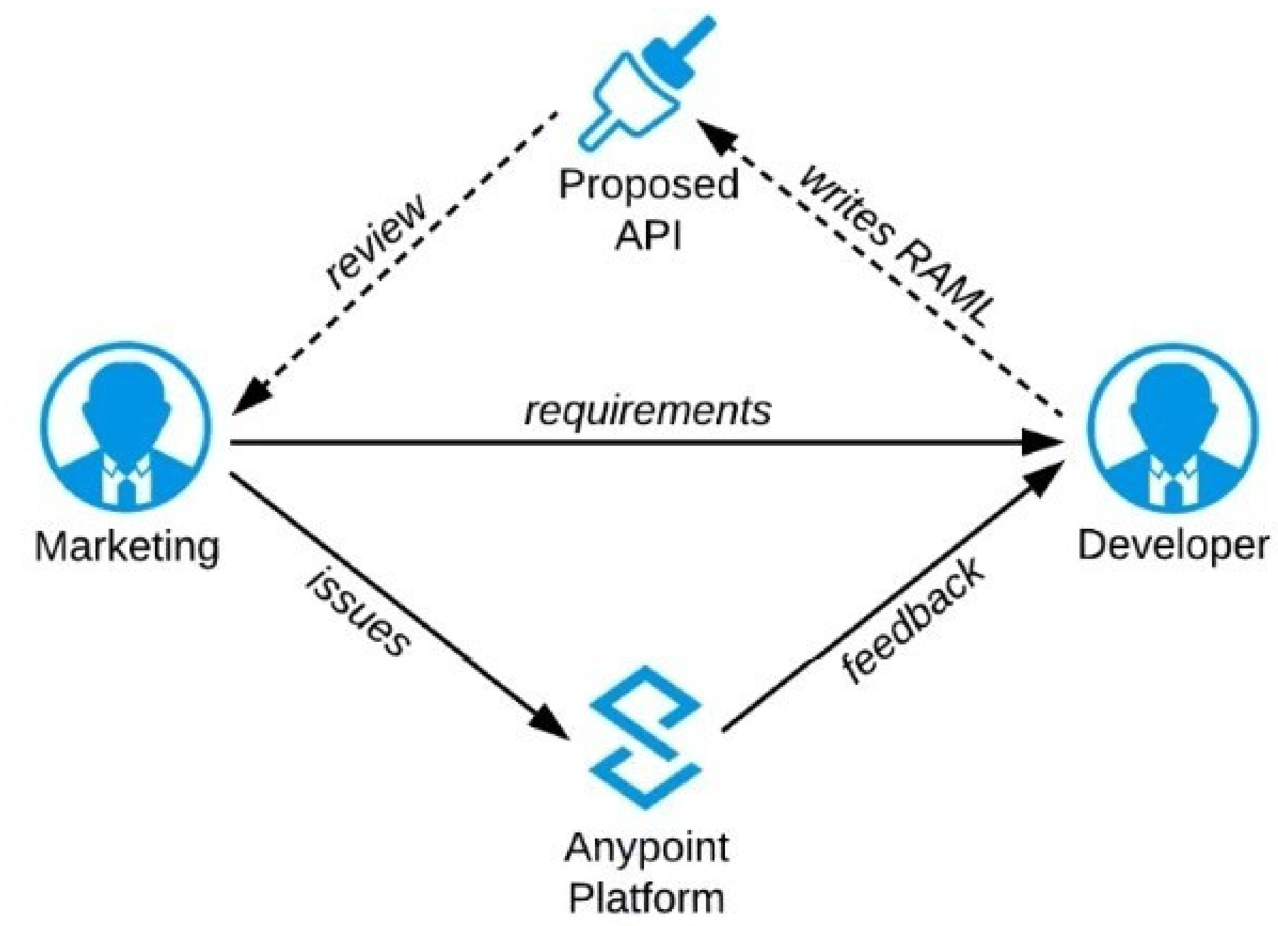
MCPA-LEVEL1 Q&As

MuleSoft Certified Platform Architect - Level 1

Pass Mulesoft MCPA-LEVEL1 Exam with 100% Guarantee

QUESTION 1

Refer to the exhibit.

