Salesforce.Mule-Arch-201.v2026-06-13.q56

試験コード:	Mule-Arch-201
試験名称:	Salesforce Certified MuleSoft Platform Architect
認定資格:	Salesforce
無料問題数:	56
バージョン:	v2026-06-13
アクセス数:	109
ページビュー数:	560
https://www.jpnpdf.com/Salesforce.Mule-Arch-201.v2026-06-13.q56-mondaishu.html	

最新問題: **1**
あるオンラインストアのマーケティングチームは、オンラインショッピングカートをチェックアウトせずにそのまま放置する顧客が増加していることに気づきました。彼らは、カートが放置される根本的な原因は技術的な問題にあると疑っています。そこで、問題を特定するための支援を求めて、Center for Enablementに連絡しました。このショッピングアプリケーションには、アプリケーションネットワークの全レイヤーにわたる複数のAPIが利用されています。
関連するすべての API からのメトリクスを同時に表示するために使用できる Anypoint Platform の機能はどれですか?
A. カスタムダッシュボード
B. 組み込みダッシュボード
C. 機能監視
D. API マネージャー
Answer: B (メーッセージを残す**)**
クロス API モニタリングの必要性を理解する:
Center for Enablement (C4E) は、顧客がカートを放棄する原因となっている可能性のある、アプリケーション ネットワーク内の複数の API にわたる潜在的な技術的問題を調査する必要があります。
これには、複数の API にわたるメトリックをリアルタイムで表示して、パフォーマンスの問題やボトルネックを特定できるソリューションが必要です。
Anypoint プラットフォームの機能の評価:
組み込みダッシュボード :Anypoint Platform は、Anypoint Monitoring に組み込みダッシュボードを提供しており、複数の API のメトリクスを単一のインターフェースで確認できます。この機能は、API のパフォーマンス、レイテンシ、エラー、スループットを監視するように設計されており、アプリケーションネットワークのあらゆるレイヤーにわたるパフォーマンスの追跡に最適です。
カスタム ダッシュボード: カスタム ダッシュボードではよりカスタマイズされたビューが可能になりますが、組み込みダッシュボードではすでに複数の API のメトリックが集約されているため、このシナリオ用にカスタム ソリューションを構築する必要はありません。
機能モニタリング: この機能は、特定の API 機能と稼働時間を監視するためのテストを設定するために使用されますが、複数の API にわたるメトリックをリアルタイムで追跡するには適していません。

API マネージャー: API マネージャーは、アプリケーション ネットワーク全体にわたる詳細なリアルタイム メトリックを提供するのではなく、API ポリシー、契約、およびアクセス制御の管理に主に重点を置いています。
結論：
オプション B (組み込みダッシュボード) は、関連するすべての API からのメトリックの包括的なビューを提供し、ショッピングカートの放棄につながる可能性のある問題を C4E チームが迅速に特定できるため、最適な選択肢です。
これらのダッシュボードを効果的に構成して使用する方法の詳細については、MuleSoft の Anypoint Monitoring および組み込みダッシュボードに関するドキュメントを参照してください。

最新問題**: 2**
質問10: スキップ
API実装は、リクエスト元のAPIクライアントに3つのX-RateLimit-* HTTPレスポンスヘッダーを返します。これらのレスポンスヘッダーは、APIクライアントにどのような情報を示すのでしょうか？
A. スロットリングによって発生するエラーコード
B. 次のリクエストで送信される相関ID
C. HTTPレスポンスサイズ
D. API実装で許可された残りの容量
Answer: (解答を表示する**)**
正解: API 実装によって許可される残りの容量。
****

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

最新問題**: 3**
すべてのデータ処理を特定の管轄区域（米国や EU など）内で実行することを要求する法的規制に対処する場合の API 実装に関して正しいのは何ですか。
A. オブジェクトストアは米国東部地域にのみデプロイされたサービスに依存しているため、使用を避ける必要があります。
B. Anypoint MQではなく、Active MQなどの管轄地域固有の外部メッセージングシステムを使用する必要があります。
C. これらは、Anypoint Platform コントロールプレーンによって管理される Anypoint Platform ランタイムプレーンにデプロイされ、両方のプレーンが同じ管轄区域内にある必要があります。
D. すべてのデータが転送中も保存中も暗号化されていることを確認する必要があります
Answer: C (メーッセージを残す**)**
正解: これらは、Anypoint Platform コントロール プレーンによって管理される Anypoint Platform ランタイム プレーンにデプロイする必要があり、両方のプレーンは同じ管轄区域内にある必要があります。
****

