Free Questions for MCPA-Level-1 by vceexamstest

Shared by Sherman on 12-12-2023

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

Which of the following best fits the definition of API-led connectivity?

Options:

A- API-led connectivity is not just an architecture or technology but also a way to organize people and processes for efficient IT delivery in the organization

B- API-led connectivity is a 3-layered architecture covering Experience, Process and System layers

C- API-led connectivity is a technology which enabled us to implement Experience, Process and System layer based APIs

Answer:

A

Explanation:

Correct Answer: API-led connectivity is not just an architecture or technology but also a way to organize people and processes for efficient IT delivery in the organization.

****

Question 2

Question Type: MultipleChoice

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?

Options:

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.

Question 3

Question Type: MultipleChoice

The application network is recomposable: it is built for change because it "bends but does not break"

Options:

A- TRUE

B- FALSE

****

>> Application Network is a disposable architecture.

>> Which means, it can be altered without disturbing entire architecture and its components.

>> It bends as per requirements or design changes but does not break

Answer:

A

Question 4

Question Type: MultipleChoice

A company has created a successful enterprise data model (EDM). The company is committed to building an application network by adopting modern APIs as a core enabler of the company's IT operating model. At what API tiers (experience, process, system) should the company require reusing the EDM when designing modern API data models?

Options:

A- At the experience and process tiers

B- At the experience and system tiers

C- At the process and system tiers

D- At the experience, process, and system tiers

Answer:

C

Explanation:

Correct Answer: At the process and system tiers

****

>> Experience Layer APIs are modeled and designed exclusively for the end user's experience. So, the data models of experience layer vary based on the nature and type of such API consumer. For example, Mobile consumers will need light-weight data models to transfer with ease on the wire, where as web-based consumers will need detailed data models to render most of the info on web pages, so on. So, enterprise data models fit for the purpose of canonical models but not of good use for experience APIs.

>> That is why, EDMs should be used extensively in process and system tiers but NOT in experience tier.

Question 5

Question Type: MultipleChoice

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?

Options:

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.

Question 6

Question Type: **MultipleChoice**

An API implementation returns three X-RateLimit-* HTTP response headers to a requesting API client. What type of information do these response headers indicate to the API client?

Options:

A- The error codes that result from throttling

B- A correlation ID that should be sent in the next request

C- The HTTP response size

D- The remaining capacity allowed by the API implementation

Answer:

D

Explanation:

Correct Answer: The remaining capacity allowed by the API implementation.

****

>> Reference: https://docs.mulesoft.com/api-manager/2.x/rate-limiting-and-throttling-sla-based-policies#response-headers

Response Headers

Three headers are included in request responses that inform users about the SLA restrictions and inform them when nearing the threshold When the SLA enforces multiple policies that limit request throughput, a single set of headers pertaining to the most restrictive of the policies provides this information.

For example, a user of your API may receive a response that includes these headers:

```
X-RateLimit-Limit: 20
X-RateLimit-Remaining: 14
X-RateLimit-Reset: 19100
```

Within the next 19100 milliseconds, only 14 more requests are allowed by the SLA, which is set to allow 20 within this time-window.

Question 7

Question Type: **MultipleChoice**

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 APIs public portal. The API endpoint does NOT change in the new version. How should the developer of an API client respond to this change?

Options:

A- The API producer should be requested to run the old version in parallel with the new one

B- The API producer should be contacted to understand the change to existing functionality

C- The API client code only needs to be changed if it needs to take advantage of the new features

D- The API clients need to update the code on their side and need to do full regression

Answer:

C

Question 8

Question Type: **MultipleChoice**

A company requires Mule applications deployed to CloudHub to be isolated between non-production and production environments. This is so Mule applications deployed to non-production environments can only access backend systems running in their customer-hosted non-production environment, and so Mule applications deployed to production environments can only access backend systems running in their customer-hosted production environment. How does MuleSoft recommend modifying Mule applications, configuring environments, or changing infrastructure to support this type of per-environment isolation between Mule applications and backend systems?