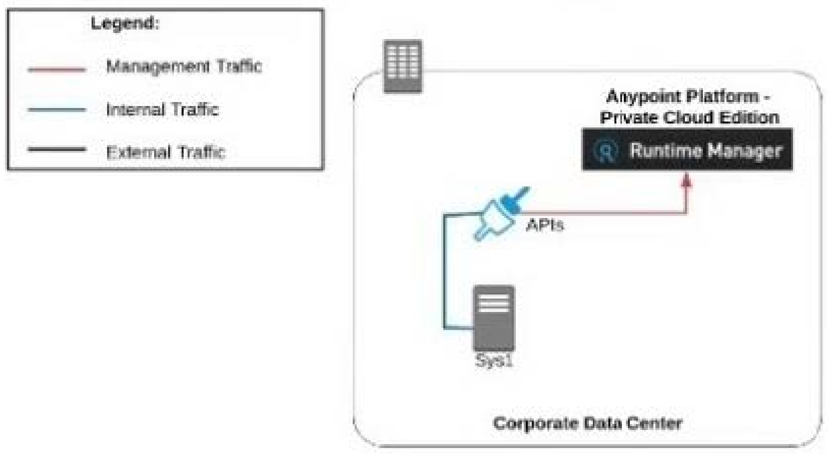


A RAML definition has been proposed for a new Promotions Process API, and has been published to Anypoint Exchange.

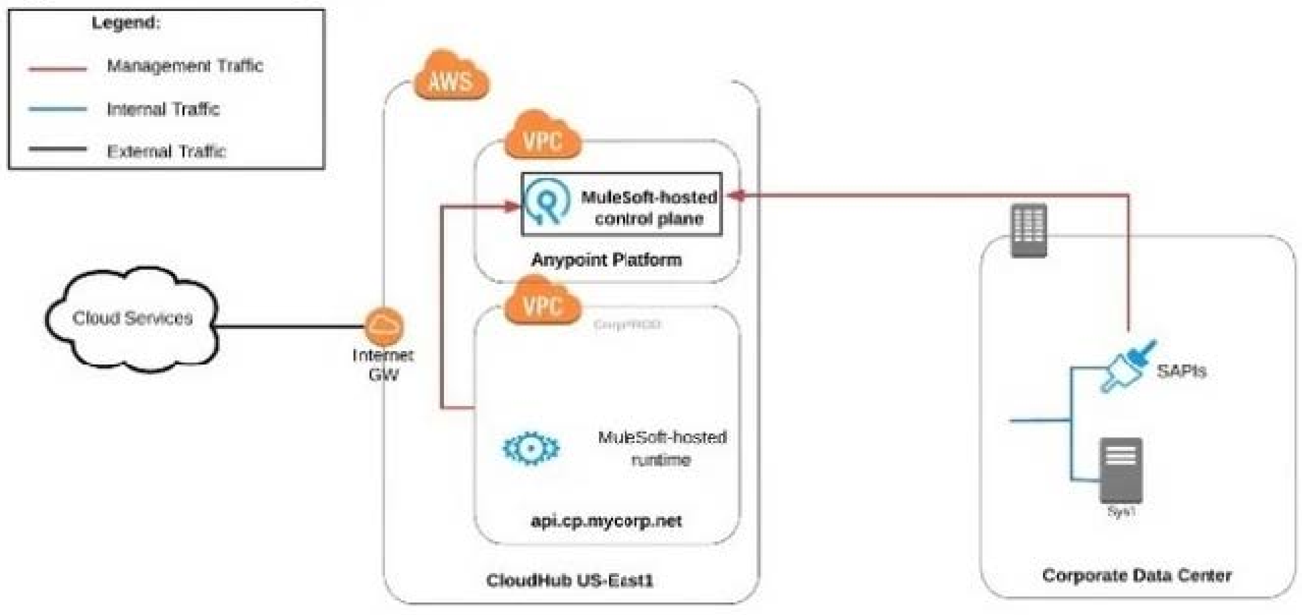
The Marketing Department, who will be an important consumer of the Promotions API, has important requirements and expectations that must be met.

What is the most effective way to use Anypoint Platform features to involve the Marketing Department in this early API design phase?

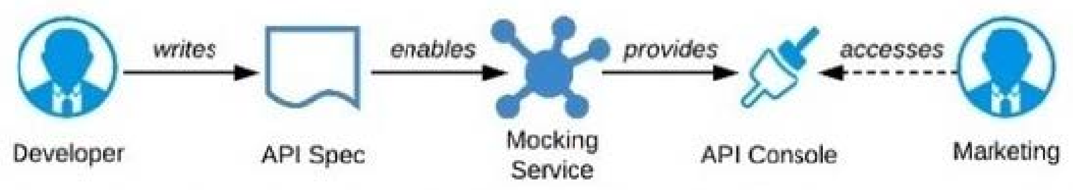
C. Use an on-premises installation of Mule runtimes that are completely isolated with NO external network access, managed by the Anypoint Platform Private Cloud Edition control plane



