



MuleSoft

Exam Questions MCPA-Level-1

MuleSoft Certified Platform Architect - Level 1

About ExamBible

[Your Partner of IT Exam](#)

Found in 1998

ExamBible is a company specialized on providing high quality IT exam practice study materials, especially Cisco CCNA, CCDA, CCNP, CCIE, Checkpoint CCSE, CompTIA A+, Network+ certification practice exams and so on. We guarantee that the candidates will not only pass any IT exam at the first attempt but also get profound understanding about the certificates they have got. There are so many alike companies in this industry, however, ExamBible has its unique advantages that other companies could not achieve.

Our Advances

* 99.9% Uptime

All examinations will be up to date.

* 24/7 Quality Support

We will provide service round the clock.

* 100% Pass Rate

Our guarantee that you will pass the exam.

* Unique Gurantee

If you do not pass the exam at the first time, we will not only arrange FULL REFUND for you, but also provide you another exam of your claim, ABSOLUTELY FREE!

NEW QUESTION 1

How are an API implementation, API client, and API consumer combined to invoke and process an API?

- A. The API consumer creates an API implementation, which receives API invocations from an API such that they are processed for an API client
- B. The API client creates an API consumer, which receives API invocations from an API such that they are processed for an API implementation
- C. The API consumer creates an API client, which sends API invocations to an API such that they are processed by an API implementation
- D. The API client creates an API consumer, which sends API invocations to an API such that they are processed by an API implementation

Answer: C

Explanation:

Correct Answer

The API consumer creates an API client, which sends API invocations to an API such that they are processed by an API implementation

***** Terminology:

>> API Client - It is a piece of code or program that is written to invoke an API

>> API Consumer - An owner/entity who owns the API Client. API Consumers write API clients.

>> API - The provider of the API functionality. Typically an API Instance on API Manager where they are managed and operated.

>> API Implementation - The actual piece of code written by API provider where the functionality of the API is implemented. Typically, these are Mule Applications running on Runtime Manager.

NEW QUESTION 2

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 a 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) solve the issue. Both help 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 add extra hands 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 3

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

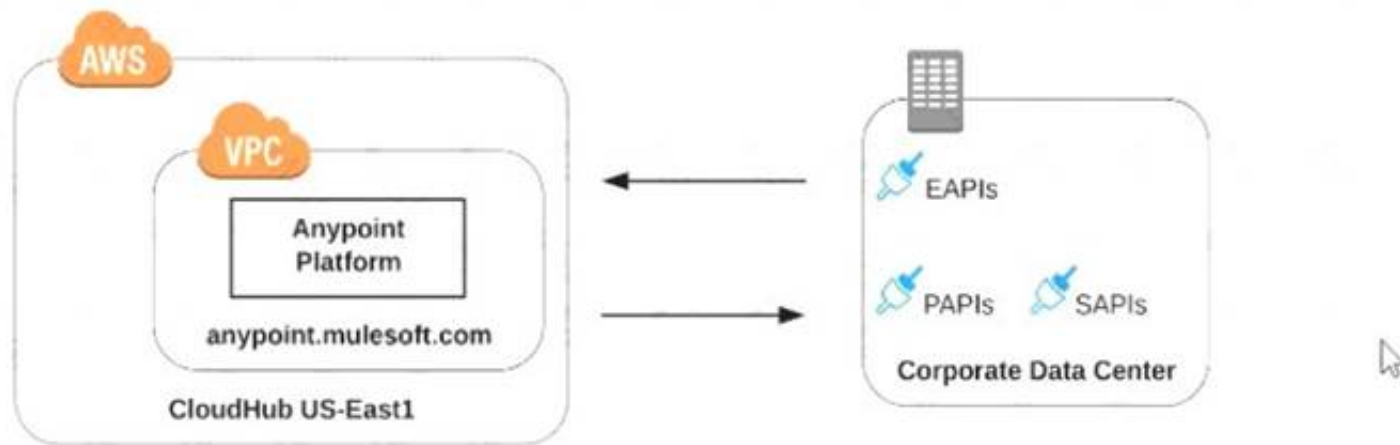
>> java.util.WeakHashMap needs a completely custom implementation of cache from scratch using Java code and is limited to the JVM where it is running. Which means the state in the cache is not worker aware when running on multiple workers. This type of cache is local to the worker. So, this is neither 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 4

Refer to the exhibit.



what is true when using customer-hosted Mule runtimes with the MuleSoft-hosted Anypoint Platform control plane (hybrid deployment)?

- A. Anypoint Runtime Manager initiates a network connection to a Mule runtime in order to deploy Mule applications
- B. The MuleSoft-hosted Shared Load Balancer can be used to load balance API invocations to the Mule runtimes
- C. API implementations can run successfully in customer-hosted Mule runtimes, even when they are unable to communicate with the control plane
- D. Anypoint Runtime Manager automatically ensures HA in the control plane by creating a new Mule runtime instance in case of a node failure

Answer: C

Explanation:

Correct Answer

API implementations can run successfully in customer-hosted Mule runtimes, even when they are unable to communicate with the control plane.

