Free Questions for MuleSoft-Platform-Architect-I by certsdeals

Shared by Russell on 22-07-2024

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

Traffic is routed through an API proxy to an API implementation. The API proxy is managed by API Manager and the API implementation is deployed to a CloudHub VPC using Runtime Manager. API policies have been applied to this API. In this deployment scenario, at what point are the API policies enforced on incoming API client requests?

Options:

A- At the API proxy

B- At the API implementation

C- At both the API proxy and the API implementation

D- At a MuleSoft-hosted load balancer

Answer:

A

Explanation:

Correct Answer : At the API proxy

****

>> API Policies can be enforced at two places in Mule platform.

>> One - As an Embedded Policy enforcement in the same Mule Runtime where API implementation is running.

>> Two - On an API Proxy sitting in front of the Mule Runtime where API implementation is running.

>> As the deployment scenario in the question has API Proxy involved, the policies will be enforced at the API Proxy.

Question 2

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.

Question 3

Question Type: MultipleChoice

An organization has created an API-led architecture that uses various API layers to integrate mobile clients with a backend system. The backend system consists of a number of specialized components and can be accessed via a REST API. The process and experience APIs share the same bounded-context model that is different from the backend data model. What additional canonical models, bounded-context models, or anti-corruption layers are best added to this architecture to help process data consumed from the backend system?

Options:

A- Create a bounded-context model for every layer and overlap them when the boundary contexts overlap, letting API developers know about the differences between upstream and downstream data models

B- Create a canonical model that combines the backend and API-led models to simplify and unify data models, and minimize data transformations.

C- Create a bounded-context model for the system layer to closely match the backend data model, and add an anti-corruption layer to let

the different bounded contexts cooperate across the system and process layers

D- Create an anti-corruption layer for every API to perform transformation for every data model to match each other, and let data simply travel between APIs to avoid the complexity and overhead of building canonical models

Answer:

C

Explanation:

Correct Answer : Create a bounded-context model for the system layer to closely match the backend data model, and add an anti-corruption layer to let the different bounded contexts cooperate across the system and process layers

****

>> Canonical models are not an option here as the organization has already put in efforts and created bounded-context models for Experience and Process APIs.

>> Anti-corruption layers for ALL APIs is unnecessary and invalid because it is mentioned that experience and process APIs share same bounded-context model. It is just the System layer APIs that need to choose their approach now.

>> So, having an anti-corruption layer just between the process and system layers will work well. Also to speed up the approach, system APIs can mimic the backend system data model.

Question 4

Question Type: MultipleChoice

Which of the following sequence is correct?

Options:

A- API Client implementes logic to call an API >> API Consumer requests access to API >> API Implementation routes the request to >> API

B- API Consumer requests access to API >> API Client implementes logic to call an API >> API routes the request to >> API Implementation

C- API Consumer implementes logic to call an API >> API Client requests access to API >> API Implementation routes the request to >> API

D- API Client implementes logic to call an API >> API Consumer requests access to API >> API routes the request to >> API Implementation

Answer:

B

Explanation:

Correct Answer : API Consumer requests access to API >> API Client implementes logic to call an API >> API routes the request to >> API Implementation

****

>> API consumer does not implement any logic to invoke APIs. It is just a role. So, the option stating 'API Consumer implementes logic to call an API' is INVALID.

>> API Implementation does not route any requests. It is a final piece of logic where functionality of target systems is exposed. So, the requests should be routed to the API implementation by some other entity. So, the options stating 'API Implementation routes the request to >> API' is INVALID

>> The statements in one of the options are correct but sequence is wrong. The sequence is given as 'API Client implementes logic to call an API >> API Consumer requests access to API >> API routes the request to >> API Implementation'. Here, the statements in the options are VALID but sequence is WRONG.

>> Right option and sequence is the one where API consumer first requests access to API on Anypoint Exchange and obtains client credentials. API client then writes logic to call an API by using the access client credentials requested by API consumer and the requests will be routed to API implementation via the API which is managed by API Manager.

Question 5

Question Type: **MultipleChoice**

Which of the below, when used together, makes the IT Operational Model effective?