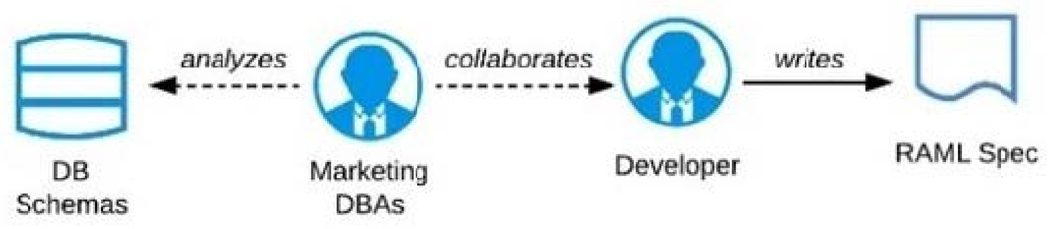
D. Use a combination of Cloud Hub-deployed and manually provisioned on-premises Mule runtimes managed by the MuleSoft-hosted Anypoint Platform control plane



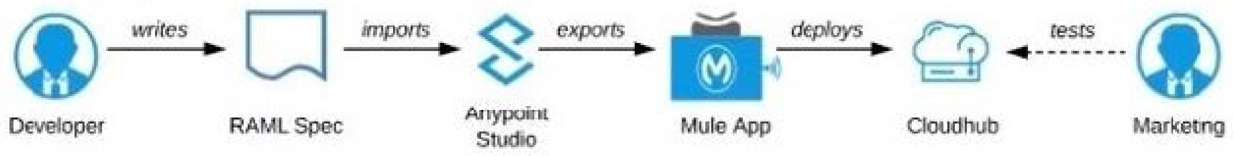
A. Ask the Marketing Department to interact with a mocking implementation of the API using the automatically generated API Console



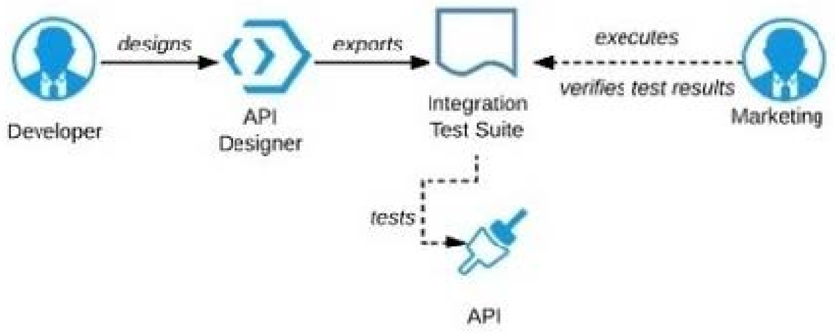
B. Organize a design workshop with the DBAs of the Marketing Department in which the database schema of the Marketing IT systems is translated into RAML



C. Use Anypoint Studio to implement the API as a Mule application, then deploy that API implementation to CloudHub and ask the Marketing Department to interact with it



D. Export an integration test suite from API designer and have the Marketing Department execute the tests In that suite to ensure they pass



A. Option A

B. Option B

C. Option C

D. Option D

Correct Answer: A

Ask the Marketing Department to interact with a mocking implementation of the API using the automatically generated API Console.

****

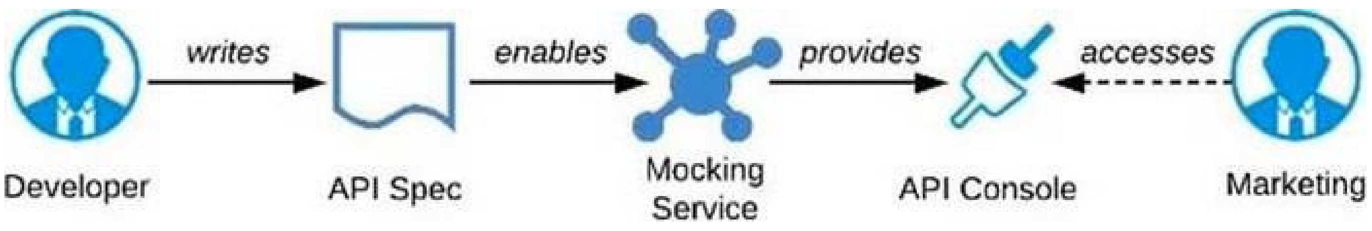
As per MuleSoft\\'s IT Operating Model:

>> API consumers need NOT wait until the full API implementation is ready. >> NO technical test-suites needs to be shared with end users to interact with APIs. >> Anypoint Platform offers a mocking capability on all the published API

specifications to Anypoint Exchange which also will be rich in documentation covering all details of API functionalities and working nature.