>> We CANNOT use Shared Load balancer to load balance APIs on customer hosted runtimes

◦ Load balancing

Load balancing is not provided for hybrid deployments. You can manage load balancing with the tools connected to your on-premises resources.

>> For Hybrid deployment models, the on-premises are first connected to Runtime Manager using Runtime Manager agent. So, the connection is initiated first from On-premises to Runtime Manager. Then all control can be done from Runtime Manager.

>> Anypoint Runtime Manager CANNOT ensure automatic HA. Clusters/Server Groups etc should be configured before hand.

Only TRUE statement in the given choices is, API implementations can run successfully in customer-hosted Mule runtimes, even when they are unable to communicate with the control plane. There are several references below to justify this statement.

References:

<https://docs.mulesoft.com/runtime-manager/deployment-strategies#hybrid-deployments> <https://help.mulesoft.com/s/article/On-Premise-Runtimes-Disconnected-From-US-Control-Plane-June-18th-2018> <https://help.mulesoft.com/s/article/Runtime-Manager-cannot-manage-On-Prem-Applications-and-Servers-from->
<https://help.mulesoft.com/s/article/On-premise-Runtimes-Appear-Disconnected-in-Runtime-Manager-May-29th>

On-Premise Runtimes Disconnected From US Control Plane - June 18th 2018

Jun 19, 2018 - RCA

Content

Impacted Platforms Impacted Duration

Anypoint Runtime Manager / On-Prem Runtimes	During this time frame, on-prem runtimes appeared disconnected from the US Anypoint Control Plane: June 18, 2018 10:35 AM PST to June 18, 2018 11:12 AM PST
---	--

Incident Description

On-premises applications weren't able to connect to Anypoint Runtime Manager during the length of the incident, which made on-premises runtimes to throw errors in their logs because they received network disconnect messages from the control plane. Other than generating the log as mentioned above entries, on-premises runtimes and applications were not impacted.

=====

Runtime Manager cannot manage On-Prem Applications and Servers from US Control Plane - June 25th 2019

🕒 Jul 3, 2019 - RCA

Content

Incident Summary

Between 2:51 p.m. PT June 25th and 12:41 a.m. PT June 26th, customers were not able to manage their On-Prem applications and servers. The availability of running applications and runtimes were not impacted.

Impacted Platforms Impact Duration

Impacted Platforms	Impact Duration
US-Prod	9 hours and 50 minutes

On-premise Runtimes Appear Disconnected in Runtime Manager - May 29th 2018

🕒 Jun 2, 2018 - RCA

Content

Impacted Platforms Impacted Duration

Impacted Platforms	Impacted Duration
Anypoint Runtime Manager / On-Prem Runtimes	During this time frame, on-prem runtimes appeared disconnected from the US Anypoint Control Plane: Tuesday, May 29, 2018, 3:35 AM PDT to 4:27 AM PDT

Incident Description

During the incident time frame, managed Runtimes running on-premises disconnected from the US Anypoint Platform Control Plane and may have encountered recurrent re-connection errors. Customers were unable to manage applications running on those runtimes or register new ones during this time. Runtimes and Applications continued to operate without impact.

NEW QUESTION 5

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

- 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.

NEW QUESTION 6

What condition requires using a CloudHub Dedicated Load Balancer?

- A. When cross-region load balancing is required between separate deployments of the same Mule application
- B. When custom DNS names are required for API implementations deployed to customer-hosted Mule runtimes
- C. When API invocations across multiple CloudHub workers must be load balanced
- D. When server-side load-balanced TLS mutual authentication is required between API implementations and API clients

Answer: D

Explanation:

Correct Answer

When server-side load-balanced TLS mutual authentication is required between API implementations and API clients

Fact/ Memory Tip: Although there are many benefits of CloudHub Dedicated Load balancer, TWO important things that should come to ones mind for considering it are:

>> Having URL endpoints with Custom DNS names on CloudHub deployed apps

>> Configuring custom certificates for both HTTPS and Two-way (Mutual) authentication. Coming to the options provided for this question:
>> We CANNOT use DLB to perform cross-region load balancing between separate deployments of the same Mule application.
>> We can have mapping rules to have more than one DLB URL pointing to same Mule app. But viceversa (More than one Mule app having same DLB URL) is NOT POSSIBLE
>> It is true that DLB helps to setup custom DNS names for Cloudhub deployed Mule apps but NOT true for apps deployed to Customer-hosted Mule Runtimes.
>> It is true to that we can load balance API invocations across multiple CloudHub workers using DLB but it is NOT A MUST. We can achieve the same (load balancing) using SLB (Shared Load Balancer) too. We DO NOT necessarily require DLB for achieve it.
So the only right option that fits the scenario and requires us to use DLB is when TLS mutual authentication is required between API implementations and API clients.