>> 法規制により、すべてのデータ処理は特定の管轄区域内で行われる必要があります。つまり、米国のデータは米国内に保存され、米国外に持ち出されるべきではありません。同様に、EUのデータはEU内に保存され、EU外に持ち出されるべきではありません。
>> つまり、転送中や保存中のデータを暗号化するだけでは、規則遵守には役立ちません。データが外部に流出しないようにすることも必要です。
>> ここで言及しているデータは、Anypoint MQにパブリッシュされるメッセージだけではありません。実行中のアプリケーション、トランザクションの状態、アプリケーションログ、イベント、メトリック情報、その他のメタデータも含まれます。したがって、Anypoint MQをローカルでホストされているActiveMQに置き換えるだけでは役に立ちません。

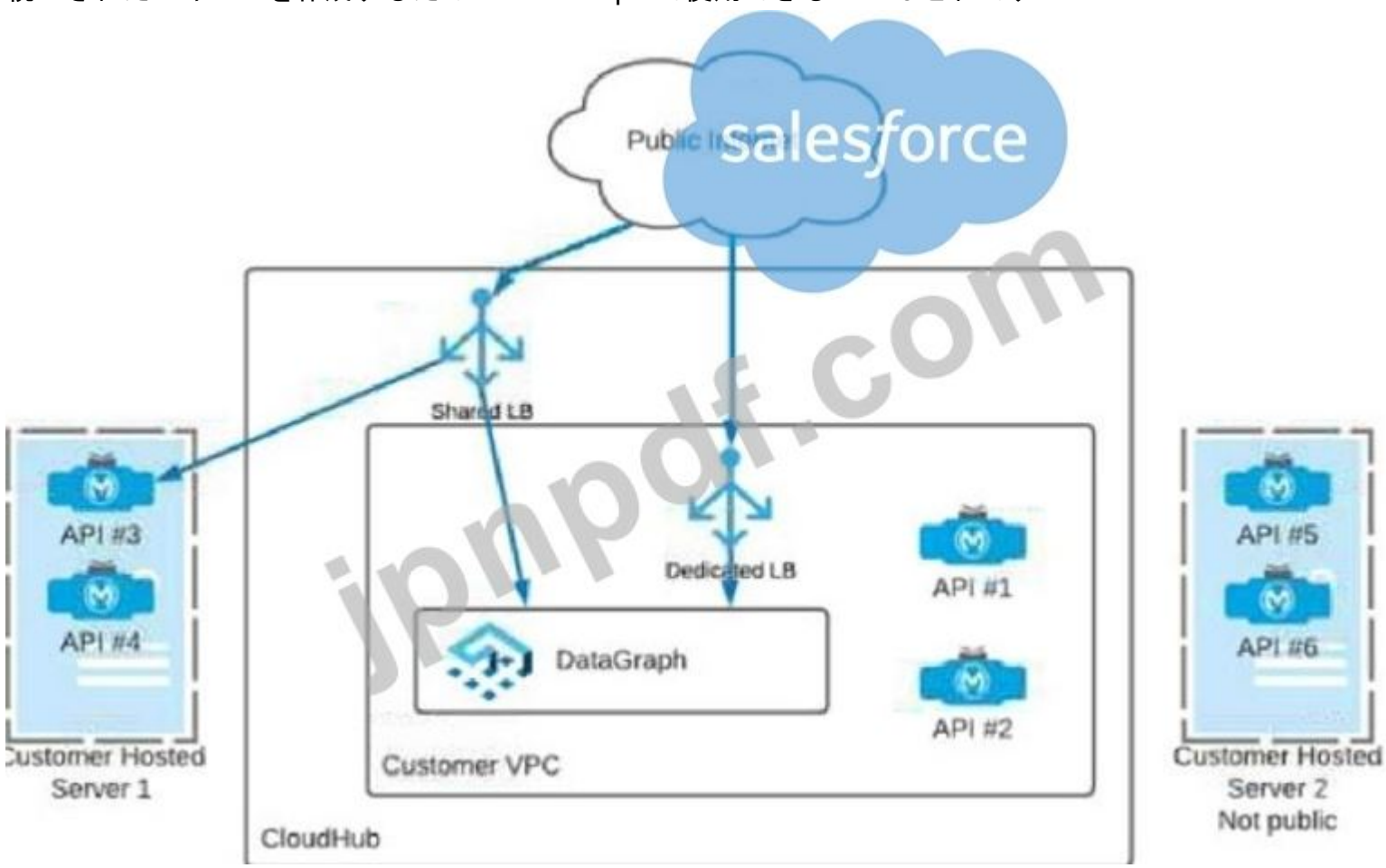
>> ここで言及しているデータは、オブジェクトストアに保存されているキー/バリューペアだけではありません。発行されたメッセージ、実行中のアプリケーション、トランザクションの状態、アプリケーションログ、イベント、メトリック情報、その他のメタデータも含まれます。したがって、オブジェクトストアの使用を避けるだけでは役に立ちません。
>> 与えられた選択肢の中で唯一残された、そして正しい選択肢は、管轄区域内にあるランタイム プレーンとコントロール プレーンにアプリケーションをデプロイすることです。