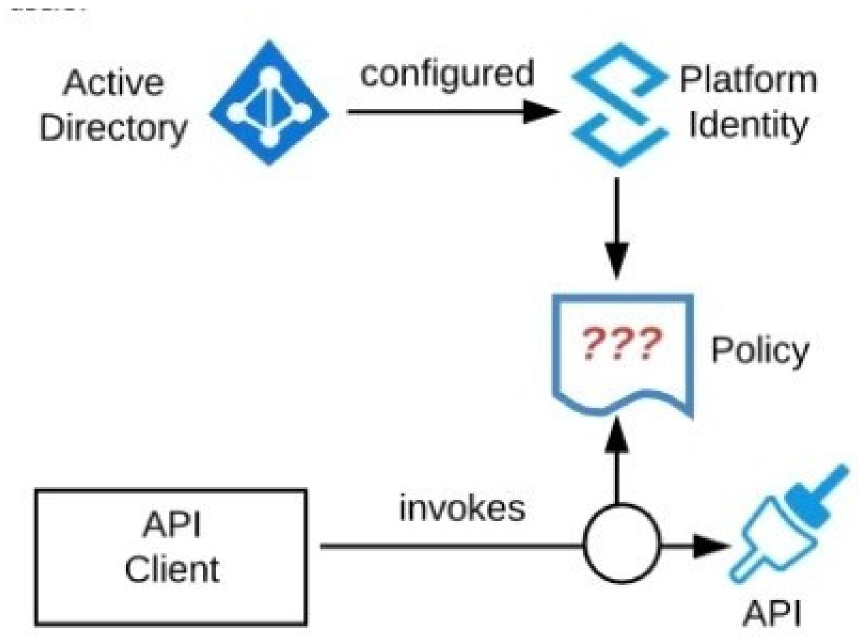
>> No needs of arranging days of workshops with end users for feedback.

API consumers can use Anypoint Exchange features on the platform and interact with the API using its mocking feature. The feedback can be shared quickly on the same to incorporate any changes.



QUESTION 2

Refer to the exhibit. An organization is running a Mule standalone runtime and has configured Active Directory as the Anypoint Platform external Identity Provider. The organization does not have budget for other system components.



What policy should be applied to all instances of APIs in the organization to most effecuvelyKestrict access to a specific group of internal users?

A. Apply a basic authentication - LDAP policy; the internal Active Directory will be configured as the LDAP source for authenticating users

B. Apply a client ID enforcement policy; the specific group of users will configure their client applications to use their specific client credentials

C. Apply an IP whitelist policy; only the specific users\\' workstations will be in the whitelist

D. Apply an OAuth 2.0 access token enforcement policy; the internal Active Directory will be configured as the OAuth server

Correct Answer: A

Apply a basic authentication - LDAP policy; the internal Active Directory will be configured as the LDAP source for authenticating users.

****

>> IP Whitelisting does NOT fit for this purpose. Moreover, the users workstations may not necessarily have static IPs in the network.

>> OAuth 2.0 enforcement requires a client provider which isn\\'t in the organizations system components.

>> It is not an effective approach to let every user create separate client credentials and configure those for their usage. The effective way it to apply a basic authentication - LDAP policy and the internal Active Directory will be configured as

the LDAP source for authenticating users. Reference: https://docs.mulesoft.com/api-manager/2.x/basic-authentication-ldap-concept

QUESTION 3

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

Correct Answer: 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
*** 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 4

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

Correct Answer: C

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.

QUESTION 5

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

Correct Answer: C

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: https://docs.mulesoft.com/exchange/to-use-api-notebook

---

QUESTION 6

Which of the following sequence is correct?

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

Correct Answer: B

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 7

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

Correct Answer: B

SLA-based rate limiting *** >> Client Id enforement 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

QUESTION 8

Mule applications that implement a number of REST APIs are deployed to their own subnet that is inaccessible from outside the organization.

External business-partners need to access these APIs, which are only allowed to be invoked from a separate subnet dedicated to partners - called Partner-subnet. This subnet is accessible from the public internet, which allows these external partners to reach it.

Anypoint Platform and Mule runtimes are already deployed in Partner-subnet. These Mule runtimes can already access the APIs.

What is the most resource-efficient solution to comply with these requirements, while having the least impact on other applications that are currently using the APIs?

