MCPA-LEVEL-1-MAINTENANCE Q&As

MuleSoft Certified Platform Architect - Level 1 MAINTENANCE

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

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

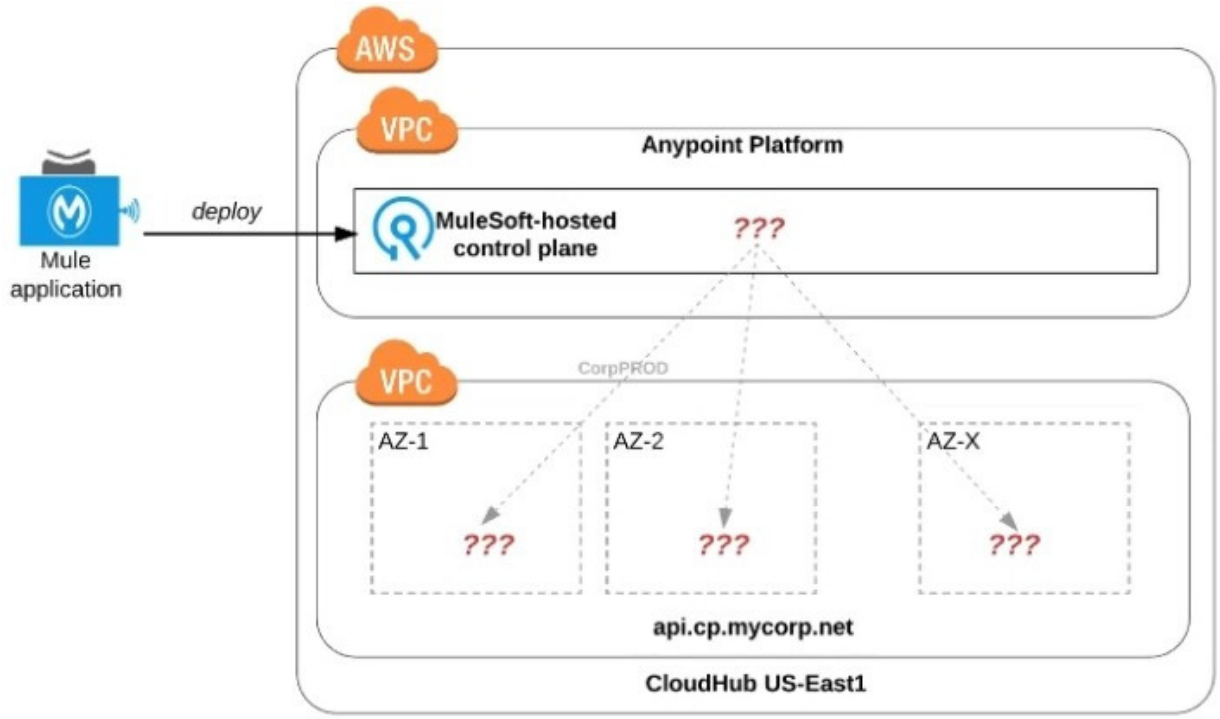
https://www.pass4itsure.com/mcpa-level-1-maintenance.html

100% Passing Guarantee
100% Money Back Assurance

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

QUESTION 1

Refer to the exhibit.



An organization uses one specific CloudHub (AWS) region for all CloudHub deployments.

How are CloudHub workers assigned to availability zones (AZs) when the organization\\'s Mule applications are deployed to CloudHub in that region?

A. Workers belonging to a given environment are assigned to the same AZ within that region

B. AZs are selected as part of the Mule application\\'s deployment configuration

C. Workers are randomly distributed across available AZs within that region

D. An AZ is randomly selected for a Mule application, and all the Mule application\\'s CloudHub workers are assigned to that one AZ

Correct Answer: D

Workers are randomly distributed across available AZs within that region.

****

>> Currently, we only have control to choose which AWS Region to choose but there is no control at all using any configurations or deployment options to decide what Availability Zone (AZ) to assign to what worker.

>> There are NO fixed or implicit rules on platform too w.r.t assignment of AZ to workers based on environment or application.

>> They are completely assigned in random. However, cloudhub definitely ensures that HA is achieved by assigning the workers to more than on AZ so that all workers are not assigned to same AZ for same application. : https://help.mulesoft.com/s/question/0D52T000051rqDj/one-cloudhub-aws-region-how- cloudhub-workers-are-assigned-to-availability-zones-azs-

QUESTION 2

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

Correct Answer: 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 ********** >> Canonical models are not an option here as the organization has already put in efforts and created bounded-context models for Experience and Process APIs. >> Anti-corruption layers for ALL APIs is unnecessary and invalid because it is mentioned that experience and process APIs share same bounded-context model. It is just the System layer APIs that need to choose their approach now. >> So, having an anti-corruption layer just between the process and system layers will work well. Also to speed up the approach, system APIs can mimic the backend system data model.

QUESTION 3

An API experiences a high rate of client requests (TPS) vwth small message paytoads. 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

Correct Answer: A

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

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

QUESTION 4

What correctly characterizes unit tests of Mule applications?

A. They test the validity of input and output of source and target systems

B. They must be run in a unit testing environment with dedicated Mule runtimes for the environment

C. They must be triggered by an external client tool or event source

D. They are typically written using MUnit to run in an embedded Mule runtime that does not require external connectivity

Correct Answer: D

They are typically written using MUnit to run in an embedded Mule runtime that does not require external connectivity.

****

Below TWO are characteristics of Integration Tests but NOT unit tests:

>> They test the validity of input and output of source and target systems. >> They must be triggered by an external client tool or event source. It is NOT TRUE that Unit Tests must be run in a unit testing environment with dedicated Mule

runtimes for the environment.

MuleSoft offers MUnit for writing Unit Tests and they run in an embedded Mule Runtime without needing any separate/ dedicated Runtimes to execute them. They also do NOT need any external connectivity as MUnit supports mocking via

stubs.

https://dzone.com/articles/munit-framework

QUESTION 5

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