最新問題**: 4**
新しい上流APIは、中央値500ミリ秒、最大値800ミリ秒（99パーセンタイル）の応答時間というSLAを提供するように設計されています。対応するAPI実装では、非常に類似した複雑さを持つ3つの下流APIを順番に呼び出す必要があります。
これらのダウンストリーム API の最初のものは、応答時間について次の SLA を提供します: 中央値: 100 ミリ秒、80 パーセンタイル: 500 ミリ秒、95 パーセンタイル: 1000 ミリ秒。
可能であれば、新しいアップストリーム API の望ましい SLA を満たすために、最初のダウンストリーム API の呼び出しに対してアップストリーム API でタイムアウトを設定するにはどうすればよいでしょうか。
A. タイムアウトを 50 ミリ秒に設定します。これにより、その API の呼び出しがさらにタイムアウトしますが、再試行の余地がさらに増えます。
B. タイムアウトを 100 ミリ秒に設定します。これにより、他の 2 つの下流 API が完了するまでに 400 ミリ秒の余裕が生まれます。
C. 上流APIの希望するSLAを満たすためのタイムアウトは不可能です。最初の下流APIと異なるSLAを交渉するか、代替APIを呼び出す必要があります。
D. タイムアウトを設定しないでください。このAPIの呼び出しは必須なので、応答するまで待つ必要があります。
Answer: B (メーッセージを残す**)**
正解: タイムアウトを 100 ミリ秒に設定します。これにより、他の 2 つの下流 API が完了するまでに 400 ミリ秒の時間が残ります。
****
与えられたシナリオから得られる重要な詳細:
>> アップストリームAPIの設計SLAは500ミリ秒（中央値）です。SLAの最大応答時間は無視します。
>> この API は 3 つのダウンストリーム API を順番に呼び出しますが、これらはすべて同様の複雑さです。
>> 最初のダウンストリーム API は、平均 SLA 100 ミリ秒、80 パーセンタイル: 500 ミリ秒、95 パーセンタイル: 1000 ミリ秒を提供します。
上記の詳細に基づいて:
>> 50msのタイムアウトを設定するという選択肢は除外できます。なぜなら、提供されるSLAの中央値が100msの場合、ほとんどの呼び出しがタイムアウトになり、再試行に時間が浪費され、最終的にはすべての再試行で限界に達してしまうからです。たとえ一部の再試行が成功したとしても、残りの時間では、下流の2番目と3番目のAPIが時間内に応答するのに十分な余裕がありません。
>> このAPIの呼び出しは必須なので、応答するまで待つ必要があるため、タイムアウトを設定しないという選択肢は不適切です。タイムアウトを設定しないことは、適切な実装パターンに反するだけでなく、最初のAPIが提示されている平均SLAの100ミリ秒以内に応答しない場合、おそらく500ミリ秒（80パーセンタイル）または1000ミリ秒（95パーセンタイル）以内に応答するでしょう。どちらの場合も、最初の下流APIから正常な応答が得られても意味がありません。なぜなら、この時点で既に上流APIのSLAである500ミリ秒を超えているからです。2番目と3番目の下流APIを呼び出す時間はありません。
>> アップストリーム API の望ましい SLA を満たすためにタイムアウトが不可能であるというのは真実ではありません。