A. Implement (or generate) an API proxy Mule application for each of the APIs, then deploy the API proxies to the Mule runtimes

B. Redeploy the API implementations to the same servers running the Mule runtimes

C. Add an additional endpoint to each API for partner-enablement consumption

D. Duplicate the APIs as Mule applications, then deploy them to the Mule runtimes

Correct Answer: A

QUESTION 9

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?

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

Correct Answer: C

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 10

In an organization, the InfoSec team is investigating Anypoint Platform related data traffic.

From where does most of the data available to Anypoint Platform for monitoring and alerting originate?

A. From the Mule runtime or the API implementation, depending on the deployment model

B. From various components of Anypoint Platform, such as the Shared Load Balancer, VPC, and Mule runtimes

C. From the Mule runtime or the API Manager, depending on the type of data

D. From the Mule runtime irrespective of the deployment model

Correct Answer: D

From the Mule runtime irrespective of the deployment model ***

>> Monitoring and Alerting metrics are always originated from Mule Runtimes irrespective of the deployment model.

>> It may seems that some metrics (Runtime Manager) are originated from Mule Runtime and some are (API Invocations/ API Analytics) from API Manager. However, this is realistically NOT TRUE. The reason is, API manager is just a

management tool for API instances but all policies upon applying on APIs eventually gets executed on Mule Runtimes only (Either Embedded or API Proxy). >> Similarly all API Implementations also run on Mule Runtimes. So, most of the day required for monitoring and alerts are originated fron Mule Runtimes only irrespective of whether the deployment model is MuleSoft-hosted or Customer- hosted or Hybrid.

QUESTION 11

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

Correct Answer: 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.

****

>> 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

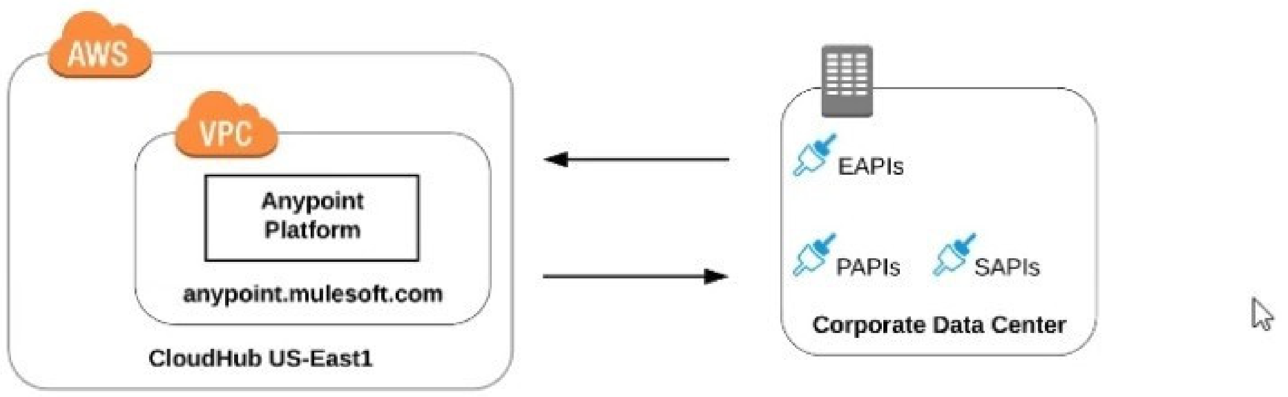
response upon retries but NOT in least possible time. So, this is also NOT a right choice of implementation for given requirement. >> One more option that is suggesting to invoke API in primary environment and API in DR environment in

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 12

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

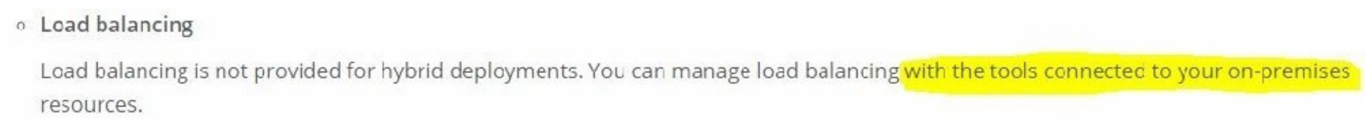
D. Anypoint Runtime Manager automatically ensures HA in the control plane by creating a new Mule runtime instance in case of a node failure