NEW QUESTION 7

Say, there is a legacy CRM system called CRM-Z which is offering below functions:

- * 1. Customer creation
- * 2. Amend details of an existing customer
- * 3. Retrieve details of a customer
- * 4. Suspend a customer

- A. Implement a system API named customerManagement which has all the functionalities wrapped in it asvarious operations/resources
- B. Implement different system APIs named createCustomer, amendCustomer, retrieveCustomer and suspendCustomer as they are modular and has seperation of concerns
- C. Implement different system APIs named createCustomerInCRMZ, amendCustomerInCRMZ, retrieveCustomerFromCRMZ and suspendCustomerInCRMZ as they are modular and has seperation of concerns

Answer: B

Explanation:

Correct Answer

Implement different system APIs named createCustomer, amendCustomer, retrieveCustomer and suspendCustomer as they are modular and has seperation of concerns

>> It is quite normal to have a single API and different Verb + Resource combinations. However, this fits well for an Experience API or a Process API but not a best architecture style for System APIs. So, option with just one customerManagement API is not the best choice here.
>> The option with APIs in createCustomerInCRMZ format is next close choice w.r.t modularization and less maintenance but the naming of APIs is directly coupled with the legacy system. A better foreseen approach would be to name your APIs by abstracting the backend system names as it allows seamless replacement/migration of any backend system anytime. So, this is not the correct choice too.
>> createCustomer, amendCustomer, retrieveCustomer and suspendCustomer is the right approach and is the best fit compared to other options as they are both modular and same time got the names decoupled from backend system and it has covered all requirements a System API needs.

NEW QUESTION 8

A code-centric API documentation environment should allow API consumers to investigate and execute API client source code that demonstrates invoking one or more APIs as part of representative scenarios.

What is the most effective way to provide this type of code-centric API documentation environment using Anypoint Platform?

- A. Enable mocking services for each of the relevant APIs and expose them via their Anypoint Exchange entry
- B. Ensure the APIs are well documented through their Anypoint Exchange entries and API Consoles and share these pages with all API consumers
- C. Create API Notebooks and include them in the relevant Anypoint Exchange entries
- D. Make relevant APIs discoverable via an Anypoint Exchange entry

Answer: C

Explanation:

Correct Answer

Create API Notebooks and Include them in the relevant Anypoint exchange entries

>> API Notebooks are the one on Anypoint Platform that enable us to provide code-centric API documentation

NEW QUESTION 9

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?

- 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.

NEW QUESTION 10

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 10

An API implementation is updated. When must the RAML definition of the API also be updated?

- A. When the API implementation changes the structure of the request or response messages
- B. When the API implementation changes from interacting with a legacy backend system deployed on-premises to a modern, cloud-based (SaaS) system
- C. When the API implementation is migrated from an older to a newer version of the Mule runtime
- D. When the API implementation is optimized to improve its average response time

Answer: A

Explanation:

Correct Answer

When the API implementation changes the structure of the request or response messages

>> RAML definition usually needs to be touched only when there are changes in the request/response schemas or in any traits on API.
>> It need not be modified for any internal changes in API implementation like performance tuning, backend system migrations etc..

NEW QUESTION 11

An API experiences a high rate of client requests (TPS) with small message payloads. How can usage limits be imposed on the API based on the type of client application?

- A. Use an SLA-based rate limiting policy and assign a client application to a matching SLA tier based on its type
- B. Use a spike control policy that limits the number of requests for each client application type
- C. Use a cross-origin resource sharing (CORS) policy to limit resource sharing between client applications, configured by the client application type
- D. Use a rate limiting policy and a client ID enforcement policy, each configured by the client application type

Answer: A

Explanation:

Correct Answer

Use an SLA-based rate limiting policy and assign a client application to a matching SLA tier based on its type.

>> SLA tiers will come into play whenever any limits to be imposed on APIs based on client type

NEW QUESTION 12

What Mule application deployment scenario requires using Anypoint Platform Private Cloud Edition or Anypoint Platform for Pivotal Cloud Foundry?

- A. When it is required to make ALL applications highly available across multiple data centers
- B. When it is required that ALL APIs are private and NOT exposed to the public cloud
- C. When regulatory requirements mandate on-premises processing of EVERY data item, including meta-data
- D. When ALL backend systems in the application network are deployed in the organization's intranet

Answer: C

Explanation:

Correct Answer

When regulatory requirements mandate on-premises processing of EVERY data item, including meta-data.

We need NOT require to use Anypoint Platform PCE or PCF for the below. So these options are OUT.

>> We can make ALL applications highly available across multiple data centers using CloudHub too.
>> We can use Anypoint VPN and tunneling from CloudHub to connect to ALL backend systems in the application network that are deployed in the organization's intranet.
>> We can use Anypoint VPC and Firewall Rules to make ALL APIs private and NOT exposed to the public cloud.
Only valid reason in the given options that requires to use Anypoint Platform PCE/ PCF is - When regulatory requirements mandate on-premises processing of EVERY data item, including meta-data.