最初のダウンストリームAPIのSLA中央値は100ミリ秒であるため、ほとんどの場合、その時間内に応答が返されます。したがって、ほとんどの呼び出しではタイムアウトを100ミリ秒に設定することで、残りの2つのダウンストリームAPI呼び出しに400ミリ秒の余裕が確保されるため、理想的です。

最新問題: **5**
Mule アプリケーションを実装するために REST API が設計されています。
REST API を定義するために使用できる標準インターフェース定義言語は何ですか?
A. AsyncAPI仕様
B. OpenAPI 仕様 (OAS)
C. ヤム
D. Webサービス定義言語(WSDL)
Answer: B (メーッセージを残す**)**

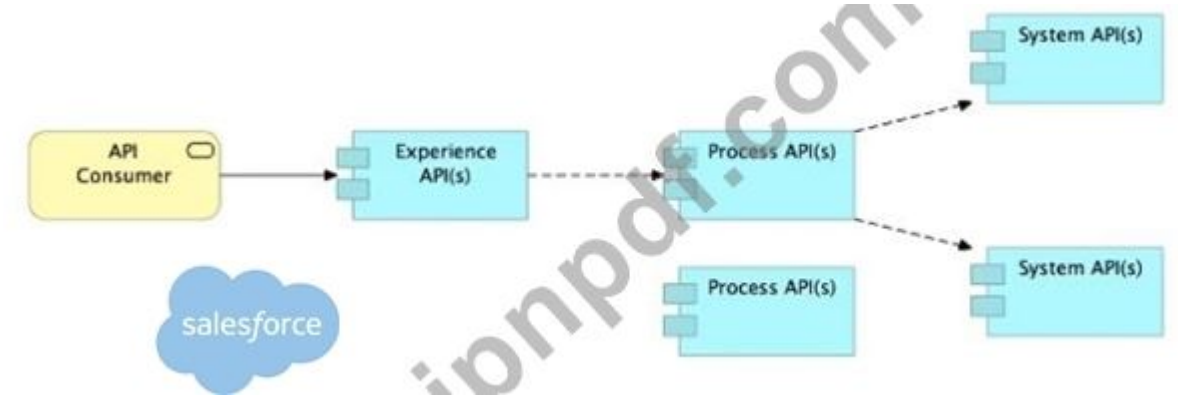
最新問題: **6**
統一されたスキーマを作成するために DataGraph で使用できる API はどれですか?