Correct Answer: C

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



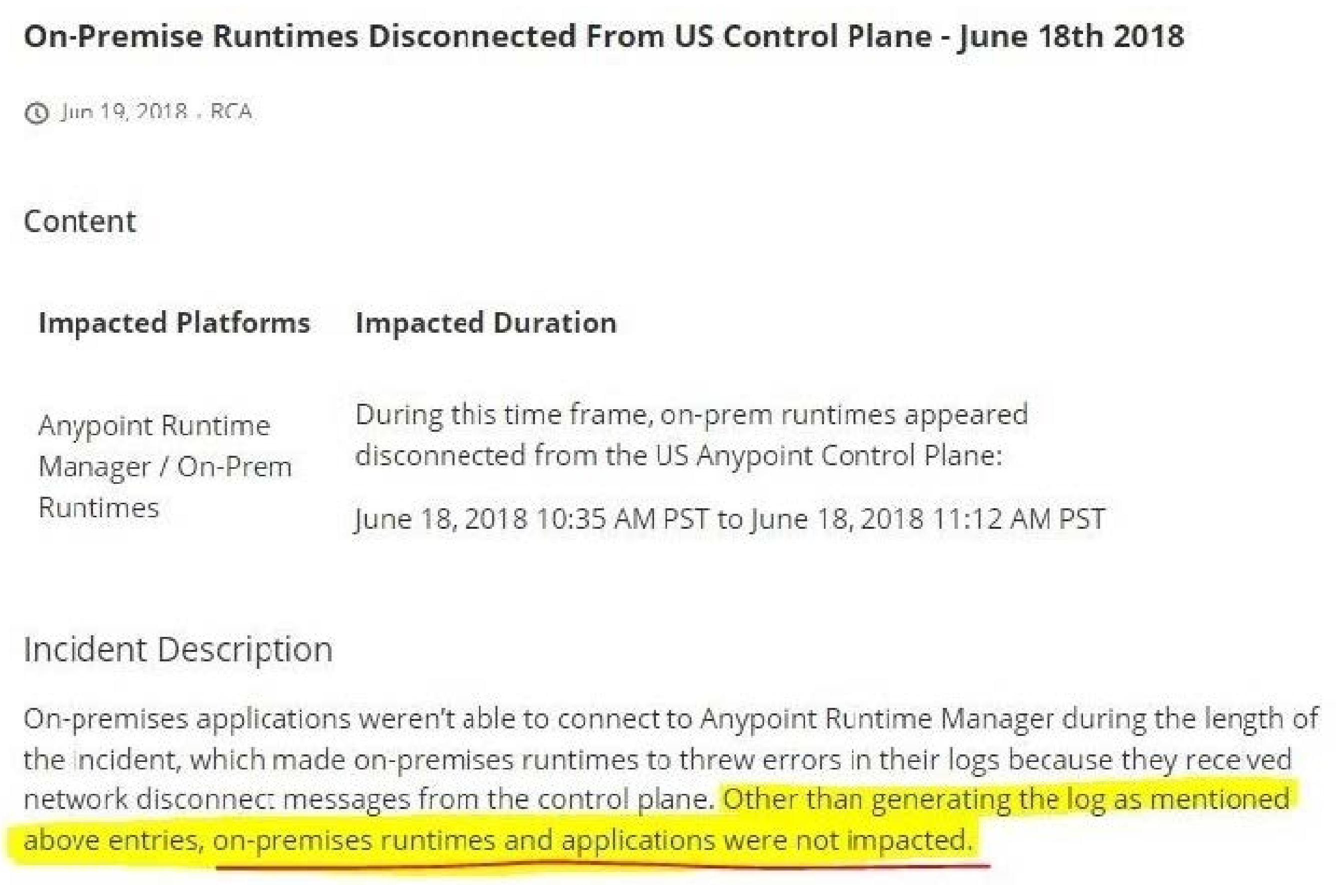
- Load balancing

  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-US-Control-Plane-June-25th-2019 https://help.mulesoft.com/s/article/On-premise-Runtimes-Appear-Disconnected-in-RuntimeManager-May-29th-2018