NEW QUESTION 17

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?

- 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.

NEW QUESTION 21

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?

- A. Integration tests
- B. Performance tests
- C. Functional tests (Blackbox)
- D. Unit tests (Whitebox)

Answer: D

Explanation:

Correct Answer

Unit tests (Whitebox)

NEW QUESTION 25

Question 10: Skipped

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?

- 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.

NEW QUESTION 27

An Order API must be designed that contains significant amounts of integration logic and involves the invocation of the Product API. The power relationship between Order API and Product API is one of "Customer/Supplier", because the Product API is used heavily throughout the organization and is developed by a dedicated development team located in the office of the CTO. What strategy should be used to deal with the API data model of the Product API within the Order API?

- A. Convince the development team of the Product API to adopt the API data model of the Order API such that the integration logic of the Order API can work with one consistent internal data model
- B. Work with the API data types of the Product API directly when implementing the integration logic of the Order API such that the Order API uses the same (unchanged) data types as the Product API
- C. Implement an anti-corruption layer in the Order API that transforms the Product API data model into internal data types of the Order API
- D. Start an organization-wide data modeling initiative that will result in an Enterprise Data Model that will then be used in both the Product API and the Order API

Answer: C

Explanation:

Correct Answer

Convince the development team of the product API to adopt the API data model of the Order API such that integration logic of the Order API can work with one consistent internal data model

***** Key details to note from the given scenario:

>> Power relationship between Order API and Product API is customer/supplier

So, as per below rules of "Power Relationships", the caller (in this case Order API) would request for features to the called (Product API team) and the Product API team would need to accomodate those requests.

NEW QUESTION 28

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 31

What is a key performance indicator (KPI) that measures the success of a typical C4E that is immediately apparent in responses from the Anypoint Platform APIs?

- A. The number of production outage incidents reported in the last 24 hours
- B. The number of API implementations that have a publicly accessible HTTP endpoint and are being managed by Anypoint Platform
- C. The fraction of API implementations deployed manually relative to those deployed using a CI/CD tool
- D. The number of API specifications in RAML or OAS format published to Anypoint Exchange

Answer: D

Explanation:

Correct Answer

The number of API specifications in RAML or OAS format published to Anypoint Exchange

>> The success of C4E always depends on their contribution to the number of reusable assets that they have helped to build and publish to Anypoint Exchange.

>> It is NOT due to any factors w.r.t # of outages, Manual vs CI/CD deployments or Publicly accessible HTTP endpoints

>> Anypoint Platform APIs helps us to quickly run and get the number of published RAML/OAS assets to Anypoint Exchange. This clearly depicts how successful a C4E team is based on number of returned assets in the response.

NEW QUESTION 32

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?

- 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.

NEW QUESTION 36

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

APIs use HTTP-based protocols: cached HTTP responses from previous HTTP requests may potentially be returned if the same HTTP request is seen again.

Safe HTTP methods are ones that do not alter the state of the underlying resource. That is, the *HTTP responses to requests using safe HTTP methods may be cached*.

The HTTP standard requires the following HTTP methods on any resource to be safe:

- GET
- HEAD
- OPTIONS

Safety must be honored by REST APIs (but not by non-REST APIs like SOAP APIs): It is the *responsibility of every API implementation* to implement **GET, HEAD or OPTIONS** methods such that they never change the state of a resource.

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

NEW QUESTION 40

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 44

What is typically NOT a function of the APIs created within the framework called API-led connectivity?

- A. They provide an additional layer of resilience on top of the underlying backend system, thereby insulating clients from extended failure of these systems.
- B. They allow for innovation at the user Interface level by consuming the underlying assets without being aware of how data is being extracted from backend systems.
- C. They reduce the dependency on the underlying backend systems by helping unlock data from backend systems in a reusable and consumable way.
- D. They can compose data from various sources and combine them with orchestration logic to create higher level value.

Answer: A

Explanation:

Correct Answer

They provide an additional layer of resilience on top of the underlying backend system, thereby insulating clients from extended failure of these systems.

***** In API-led connectivity,

>> Experience APIs - allow for innovation at the user interface level by consuming the underlying assets without being aware of how data is being extracted from backend systems.