A. API 1、3、5
B. API 2、4、6
C. API 1、2、s5、6
D. API 1、2、3、4
Answer: (解答を表示する)

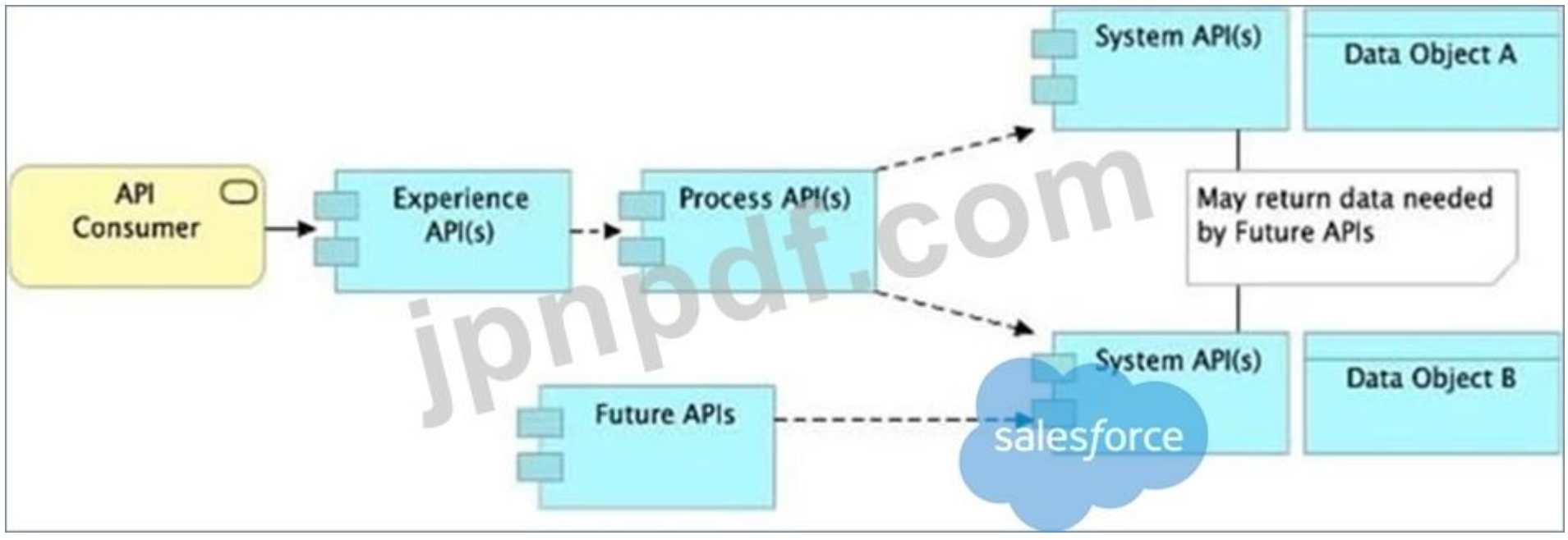
MuleSoft の DataGraph で統一されたスキーマを作成するには、API を DataGraph がこれらの API からデータを取得し、ユーザーがアクセスできる単一のスキーマに統合できるように公開する必要があります。DataGraph は、複数の API を統合して単一の統合 API エンドポイントを形成するフェデレーションアプローチを提供します。
この設定では次のようになります。
API 1、2、3、4 は、CloudHub 上のお客様の VPC 内でホストされており、共有ロードバランサー (LB) または専用ロードバランサー (DLB) を介してアクセスできるため、DataGraph に適しています。これらのロードバランサーはどちらもパブリックアクセスを提供しており、これは DataGraph がデータを集約するために API にアクセスする必要があるため、必須条件です。
API 5および6は、明示的に 非公開」とマークされているカスタマーホストサーバー2でホストされています。DataGraphは、APIを統合スキーマに集約するために、パブリックにアクセス可能なエンドポイントを介したAPIアクセスを必要とするため、この構成ではAPI 5および6をDataGraphで使用することはできません。
顧客ホスト サーバー 1 上の API 3 と 4 は共有 LB を介してアクセス可能であるように見えます。これは、DataGraph の要件を満たすパブリック アクセス可能性を意味します。
DataGraph 内で API 1、2、3、4 を組み合わせることで、クライアントが単一のソースからのデータであるかのように、これらすべての API からシームレスにデータをクエリできる統合スキーマを作成できます。
この設定により、効率的なデータ取得が可能になり、複数の API を個別に呼び出す必要性が減るため API の使用が簡素化され、パフォーマンスと開発者のエクスペリエンスが最適化されます。
参照
DataGraph での統合スキーマの設定と管理の詳細については、スキーマの集約と API フェデレーションに関する詳細なガイドラインが記載されている DataWeave のドキュメントと MuleSoft DataGraph リソースを参照してください。

