is breached. There is no time left to call 2nd and 3rd downstream APIs.

>> It is NOT true that no timeout is possible to meet the upstream APIs desired SLA.

As 1st downstream API is offering its median SLA of 100ms, it means MOST of the time we would get the responses within that time. So, setting a timeout of 100ms would be ideal for MOST calls as it leaves enough room of 400ms for remaining 2 downstream API calls.

#### NEW QUESTION 27

Refer to the exhibit. 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) 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

B) One set of APIs (Experience API, Process API, and System API) for the web app, and another set for the mobile app

C) 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

D) 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

A. Option A

B. Option B

C. Option C

D. Option D

**Answer: C**

**Explanation:**

Correct Answer

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

\*\*\*\*\* As per MuleSoft's API-led connectivity:

>> Experience APIs should be built as per each consumer needs and their experience.

>> Process APIs should contain all the orchestration logic to achieve the business functionality.

>> System APIs should be built for each backend system to unlock their data.

#### NEW QUESTION 31

What is a key requirement when using an external Identity Provider for Client Management in Anypoint Platform?

A. Single sign-on is required to sign in to Anypoint Platform

B. The application network must include System APIs that interact with the Identity Provider

C. To invoke OAuth 2.0-protected APIs managed by Anypoint Platform, API clients must submit access tokens issued by that same Identity Provider

D. APIs managed by Anypoint Platform must be protected by SAML 2.0 policies

**Answer:** C

**Explanation:**

<https://www.folkstalk.com/2019/11/mulesoft-integration-and-platform.html>

Correct Answer

To invoke OAuth 2.0-protected APIs managed by Anypoint Platform, API clients must submit access tokens issued by that same Identity Provider

\*\*\*\*\*

>> It is NOT necessary that single sign-on is required to sign in to Anypoint Platform because we are using an external Identity Provider for Client Management

>> It is NOT necessary that all APIs managed by Anypoint Platform must be protected by SAML 2.0 policies because we are using an external Identity Provider for Client Management

>> Not TRUE that the application network must include System APIs that interact with the Identity Provider because we are using an external Identity Provider for Client Management

Only TRUE statement in the given options is - "To invoke OAuth 2.0-protected APIs managed by Anypoint Platform, API clients must submit access tokens issued by that same Identity Provider"

References:

<https://docs.mulesoft.com/api-manager/2.x/external-oauth-2.0-token-validation-policy> <https://blogs.mulesoft.com/dev/api-dev/api-security-ways-to-authenticate-and-authorize/>

**NEW QUESTION 36**

A REST API is being designed to implement a Mule application.

What standard interface definition language can be used to define REST APIs?

- A. Web Service Definition Language(WSDL)
- B. OpenAPI Specification (OAS)
- C. YAML
- D. AsyncAPI Specification

**Answer:** B

**NEW QUESTION 37**

What Anypoint Connectors support transactions?

- A. Database, JMS, VM
- B. Database, 3MS, HTTP
- C. Database, JMS, VM, SFTP
- D. Database, VM, File

**Answer:** A

**NEW QUESTION 40**

A company has started to create an application network and is now planning to implement a Center for Enablement (C4E) organizational model. What key factor would lead the company to decide upon a federated rather than a centralized C4E?

- A. When there are a large number of existing common assets shared by development teams
- B. When various teams responsible for creating APIs are new to integration and hence need extensive training
- C. When development is already organized into several independent initiatives or groups
- D. When the majority of the applications in the application network are cloud based

**Answer:** C

**Explanation:**

Correct Answer

When development is already organized into several independent initiatives or groups

\*\*\*\*\*

>> It would require lot of process effort in an organization to have a single C4E team coordinating with multiple already organized development teams which are into several independent initiatives. A single C4E works well with different teams having at least a common initiative. So, in this scenario, federated C4E works well instead of centralized C4E.

**NEW QUESTION 42**

A retail company with thousands of stores has an API to receive data about purchases and insert it into a single database. Each individual store sends a batch of purchase data to the API about every 30 minutes. The API implementation uses a database bulk insert command to submit all the purchase data to a database using a custom JDBC driver provided by a data analytics solution provider. The API implementation is deployed to a single CloudHub worker. The JDBC driver processes the data into a set of several temporary disk files on the CloudHub worker, and then the data is sent to an analytics engine using a proprietary protocol. This process usually takes less than a few minutes. Sometimes a request fails. In this case, the logs show a message from the JDBC driver indicating an out-of-file-space message. When the request is resubmitted, it is successful. What is the best way to try to resolve this throughput issue?