>> Process APIs - compose data from various sources and combine them with orchestration logic to create higher level value

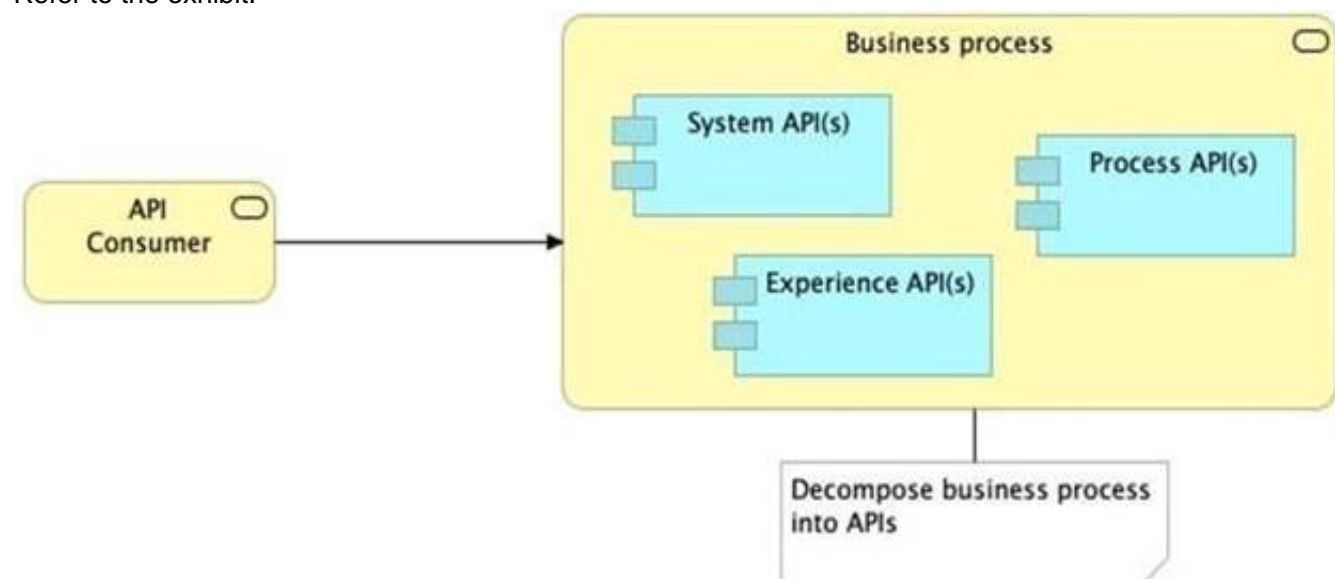
>> System APIs - reduce the dependency on the underlying backend systems by helping unlock data from backend systems in a reusable and consumable way.

However, they NEVER promise that they provide an additional layer of resilience on top of the underlying backend system, thereby insulating clients from extended failure of these systems.

<https://dzone.com/articles/api-led-connectivity-with-mule>

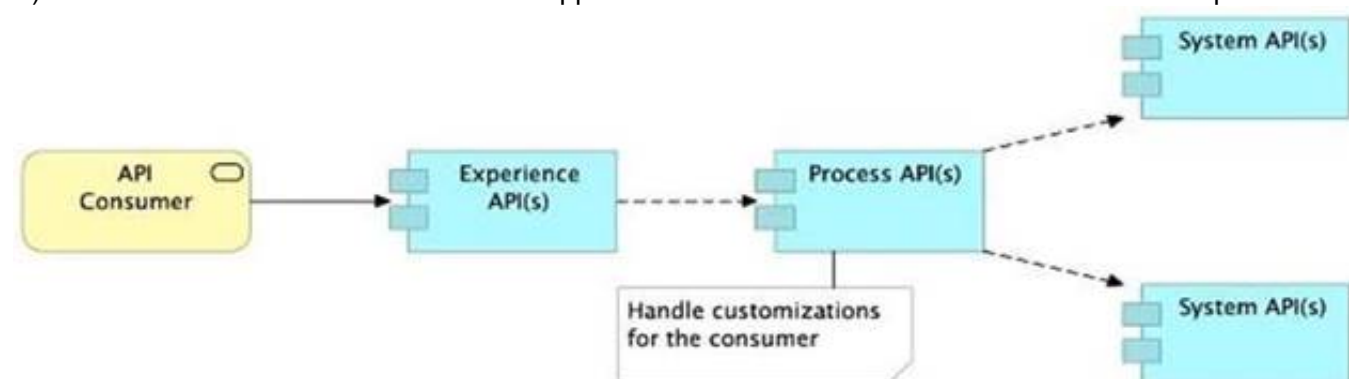
NEW QUESTION 46

Refer to the exhibit.