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

Jun 19, 2018 · RCA

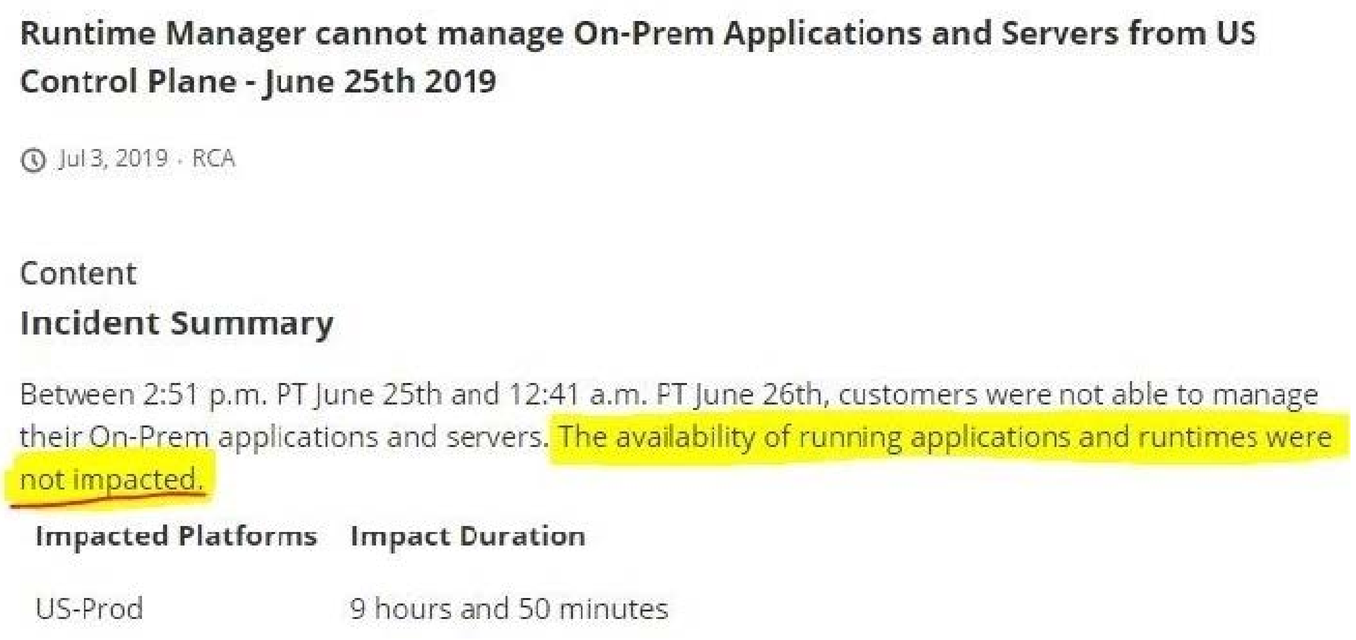
Content

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 threw 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