Options:

A- Create reusable assets, Do marketing on the created assets across organization, Arrange time to time LOB reviews to ensure assets are being consumed or not

B- Create reusable assets, Make them discoverable so that LOB teams can self-serve and browse the APIs, Get active feedback and usage metrics

C- Create resuable assets, make them discoverable so that LOB teams can self-serve and browse the APIs

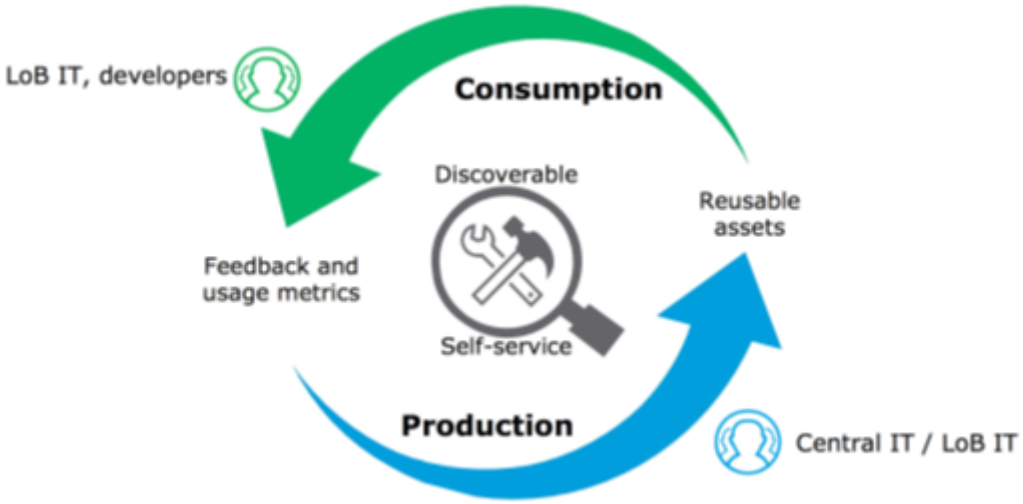
Answer:

C

Explanation:

Correct Answer :Create reusable assets, Make them discoverable so that LOB teams can self-serve and browse the APIs, Get active feedback and usage metrics.

****



Question 6

Question Type: MultipleChoice

A set of tests must be performed prior to deploying API implementations to a staging environment. Due to data security and access restrictions, untested APIs cannot be granted access to the backend systems, so instead mocked data must be used for these tests. The amount of available mocked data and its contents is sufficient to entirely test the API implementations with no active connections to the backend systems. What type of tests should be used to incorporate this mocked data?

Options:

A- Integration tests

B- Performance tests

C- Functional tests (Blackbox)

D- Unit tests (Whitebox)

Answer:

D

Explanation:

Correct Answer : Unit tests (Whitebox)

****

As per general IT testing practice and MuleSoft recommended practice, Integration and Performance tests should be done on full end to end setup for right evaluation. Which means all end systems should be connected while doing the tests. So, these options are OUT and we are left with Unit Tests and Functional Tests.

As per attached reference documentation from MuleSoft:

Unit Tests - are limited to the code that can be realistically exercised without the need to run it inside Mule itself. So good candidates are Small pieces of modular code, Sub Flows, Custom transformers, Custom components, Custom expression evaluators etc.

Functional Tests - are those that most extensively exercise your application configuration. In these tests, you have the freedom and tools for simulating happy and unhappy paths. You also have the possibility to create stubs for target services and make them success or fail to easily simulate happy and unhappy paths respectively.

As the scenario in the question demands for API implementation to be tested before deployment to Staging and also clearly indicates that there is enough/ sufficient amount of mock data to test the various components of API implementations with no active connections to the backend systems, Unit Tests are the one to be used to incorporate this mocked data.

Question 7

Question Type: **MultipleChoice**

A company wants to move its Mule API implementations into production as quickly as possible. To protect access to all Mule application data and metadata, the company requires that all Mule applications be deployed to the company's customer-hosted infrastructure within the corporate firewall. What combination of runtime plane and control plane options meets these project lifecycle goals?

Options:

A- Manually provisioned customer-hosted runtime plane and customer-hosted control plane

B- MuleSoft-hosted runtime plane and customer-hosted control plane

C- Manually provisioned customer-hosted runtime plane and MuleSoft-hosted control plane

D- iPaaS provisioned customer-hosted runtime plane and MuleSoft-hosted control plane

Answer:

A

Explanation:

Correct Answer : Manually provisioned customer-hosted runtime plane and customer-hosted control plane

****

There are two key factors that are to be taken into consideration from the scenario given in the question.

>> Company requires both data and metadata to be resided within the corporate firewall

>> Company would like to go with customer-hosted infrastructure.

Any deployment model that is to deal with the cloud directly or indirectly (Mulesoft-hosted or Customer's own cloud like Azure, AWS) will have to share atleast the metadata.

Application data can be controlled inside firewall by having Mule Runtimes on customer hosted runtime plane. But if we go with Mulsoft-hosted/ Cloud-based control plane, the control plane required atleast some minimum level of metadata to be sent outside the corporate firewall.

As the customer requirement is pretty clear about the data and metadata both to be within the corporate firewall, even though customer wants to move to production as quickly as possible, unfortunately due to the nature of their security requirements, they have no other option but to go with manually provisioned customer-hosted runtime plane and customer-hosted control plane.