Options:

A- Modify properties of Mule applications deployed to the production Anypoint Platform environments to prevent access from non-production Mule applications

B- Configure firewall rules in the infrastructure inside each customer-hosted environment so that only IP addresses from the corresponding Anypoint Platform environments are allowed to communicate with corresponding backend systems

C- Create non-production and production environments in different Anypoint Platform business groups

D- Create separate Anypoint VPCs for non-production and production environments, then configure connections to the backend systems in the corresponding customer-hosted environments

Answer:

D

Explanation:

Correct Answer:Create separate Anypoint VPCs for non-production and production environments, then configure connections to the backend systems in the corresponding customer-hosted environments.

****

>>Creating different Business Groups does NOT make any difference w.r.t accessing the non-prod and prod customer-hosted environments. Still they will be accessing from both Business Groups unless process network restrictions are put in place.

>>We need to modify or couple the Mule Application Implementations with the environment. In fact, we should never implements application coupled with environments by binding them in the properties. Only basic things like endpoint URL etc should be bundled in properties but not environment level access restrictions.

>>IP addresses on CloudHub are dynamic until unless a special static addresses are assigned. So it is not possible to setup firewall rules in customer-hosted infrastrcture. More over, even if static IP addresses are assigned, there could be 100s of applications running on cloudhub and setting up rules for all of them would be a hectic task, non-maintainable and definitely got a good practice.

>>Thebest practice recommendedby Mulesoft (In fact any cloud provider), is to have your Anypoint VPCs seperated for Prod and Non-Prod and perform the VPC peering or VPN tunneling for these Anypoint VPCs to respective Prod and Non-Prod customer-hosted environment networks.

Question 9

Question Type: **MultipleChoice**

An organization wants to make sure only known partners can invoke the organization's APIs. To achieve this security goal, the organization wants to enforce a Client ID Enforcement policy in API Manager so that only registered partner applications can invoke the organization's APIs. In what type of API implementation does MuleSoft recommend adding an API proxy to enforce the Client ID Enforcement policy, rather than embedding the policy directly in the application's JVM?

Options:

A- A Mule 3 application using APIkit

B- A Mule 3 or Mule 4 application modified with custom Java code

C- A Mule 4 application with an API specification

D- A Non-Mule application

Answer:

D

Explanation:

Correct Answer: A Non-Mule application

****

>> All type of Mule applications (Mule 3/ Mule 4/ with APIkit/ with Custom Java Code etc) running on Mule Runtimes support the Embedded Policy Enforcement on them.

>> The only option that cannot have or does not support embedded policy enforcement and must have API Proxy is for Non-Mule Applications.

So, Non-Mule application is the right answer.

Question 10

Question Type: MultipleChoice

A company uses a hybrid Anypoint Platform deployment model that combines the EU control plane with customer-hosted Mule runtimes. After successfully testing a Mule API implementation in the Staging environment, the Mule API implementation is set with environment-specific properties and must be promoted to the Production environment. What is a way that MuleSoft recommends to configure the Mule API implementation and automate its promotion to the Production environment?

Options:

A- Bundle properties files for each environment into the Mule API implementation's deployable archive, then promote the Mule API implementation to the Production environment using Anypoint CLI or the Anypoint Platform REST APIsB.

B- Modify the Mule API implementation's properties in the API Manager Properties tab, then promote the Mule API implementation to the Production environment using API Manager

C- Modify the Mule API implementation's properties in Anypoint Exchange, then promote the Mule API implementation to the Production environment using Runtime Manager

D- Use an API policy to change properties in the Mule API implementation deployed to the Staging environment and another API policy to deploy the Mule API implementation to the Production environment

Answer:

A

Explanation:

Correct Answer: Bundle properties files for each environment into the Mule API implementation's deployable archive, then promote the Mule API implementation to the Production environment using Anypoint CLI or the Anypoint Platform REST APIs

****

>> Anypoint Exchange is for asset discovery and documentation. It has got no provision to modify the properties of Mule API implementations at all.

>> API Manager is for managing API instances, their contracts, policies and SLAs. It has also got no provision to modify the properties of API implementations.

>> API policies are to address Non-functional requirements of APIs and has again got no provision to modify the properties of API implementations.