最新問題: **7**
展示品を参照してください。
1 つのエンドツーエンドのビジネス プロセスをエクスペリエンス、プロセス、およびシステム API のコラボレーションに分解する最適な方法は何ですか?
A) エンドユーザー アプリケーションのカスタマイズをエクスペリエンス API レベルではなくプロセス API レベルで処理する B) 特定のプロセス API またはエクスペリエンス API で現在必要とされていないデータをシステム API が返すことを許可する C) 3 つのレイヤー (エクスペリエンス API、プロセス API、システム API) ごとに 1 つの API を作成することにより、常に階層型アプローチを使用する D) プロセス API を使用して複数のシステム API への呼び出しを調整し、他のプロセス API への呼び出しは調整しない



A. オプションA
B. オプションB
C. オプションC
D. オプションD

Answer: B (メーッセージを残す)
正解: 特定されたプロセス API またはエクスペリエンス API で現在必要とされていないデータをシステム API が返すことを許可します。
****
>> エンドユーザーアプリケーションのすべてのカスタマイズは、「エクスペリエンスAPI」のみで処理する必要があります。プロセスAPIでは処理できません。
>> 階層型アプローチを採用する必要がありますが、必ずしも3つのレイヤーそれぞれに1つのAPIを作成する必要はありません。エクスペリエンスAPIは1つで済むかもしれませんが、プロセスAPIとシステムAPIは複数になることが多いです。システムAPIは、エンドシステムの前面に構築される最小のモジュール型APIであるため、常に複数になるはずです。
>> プロセスAPIは、他のプロセスAPIと同様にシステムAPIを呼び出すことができます。API主導の接続には、プロセスAPIが他のプロセスAPIを呼び出すべきではないという反設計パターンは存在しません。
したがって、API主導の接続原則に則り、与えられた選択肢の中で理にかなった正しい答えは、システムAPIが、特定のプロセスAPIまたはエクスペリエンスAPIで現在必要とされていないデータを返せるようにすることです。こうすることで、将来のプロセスAPIはシステムAPIからそのデータを利用できるようになり、システム層APIに何度もアクセスする必要がなくなります。



最新問題: **8**
スケジュールされた間隔で API とエンドポイントを監視し、テストの合格か不合格かのレポートを受信し、API とエンドポイントのパフォーマンスに関する統計を表示するコンポーネントはどれですか。
A. API 分析
B. Anypoint Monitoring ダッシュボード
C. APT機能監視
D. Anypoint Runtime Manager アラート
Answer: C (メーッセージを残す)
API 機能モニタリングについて理解する:
API 機能モニタリングは、MuleSoft の Anypoint プラットフォーム内の機能であり、ユーザーはスケジュールされた間隔で機能テストを実行して、API とエンドポイントの健全性とパフォーマンスを監視できます。

テスト呼び出しを実行し、レスポンスが期待通りの条件を満たしているかどうかを評価することで、APIが期待通りに機能しているかどうかを確認します。これは、エンドポイントの可用性テスト、レスポンス内の特定データの確認、APIパフォーマンスの経時的な測定に特に役立ちます。
コンポーネントの特徴:
スケジュールされた間隔: 機能監視では、監視要件に応じて、毎分、毎時間、毎日などの定期的な間隔でテストを実行するように構成できます。
テストの合格/不合格ステータスのレポート: 各テストの実行後、API 機能モニタリングは、API がテスト条件に合格したか不合格になったかを報告します。
パフォーマンス統計: 平均応答時間、成功率、エラー率などの指標が表示され、API の健全性とパフォーマンスに関する洞察が得られます。
オプションの評価:
オプション A (API 分析): API 分析では、API の使用状況とメトリックに関する分析情報が提供されますが、合格/不合格ステータスやエンドポイントのヘルス チェックのスケジュールされたテストは含まれません。
オプション B (Anypoint モニタリング ダッシュボード): これらのダッシュボードには API メトリクスが表示されますが、API エンドポイントをアクティブにテストしたり、スケジュールに基づいて合格/不合格のレポートを提供したりはしません。
オプション C (正解): API 機能モニタリングは、スケジュールされたテスト実行で API とエンドポイントの健全性を監視し、パフォーマンスに関する統計を表示するように設計されているため、説明に適合します。
オプション D (Anypoint Runtime Manager アラート): Runtime Manager アラートは、アプリケーション ステータスに関する問題をユーザーに通知しますが、スケジュールされた間隔でエンドポイントをアクティブにテストすることはありません。
結論：
オプション C (API 機能モニタリング) は、API 機能をテストし、エンドポイントの健全性を監視し、パフォーマンス統計をリアルタイムで表示するために必要なツールを提供するため、正解です。
Anypoint Platform でこれらのテストをセットアップおよび構成する方法の詳細については、API 機能モニタリングに関する MuleSoft のドキュメントを参照してください。

