MCIA-LEVEL-1-MAINTENANCE Q&As

MuleSoft Certified Integration Architect - Level 1 MAINTENANCE

Pass Mulesoft MCIA-LEVEL-1-MAINTENANCE Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

https://www.pass4itsure.com/mcia-level-1-maintenance.html

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by Mulesoft Official Exam Center

QUESTION 1

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 100 ms; that leaves 400 ms for the other two downstream APIs to complete

B. Do not set a timeout; the Invocation of this API Is mandatory and so we must wait until it responds

C. Set a timeout of 50 ms; this times out more invocations of that API but gives additional room for retries

D. 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

Correct Answer: D

Before we answer this question , we need to understand what median (50th percentile) and 80th percentile means. If the 50th percentile (median) of a response time is 500ms that means that 50% of my transactions are either as fast or faster than 500ms. If the 90th percentile of the same transaction is at 1000ms it means that 90% are as fast or faster and only 10% are slower. Now as per upstream SLA , 99th percentile is 800 ms which means 99% of the incoming requests should have response time less than or equal to 800 ms. But as per one of the backend API , their 95th percentile is 1000 ms which means that backend API will take 1000 ms or less than that for 95% of. requests. As there are three API invocation from upstream API , we can not conclude a timeout that can be set to meet the desired SLA as backend SLA\\'s do not support it. Let see why other answers are not correct. 1) Do not set a timeout --> This can potentially violate SLA\\'s of upstream API 2) Set a timeout of 100 ms; ---> This will not work as backend API has 100 ms as median meaning only 50% requests will be answered in this time and we will get timeout for 50% of the requests. Important thing to note here is, All APIs need to be executed sequentially, so if you get timeout in first API, there is no use of going to second and third API. As a service provider you wouldn\\'t want to keep 50% of your consumers dissatisfied. So not the best option to go with. *To quote an example: Let\\'s assume you have built an API to update customer contact details.

-First API is fetching customer number based on login credentials

-Second API is fetching Info in 1 table and returning unique key

-Third API, using unique key provided in second API as primary key, updating remaining details * Now consider, if API times out in first API and can\\'t fetch customer number, in this case, it\\'s useless to call API 2 and 3 and that is why question mentions specifically that all APIs need to be executed sequentially. 3) Set a timeout of 50 ms --> Again not possible due to the same reason as above Hence correct answer is 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

QUESTION 2

A Mule application is built to support a local transaction for a series of operations on a single database. The mule application has a Scatter-Gather scope that participates in the local transaction.

What is the behavior of the Scatter-Gather when running within this local transaction?

A. Execution of all routes within Scatter-Gather occurs in parallel Any error that occurs inside Scatter-Gather will result in a roll back of all the database operations

B. Execution of all routes within Scatter-Gather occurs sequentially Any error that occurs inside Scatter-Gather will be handled by error handler and will not result in roll back

C. Execution of all routes within Scatter-Gather occurs sequentially Any error that occurs inside Scatter-Gather will result in a roll back of all the database operations

D. Execution of all routes within Scatter-Gather occurs in parallel Any error that occurs inside Scatter-Gather will be handled by error handler and will not result in roll back

Correct Answer: A

QUESTION 3

A company is planning to extend its Mule APIs to the Europe region. Currently all new applications are deployed to Cloudhub in the US region following this naming convention {API name}-{environment}. for example, Orders-SAPI-dev, Orders-SAPI-prod etc. Considering there is no network restriction to block communications between API\\'s, what strategy should be implemented in order to apply the same new API\\'s running in the EU region of CloudHub as well to minimize latency between API\\'s

and target users and systems in Europe?

A. Set region property to Europe (eu-de) in API manager for all the mule application No need to change the naming convention

B. Set region property to Europe (eu-de) in API manager for all the mule application Change the naming convention to {API name}-{environment}-{region} and communicate this change to the consuming applications and users

C. Set region property to Europe (eu-de) in runtime manager for all the mule application No need to change the naming convention

D. Set region property to Europe (eu-de) in runtime manager for all the mule application Change the naming convention to {API name}-{environment}-{region} and communicate this change to the consuming applications and users

Correct Answer: D

QUESTION 4

A project uses Jenkins to implement CI/CD process. It was observed that each Mule package contains some of the Jenkins files and folders for configurations of CI/CD jobs.

As these files and folders are not part of the actual package, expectation is that these should not be part of deployed archive.

Which file can be used to exclude these files and folders from the deployed archive?

A. muleignore

B. _unTrackMule

C. muleInclude

D. _muleExclude

Correct Answer: D

QUESTION 5

An organization has several APIs that accept JSON data over HTTP POST. The APIs are all publicly available and are associated with several mobile applications and web applications. The organization does NOT want to use any authentication or compliance policies for these APIs, but at the same time, is worried that some bad actor could send payloads that could somehow compromise the applications or servers running the API implementations. What out-of-thebox Anypoint Platform policy can address exposure to this threat?

A. Apply a Header injection and removal policy that detects the malicious data before it is used

B. Apply an IP blacklist policy to all APIs; the blacklist will Include all bad actors

C. Shut out bad actors by using HTTPS mutual authentication for all API invocations

D. Apply a JSON threat protection policy to all APIs to detect potential threat vectors

Correct Answer: D

We need to note few things about the scenario which will help us in reaching the correct solution. Point 1 : The APIs are all publicly available and are associated with several mobile applications and web applications. This means Apply an IP blacklist policy is not viable option. as blacklisting IPs is limited to partial web traffic. It can\\'t be useful for traffic from mobile application Point 2 : The organization does NOT want to use any authentication or compliance policies for these APIs. This means we can not apply HTTPS mutual authentication scheme. Header injection or removal will not help the purpose. By its nature, JSON is vulnerable to JavaScript injection. When you parse the JSON object, the malicious code inflicts its damages. An inordinate increase in the size and depth of the JSON payload can indicate injection. Applying the JSON threat protection policy can limit the size of your JSON payload and thwart recursive additions to the JSON hierarchy. Hence correct answer is Apply a JSON threat protection policy to all APIs to detect potential threat vectors