- A. se a CloudHub autoscaling policy to add CloudHub workers
- B. Use a CloudHub autoscaling policy to increase the size of the CloudHub worker
- C. Increase the size of the CloudHub worker(s)
- D. Increase the number of CloudHub workers

**Answer:** D

**Explanation:**

Correct Answer

Increase the size of the CloudHub worker(s)

\*\*\*\*\*

The key details that we can take out from the given scenario are:

>> API implementation uses a database bulk insert command to submit all the purchase data to a database



>> JDBC driver processes the data into a set of several temporary disk files on the CloudHub worker  
>> Sometimes a request fails and the logs show a message indicating an out-of-file-space message Based on above details:  
>> Both auto-scaling options does NOT help because we cannot set auto-scaling rules based on error messages. Auto-scaling rules are kicked-off based on CPU/Memory usages and not due to some given error or disk space issues.  
>> Increasing the number of CloudHub workers also does NOT help here because the reason for the failure is not due to performance aspects w.r.t CPU or Memory. It is due to disk-space.  
>> Moreover, the API is doing bulk insert to submit the received batch data. Which means, all data is handled by ONE worker only at a time. So, the disk space issue should be tackled on "per worker" basis. Having multiple workers does not help as the batch may still fail on any worker when disk is out of space on that particular worker.  
Therefore, the right way to deal this issue and resolve this is to increase the vCore size of the worker so that a new worker with more disk space will be provisioned.

#### NEW QUESTION 47

A system API has a guaranteed SLA of 100 ms per request. The system API is deployed to a primary environment as well as to a disaster recovery (DR) environment, with different DNS names in each environment. An upstream process API invokes the system API and the main goal of this process API is to respond to client requests in the least possible time. In what order should the system APIs be invoked, and what changes should be made in order to speed up the response time for requests from the process API?

- A. In parallel, invoke the system API deployed to the primary environment and the system API deployed to the DR environment, and ONLY use the first response
- B. In parallel, invoke the system API deployed to the primary environment and the system API deployed to the DR environment using a scatter-gather configured with a timeout, and then merge the responses
- C. Invoke the system API deployed to the primary environment, and if it fails, invoke the system API deployed to the DR environment
- D. Invoke ONLY the system API deployed to the primary environment, and add timeout and retry logic to avoid intermittent failures

**Answer:** A

#### Explanation:

Correct Answer

In parallel, invoke the system API deployed to the primary environment and the system API deployed to the DR environment, and ONLY use the first response.  
\*\*\*\*\*

>> The API requirement in the given scenario is to respond in least possible time.  
>> The option that is suggesting to first try the API in primary environment and then fallback to API in DR environment would result in successful response but NOT in least possible time. So, this is NOT a right choice of implementation for given requirement.  
>> Another option that is suggesting to ONLY invoke API in primary environment and to add timeout and retries may also result in successful response upon retries but NOT in least possible time. So, this is also NOT a right choice of implementation for given requirement.  
>> One more option that is suggesting to invoke API in primary environment and API in DR environment in parallel using Scatter-Gather would result in wrong API response as it would return merged results and moreover, Scatter-Gather does things in parallel which is true but still completes its scope only on finishing all routes inside it. So again, NOT a right choice of implementation for given requirement  
The Correct choice is to invoke the API in primary environment and the API in DR environment parallelly, and using ONLY the first response received from one of them.

#### NEW QUESTION 51

Refer to the exhibit.

A developer is building a client application to invoke an API deployed to the STAGING environment that is governed by a client ID enforcement policy. What is required to successfully invoke the API?

- A. The client ID and secret for the Anypoint Platform account owning the API in the STAGING environment
- B. The client ID and secret for the Anypoint Platform account's STAGING environment
- C. The client ID and secret obtained from Anypoint Exchange for the API instance in the STAGING environment
- D. A valid OAuth token obtained from Anypoint Platform and its associated client ID and secret

**Answer:** C

#### Explanation:

Correct Answer

The client ID and secret obtained from Anypoint Exchange for the API instance in the STAGING environment  
\*\*\*\*\*

>> We CANNOT use the client ID and secret of Anypoint Platform account or any individual environments for accessing the APIs  
>> As the type of policy that is enforced on the API in question is "Client ID Enforcment Policy", OAuth token based access won't work.  
Right way to access the API is to use the client ID and secret obtained from Anypoint Exchange for the API instance in a particular environment we want to work on.  
References:  
Managing API instance Contracts on API Manager <https://docs.mulesoft.com/api-manager/1.x/request-access-to-api-task> <https://docs.mulesoft.com/exchange/to-request-access> <https://docs.mulesoft.com/api-manager/2.x/policy-mule3-client-id-based-policies>

#### NEW QUESTION 52

.....

## Thank You for Trying Our Product

### We offer two products:

1st - We have Practice Tests Software with Actual Exam Questions

2nd - Questions and Answers in PDF Format

### MCPA-Level-1 Practice Exam Features:

- \* MCPA-Level-1 Questions and Answers Updated Frequently
- \* MCPA-Level-1 Practice Questions Verified by Expert Senior Certified Staff
- \* MCPA-Level-1 Most Realistic Questions that Guarantee you a Pass on Your FirstTry
- \* MCPA-Level-1 Practice Test Questions in Multiple Choice Formats and Updatesfor 1 Year

**100% Actual & Verified — Instant Download, Please Click**  
**[Order The MCPA-Level-1 Practice Test Here](#)**