最新問題: **9**
顧客は、MuleSoft アプリケーションを CloudHub 1.0 でホストしたいと考えています。これらのアプリケーションは、ドメイン https://api.acmecorp.com で利用できる必要があります。
acme-dib-prod という専用ロードバランサー (DLB) を作成した後、構成を完了するために顧客がさらに実行する必要があるアクションは何ですか?
A. api.acmecorp.com の TLS 証明書を使用して DLB を構成し、DLB に関連付けられたパブリック IP アドレスに api.acmecorp.com の A レコードを作成します。
B. api.acmecorp.com の TLS 証明書を使用して DLB を構成し、api.acmecorp.com から acme-dib-prod.|lb.anypointdns.net への CNAME レコードを作成します。
C. acme-dib-prod.Jb.anypointdns.net の TLS 証明書を使用して DLB を構成し、api.acmecorp:com から acme-dlb-prod.lb.anypointdns.net への CNAME レコードを作成します。
D. aplacmecorp.com の TLS 証明書を使用して DLB を構成し、api.aomecorp.com から acme-dib-prod.ei.cloubhub.io への CNAME レコードを作成します。
Answer: B (メーッセージを残す**)**

専用ロードバランサ (DLB) を使用して CloudHub 1.0 でホストされている MuleSoft アプリケーションのカスタム ドメインを設定する場合は、次の手順に従います。
TLS証明書の設定 :カスタムドメインapi.acmecorp.comをカバーするTLS証明書を使用してDLB（acme-dib-prod）を設定します。この証明書により、HTTPSトラフィックがDLB経由で安全にMuleアプリケーションに転送されるようになります。
CNAME を使用した DNS 構成:
api.acmecorp.com を DLB ホスト名 acme-dib-prod.lb.anypointdns.net にポイントする CNAME レコードを作成します。
CNAMEレコードにより、カスタムドメインはMuleSoftのAnypoint Platformが提供するDLBに解決されます。このCNAMEマッピングにより、すべてのトラフィックが適切なDLBに転送され、処理と負荷分散が行われます。
オプションBが正しい理由:
CNAME レコードは、DLB 用に Anypoint Platform によって管理されるエンドポイントである acme-dib-prod.lb.anypointdns.net に必要なエイリアスを提供します。
オプション B では、DLB の内部ホスト名ではなく、api.acmecorp.com 専用の TLS 証明書を使用して DLB を構成する必要があることも正しく識別されます。
誤った選択肢:
DLBの内部ホスト名にTLS証明書を使用するか、Aレコードを使用するようにDLBを構成することを推奨するオプションは、このシナリオには適していません。MuleSoft CloudHub 1.0 DLBはCNAMEレコードを使用して柔軟かつスケーラブルなドメイン管理を提供しますが、これらのロードバランサーでは直接IP（Aレコード）はサポートされていません。
参照
CloudHub 1.0 でカスタム ドメインと DLB を構成する方法の詳細については、DLB のセットアップと DNS 構成に関する MuleSoft のドキュメントを参照してください。