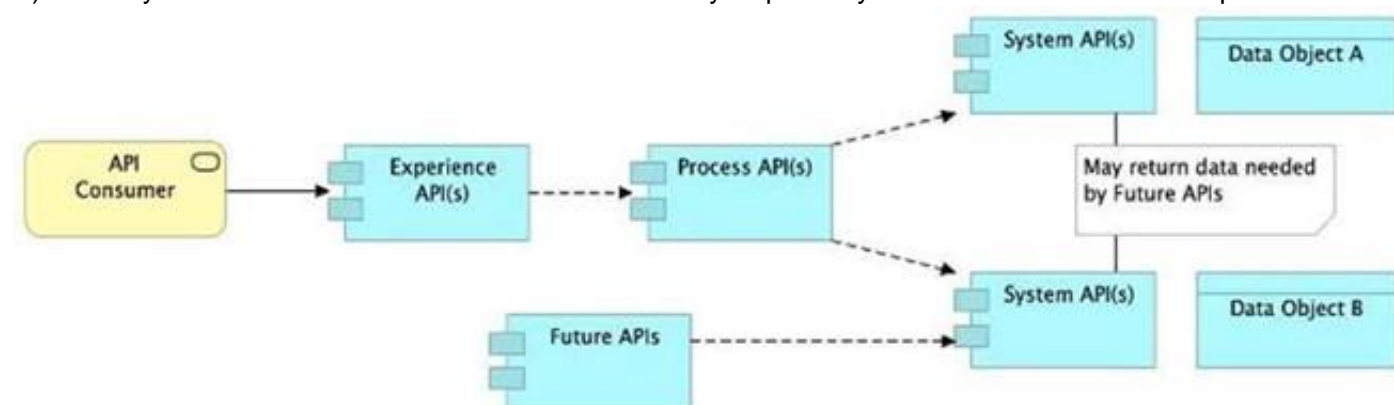


What is the best way to decompose one end-to-end business process into a collaboration of Experience, Process, and System APIs?

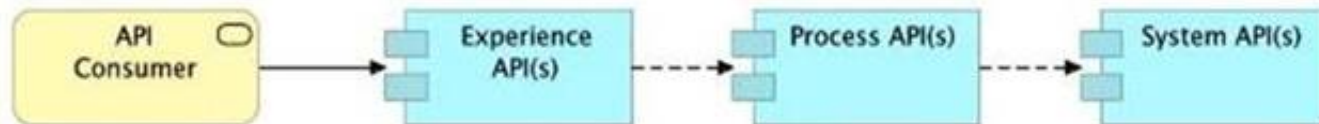
- A) Handle customizations for the end-user application at the Process API level rather than the Experience API level



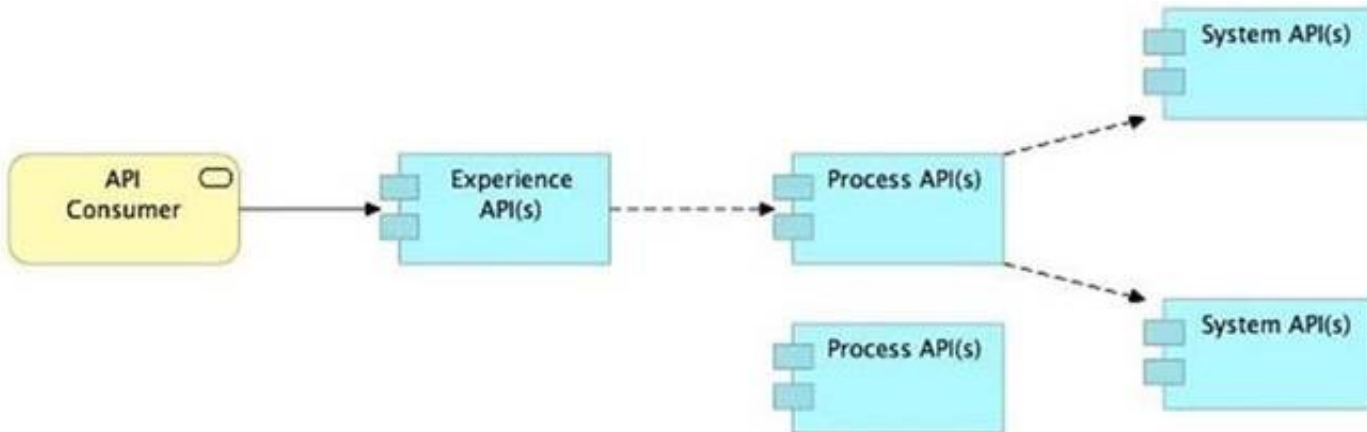
- B) Allow System APIs to return data that is NOT currently required by the identified Process or Experience APIs



C) Always use a tiered approach by creating exactly one API for each of the 3 layers (Experience, Process and System APIs)



D) Use a Process API to orchestrate calls to multiple System APIs, but NOT to other Process APIs



- A. Option A
- B. Option B
- C. Option C
- D. Option D

Answer: B

Explanation:

Correct Answer

Allow System APIs to return data that is NOT currently required by the identified Process or Experience APIs.

>> All customizations for the end-user application should be handled in "Experience API" only. Not in Process API

>> We should use tiered approach but NOT always by creating exactly one API for each of the 3 layers. Experience APIs might be one but Process APIs and System APIs are often more than one. System APIs for sure will be more than one all the time as they are the smallest modular APIs built in front of end systems.

>> Process APIs can call System APIs as well as other Process APIs. There is no such anti-design pattern in API-Led connectivity saying Process APIs should not call other Process APIs.

So, the right answer in the given set of options that makes sense as per API-Led connectivity principles is to allow System APIs to return data that is NOT currently required by the identified Process or Experience APIs. This way, some future Process APIs can make use of that data from System APIs and we need NOT touch the System layer APIs again and again.