So, the right way and recommended way to do this as part of development practice is to bundle properties files for each environment into the Mule API implementation and just point and refer to respective file per environment.

Question 11

Question Type: MultipleChoice

A system API is deployed to a primary environment as well as to a disaster recovery (DR) environment, with different DNS names in each environment. A process API is a client to the system API and is being rate limited by the system API, with different limits in each of the environments. The system API's DR environment provides only 20% of the rate limiting offered by the primary environment. What is the best API fault-tolerant invocation strategy to reduce overall errors in the process API, given these conditions and constraints?

Options:

A- Invoke the system API deployed to the primary environment; add timeout and retry logic to the process API to avoid intermittent failures; if it still fails, invoke the system API deployed to the DR environment

B- Invoke the system API deployed to the primary environment; add retry logic to the process API to handle intermittent failures by invoking the system API deployed to the DR environment

C- In parallel, invoke the system API deployed to the primary environment and the system API deployed to the DR environment; add timeout and retry logic to the process API to avoid intermittent failures; add logic to the process API to combine the results

D- Invoke the system API deployed to the primary environment; add timeout and retry logic to the process API to avoid intermittent failures; if it still fails, invoke a copy of the process API deployed to the DR environment

Answer:

A

Explanation:

Correct Answer: Invoke the system API deployed to the primary environment; add timeout and retry logic to the process API to avoid intermittent failures; if it still fails, invoke the system API deployed to the DR environment

****

There is one important consideration to be noted in the question which is - System API in DR environment provides only 20% of the rate limiting offered by the primary environment. So, comparitively, very less calls will be allowed into the DR environment API opposed to its primary environment. With this in mind, lets analyse what is the right and best fault-tolerant invocation strategy.

1. Invoking both the system APIs in parallel is definitely NOT a feasible approach because of the 20% limitation we have on DR environment. Calling in parallel every time would easily and quickly exhaust the rate limits on DR environment and may not give chance to genuine intermittent error scenarios to let in during the time of need.

2. Another option given is suggesting to add timeout and retry logic to process API while invoking primary environment's system API. This is good so far. However, when all retries failed, the option is suggesting to invoke the copy of process API on DR environment which is not right or recommended. Only system API is the one to be considered for fallback and not the whole process API. Process APIs usually have lot of heavy orchestration calling many other APIs which we do not want to repeat again by calling DR's process API. So this option is NOT right.

3. One more option given is suggesting to add the retry (no timeout) logic to process API to directly retry on DR environment's system API instead of retrying the primary environment system API first. This is not at all a proper fallback. A proper fallback should occur only after all retries are performed and exhausted on Primary environment first. But here, the option is suggesting to directly retry fallback API on first failure itself without trying main API. So, this option is NOT right too.

This leaves us one option which is right and best fit.

- Invoke the system API deployed to the primary environment

- Add Timeout and Retry logic on it in process API

- If it fails even after all retries, then invoke the system API deployed to the DR environment.

Question 12

Question Type: **MultipleChoice**

An API client calls one method from an existing API implementation. The API implementation is later updated. What change to the API implementation would require the API client's invocation logic to also be updated?

Options:

A- When the data type of the response is changed for the method called by the API client

B- When a new method is added to the resource used by the API client

C- When a new required field is added to the method called by the API client

D- When a child method is added to the method called by the API client

Answer:

C

Explanation:

Correct Answer: When a new required field is added to the method called by the API client

****

>> Generally, the logic on API clients need to be updated when the API contract breaks.

>> When a new method or a child method is added to an API , the API client does not break as it can still continue to use its existing method. So these two options are out.

>> We are left for two more where 'datatype of the response if changed' and 'a new required field is added'.

>> Changing the datatype of the response does break the API contract. However, the question is insisting on the 'invocation' logic and not about the response handling logic. The API client can still invoke the API successfully and receive the response but the response will have a different datatype for some field.

>> Adding a new required field will break the API's invocation contract. When adding a new required field, the API contract breaks the RAML or API spec agreement that the API client/API consumer and API provider has between them. So this requires the API client invocation logic to also be updated.