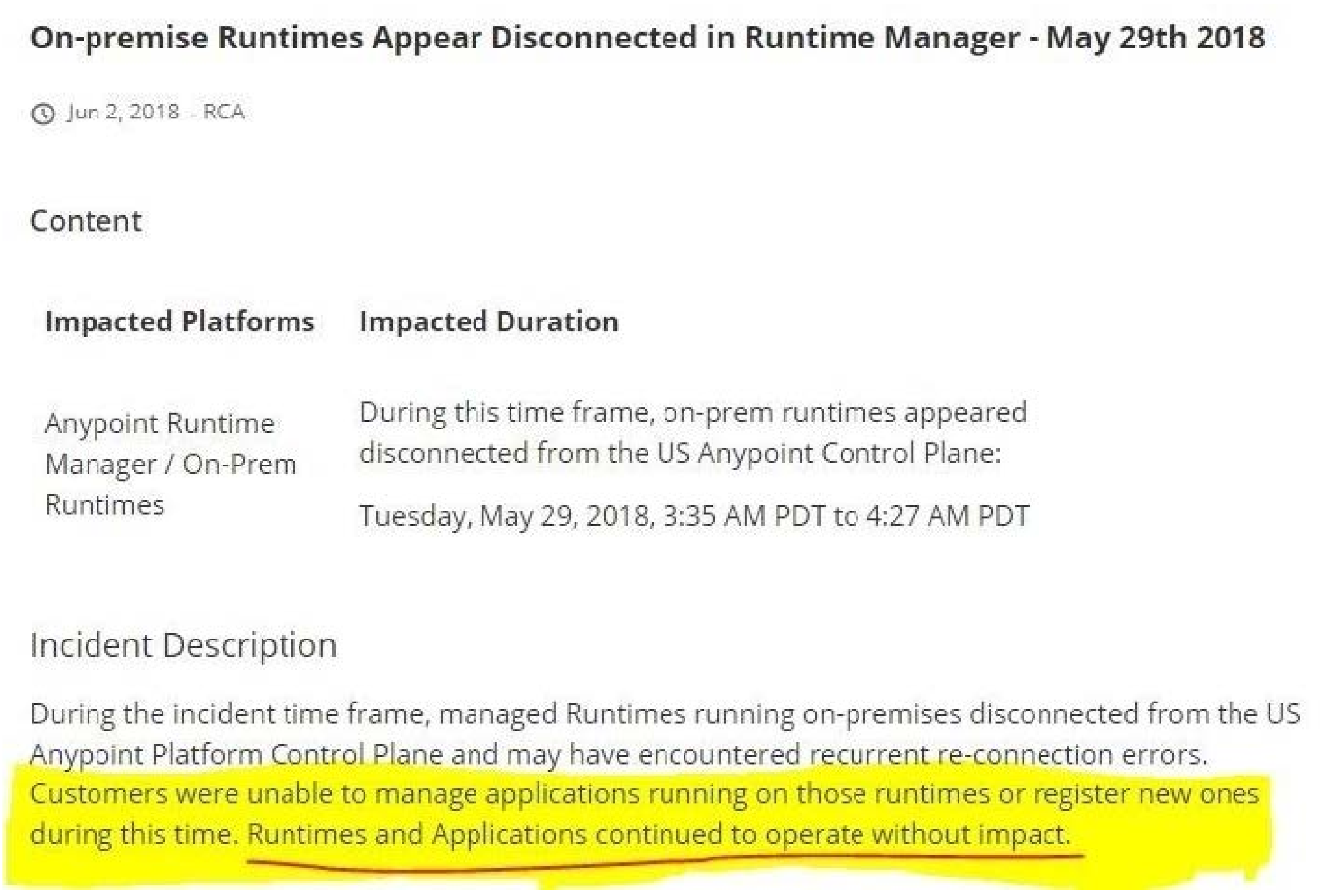
Content

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
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
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.

QUESTION 13

When could the API data model of a System API reasonably mimic the data model exposed by the corresponding backend system, with minimal improvements over the backend system\\'s data model?

A. When there is an existing Enterprise Data Model widely used across the organization

B. When the System API can be assigned to a bounded context with a corresponding data model

C. When a pragmatic approach with only limited isolation from the backend system is deemed appropriate

D. When the corresponding backend system is expected to be replaced in the near future

Correct Answer: C

When a pragmatic approach with only limited isolation from the backend system is deemed appropriate. *** General guidance w.r.t choosing Data Models: >> If an Enterprise Data Model is in use then the API data model of System APIs should make use of data types from that Enterprise Data Model and the corresponding API implementation should translate between these data types from the Enterprise Data Model and the native data model of the backend system. >> If no Enterprise Data Model is in use then each System API should be assigned to a Bounded Context, the API data model of System APIs should make use of data types from the corresponding Bounded Context Data Model and the corresponding API implementation should translate between these data types from the Bounded Context Data Model and the native data model of the backend system. In this scenario, the data types in the Bounded Context Data Model are defined purely in terms of their business characteristics and are typically not related to the native data model of the backend system. In other words, the translation effort may be significant. >> If no Enterprise Data Model is in use, and the definition of a clean Bounded Context Data Model is considered too much effort, then the API data model of System APIs should make use of data types that approximately mirror those from the backend system, same semantics and naming as backend system, lightly sanitized, expose all fields needed for the given System API\\'s functionality, but not significantly more and making good use of REST conventions. The latter approach, i.e., exposing in System APIs an API data model that basically mirrors that of the backend system, does not provide satisfactory isolation from backend systems through the System API tier on its own. In particular, it will typically not be possible to "swap out" a backend system without significantly changing all System APIs in front of that backend system and therefore the API implementations of all Process APIs that depend on those System APIs! This is so because it is not desirable to prolong the life of a previous backend system\\'s data model in the form of the API data model of System APIs that now front a new backend system. The API data models of System APIs following this approach must therefore change when the backend system is replaced. On the other hand: >> It is a very pragmatic approach that adds comparatively little overhead over accessing the backend system directly >> Isolates API clients from intricacies of the backend system outside the data model (protocol, authentication, connection pooling, network address, ...) >> Allows the usual API policies to be applied to System APIs >> Makes the API data model for interacting with the backend system explicit and visible, by exposing it in the RAML definitions of the System APIs >> Further isolation from the backend system data model does occur in the API implementations of the Process API tier

QUESTION 14

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

Correct Answer: D

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 vicevera (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. Reference: https://docs.mulesoft.com/runtime-manager/cloudhub-dedicated-load-balancer

QUESTION 15

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

A. Guarding against Denial of Service attacks

B. Maintaining tamper-proof credentials between APIs

C. Logging HTTP requests and responses

D. Backend system overloading

Correct Answer: A

Guarding against Denial of Service attacks ***

>> Backend system overloading can be handled by enforcing "Spike Control Policy" >> Logging HTTP requests and responses can be done by enforcing "Message Logging Policy" >> Credentials can be tamper-proofed using "Security" and

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

Reference: https://help.mulesoft.com/s/article/DDos-Dos-at