NEW QUESTION 49

What is true about API implementations when dealing with legal regulations that require all data processing to be performed within a certain jurisdiction (such as in the USA or the EU)?

- A. They must avoid using the Object Store as it depends on services deployed ONLY to the US East region
- B. They must use a Jurisdiction-local external messaging system such as Active MQ rather than Anypoint MQ
- C. They must be deployed to Anypoint Platform runtime planes that are managed by Anypoint Platform control planes, with both planes in the same Jurisdiction
- D. They must ensure ALL data is encrypted both in transit and at rest

Answer: C

Explanation:

Correct Answer

They must be deployed to Anypoint Platform runtime planes that are managed by Anypoint Platform control planes, with both planes in the same Jurisdiction.

>> As per legal regulations, all data processing to be performed within a certain jurisdiction. Meaning, the data in USA should reside within USA and should not go out. Same way, the data in EU should reside within EU and should not go out.

>> So, just encrypting the data in transit and at rest does not help to be compliant with the rules. We need to make sure that data does not go out too.

>> The data that we are talking here is not just about the messages that are published to Anypoint MQ. It includes the apps running, transaction states, application logs, events, metric info and any other metadata. So, just replacing Anypoint MQ with a locally hosted ActiveMQ does NOT help.

>> The data that we are talking here is not just about the key/value pairs that are stored in Object Store. It includes the messages published, apps running, transaction states, application logs, events, metric info and any other metadata. So, just avoiding using Object Store does NOT help.

>> The only option left and also the right option in the given choices is to deploy application on runtime and control planes that are both within the jurisdiction.

NEW QUESTION 51

A System API is designed to retrieve data from a backend system that has scalability challenges. What API policy can best safeguard the backend system?

- A. IPwhitelist
- B. SLA-based rate limiting
- C. Auth 2 token enforcement
- D. Client ID enforcement

Answer: B

Explanation:

Correct Answer

SLA-based rate limiting

>> Client Id enforcement policy is a "Compliance" related NFR and does not help in maintaining the "Quality of Service (QoS)". It CANNOT and NOT meant for protecting the backend systems from scalability challenges.

>> IP Whitelisting and OAuth 2.0 token enforcement are "Security" related NFRs and again does not help in maintaining the "Quality of Service (QoS)". They CANNOT and are NOT meant for protecting the backend systems from scalability challenges.

Rate Limiting, Rate Limiting-SLA, Throttling, Spike Control are the policies that are "Quality of Service (QoS)" related NFRs and are meant to help in protecting the backend systems from getting overloaded.

<https://dzone.com/articles/how-to-secure-apis>

NEW QUESTION 56

An Anypoint Platform organization has been configured with an external identity provider (IdP) for identity management and client management. What credentials or token must be provided to Anypoint CLI to execute commands against the Anypoint Platform APIs?

- A. The credentials provided by the IdP for identity management
- B. The credentials provided by the IdP for client management
- C. An OAuth 2.0 token generated using the credentials provided by the IdP for client management
- D. An OAuth 2.0 token generated using the credentials provided by the IdP for identity management

Answer: A

Explanation:

Correct Answer

The credentials provided by the IdP for identity management

NEW QUESTION 59

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-the-box Anypoint Platform policy can address exposure to this threat?

- A. Shut out bad actors by using HTTPS mutual authentication for all API invocations
- B. Apply an IP blacklist policy to all APIs; the blacklist will include all bad actors
- C. Apply a Header injection and removal policy that detects the malicious data before it is used
- D. Apply a JSON threat protection policy to all APIs to detect potential threat vectors

Answer: D

Explanation:

Correct Answer

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

>> Usually, if the APIs are designed and developed for specific consumers (known consumers/customers) then we would IP Whitelist the same to ensure that traffic only comes from them.

>> However, as this scenario states that the APIs are publicly available and being used by so many mobile and web applications, it is NOT possible to identify and blacklist all possible bad actors.

>> So, JSON threat protection policy is the best chance to prevent any bad JSON payloads from such bad actors.

NEW QUESTION 63

An organization is deploying their new implementation of the OrderStatus System API to multiple workers in CloudHub. This API fronts the organization's on-premises Order Management System, which is accessed by the API implementation over an IPsec tunnel.

What type of error typically does NOT result in a service outage of the OrderStatus System API?

- A. A CloudHub worker fails with an out-of-memory exception
- B. API Manager has an extended outage during the initial deployment of the API implementation
- C. The AWS region goes offline with a major network failure to the relevant AWS data centers
- D. The Order Management System is Inaccessible due to a network outage in the organization's on-premises data center

Answer: A

Explanation:

Correct Answer

A CloudHub worker fails with an out-of-memory exception.

>> An AWS Region itself going down will definitely result in an outage as it does not matter how many workers are assigned to the Mule App as all of those in that region will go down. This is a complete downtime and outage.

>> Extended outage of API manager during initial deployment of API implementation will of course cause issues in proper application startup itself as the API Autodiscovery might fail or API policy templates and policies may not be downloaded to embed at the time of application startup etc... there are many reasons that could cause issues.

>> A network outage on-premises would of course cause the Order Management System not accessible and it does not matter how many workers are assigned to the app they all will fail and cause outage for sure.

The only option that does NOT result in a service outage is if a CloudHub worker fails with an out-of-memory exception. Even if a worker fails and goes down, there are still other workers to handle the requests and keep the API UP and Running. So, this is the right answer.

NEW QUESTION 66

An organization makes a strategic decision to move towards an IT operating model that emphasizes consumption of reusable IT assets using modern APIs (as defined by MuleSoft).

What best describes each modern API in relation to this new IT operating model?

- A. Each modern API has its own software development lifecycle, which reduces the need for documentation and automation
- B. Each modern API must be treated like a product and designed for a particular target audience (for instance, mobile app developers)
- C. Each modern API must be easy to consume, so should avoid complex authentication mechanisms such as SAML or JWT D
- D. Each modern API must be REST and HTTP based

Answer: B

Explanation:

Correct Answers

* 1. Each modern API must be treated like a product and designed for a particular target audience (for instance mobile app developers)

Bottom of Form Top of Form

NEW QUESTION 69

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 71

What is a typical result of using a fine-grained rather than a coarse-grained API deployment model to implement a given business process?

- A. A decrease in the number of connections within the application network supporting the business process
- B. A higher number of discoverable API-related assets in the application network
- C. A better response time for the end user as a result of the APIs being smaller in scope and complexity
- D. An overall tower usage of resources because each fine-grained API consumes less resources

Answer: B

Explanation:

Correct Answer

A higher number of discoverable API-related assets in the application network.

>> We do NOT get faster response times in fine-grained approach when compared to coarse-grained approach.

>> In fact, we get faster response times from a network having coarse-grained APIs compared to a network having fine-grained APIs model. The reasons are below.

Fine-grained approach:

* 1. will have more APIs compared to coarse-grained

* 2. So, more orchestration needs to be done to achieve a functionality in business process.

* 3. Which means, lots of API calls to be made. So, more connections will needs to be established. So, obviously more hops, more network i/o, more number of integration points compared to coarse-grained approach where fewer APIs with bulk functionality embedded in them.

* 4. That is why, because of all these extra hops and added latencies, fine-grained approach will have bit more response times compared to coarse-grained.

* 5. Not only added latencies and connections, there will be more resources used up in fine-grained approach due to more number of APIs.

That's why, fine-grained APIs are good in a way to expose more number of resuable assets in your network and make them discoverable. However, needs more maintenance, taking care of integration points, connections, resources with a little compromise w.r.t network hops and response times.

NEW QUESTION 72

What is true about where an API policy is defined in Anypoint Platform and how it is then applied to API instances?

- A. The API policy is defined in Runtime Manager as part of the API deployment to a Mule runtime, and then ONLY applied to the specific API Instance
- B. The API policy is defined in API Manager for a specific API Instance, and then ONLY applied to the specific API instance
- C. The API policy is defined in API Manager and then automatically applied to ALL API instances
- D. The API policy is defined in API Manager, and then applied to ALL API instances in the specified environment

Answer: B

Explanation:

Correct Answer

The API policy is defined in API Manager for a specific API instance, and then ONLY applied to the specific API instance.

>> Once our API specifications are ready and published to Exchange, we need to visit API Manager and register an API instance for each API.
>> API Manager is the place where management of API aspects takes place like addressing NFRs by enforcing policies on them.
>> We can create multiple instances for a same API and manage them differently for different purposes.
>> One instance can have a set of API policies applied and another instance of same API can have different set of policies applied for some other purpose.
>> These APIs and their instances are defined PER environment basis. So, one needs to manage them separately in each environment.
>> We can ensure that same configuration of API instances (SLAs, Policies etc..) gets promoted when promoting to higher environments using platform feature. But this is optional only. Still one can change them per environment basis if they have to.
>> Runtime Manager is the place to manage API Implementations and their Mule Runtimes but NOT APIs itself. Though API policies get executed in Mule Runtimes, We CANNOT enforce API policies in Runtime Manager. We would need to do that via API Manager only for a cherry-picked instance in an environment. So, based on these facts, right statement in the given choices is - "The API policy is defined in API Manager for a specific API instance, and then ONLY applied to the specific API instance".

NEW QUESTION 75

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 77

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?

- 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, comparatively, very less calls will be allowed into the DR environment API opposed to its primary environment. With this in mind, let's 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.

NEW QUESTION 80

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?

- 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.

NEW QUESTION 85

.....

Relate Links

100% Pass Your MCPA-Level-1 Exam with ExamBible Prep Materials

<https://www.exambible.com/MCPA-Level-1-exam/>

Contact us

We are proud of our high-quality customer service, which serves you around the clock 24/7.

Viste - <https://www.exambible.com/>