最新問題: **10**
ある企業は、Mule API実装をできるだけ早く本番環境に移行したいと考えています。Muleアプリケーションのすべてのデータとメタデータへのアクセスを保護するため、すべてのMuleアプリケーションを社内ファイアウォール内の顧客ホスト型インフラストラクチャにデプロイする必要があります。これらのプロジェクトライフサイクルの目標を満たすには、ランタイムプレーンとコントロールプレーンのオプションをどのように組み合わせればよいでしょうか？
A. 手動でプロビジョニングされた顧客ホスト型ランタイムプレーンと顧客ホスト型コントロールプレーン
B. MuleSoft がホストするランタイム プレーンと顧客がホストするコントロール プレーン
C. 手動でプロビジョニングされた顧客ホストのランタイムプレーンとMuleSoftホストのコントロールプレーン
D. iPaaS プロビジョニングされた顧客ホストのランタイム プレーンと MuleSoft ホストのコントロール プレーン
Answer: (解答を表示する**)**
正解: 手動でプロビジョニングされた顧客ホスト型ランタイムプレーンと顧客ホスト型コントロールプレーン
****
質問に示されているシナリオから考慮すべき重要な要素が 2 つあります。
>> 企業では、データとメタデータの両方を企業のファイアウォール内に保存することを要求しています
>> 当社は顧客がホストするインフラストラクチャを採用したいと考えています。
クラウドを直接的または間接的に扱うデプロイメント モデル (Mulesoft がホストするもの、または Azure、AWS などの顧客独自のクラウド) では、少なくともメタデータを共有する必要があります。

アプリケーションデータは、顧客がホストするランタイムプレーン上にMule Runtimesを配置することで、ファイアウォール内で制御できます。しかし、Mulsoftがホストするクラウドベースのコントロールプレーンを使用する場合、コントロールプレーンは少なくとも最低限のメタデータを企業のファイアウォール外に送信する必要がありました。
顧客要件は、データとメタデータの両方が企業のファイアウォール内にあることを明確にしており、顧客はできるだけ早く本番環境に移行したいと考えていても、残念ながらセキュリティ要件の性質上、手動でプロビジョニングされた顧客ホスト型ランタイム プレーンと顧客ホスト型コントロール プレーンを使用するしか選択肢がありません。

最新問題: **11**
API プロデューサーは、Anypoint Exchange において、セマンティックバージョニングの慣例に従い、API バージョン 3.1.1 から 3.2.0 に API を更新しました。この変更は API 公開ポータルを通じて通知されています。新しいバージョンでは API エンドポイントは変更されません。API クライアントの開発者はこの変更にどのように対応すべきでしょうか？
A. APIクライアントコードは、新しい機能を利用する必要がある場合にのみ変更する必要があります。
B. APIクライアントは自身のコードを更新し、完全な回帰テストを実行する必要があります。
C. APIプロデューサーは、新しいバージョンと並行して古いバージョンを実行するように要求される必要があります。
D. 既存の機能の変更を理解するために、APIプロデューサーに連絡する必要があります。
Answer: A (メーッセージを残す**)**

最新問題: **12**
大量の統合ロジックが含まれ、製品 API の呼び出しが含まれる注文 API を設計する必要があります。
製品 API は組織全体で頻繁に使用され、CTO のオフィスにある専用の開発チームによって開発されているため、注文 API と製品 API 間の力関係は 顧客/サプライヤー」の関係となります。
Order API 内で Product API の API データ モデルを処理するにはどのような戦略を使用する必要がありますか?
A. 製品 API の開発チームに、注文 API の API データ モデルを採用するように説得し、注文 API の統合ロジックが 1 つの一貫した内部データ モデルで動作できるようにします。
B. 注文APIの統合ロジックを実装するときに、製品APIのAPIデータ型を直接操作して、注文APIが製品APIと同じ（変更されていない）データ型を使用するようにします。
C. Order API に、Product API データ モデルを Order API の内部データ型に変換する破損防止レイヤーを実装します。
D. 組織全体のデータモデリングイニシアチブを開始し、製品APIと注文APIの両方で使用されるエンタープライズデータモデルを作成します。
Answer: C (メーッセージを残す**)**
正解: 製品 API の開発チームに、注文 API の API データ モデルを採用するように説得し、注文 API の統合ロジックが 1 つの一貫した内部データ モデルで動作できるようにします。
****
与えられたシナリオから注目すべき重要な詳細:
>> 注文 API と製品 API 間の力関係は顧客/サプライヤーです。したがって、以下の 力関係」のルールに従って、呼び出し元 (この場合は注文 API) は呼び出された側 (製品 API チーム) に機能を要求し、製品 API チームはそれらの要求に対応する必要があります。

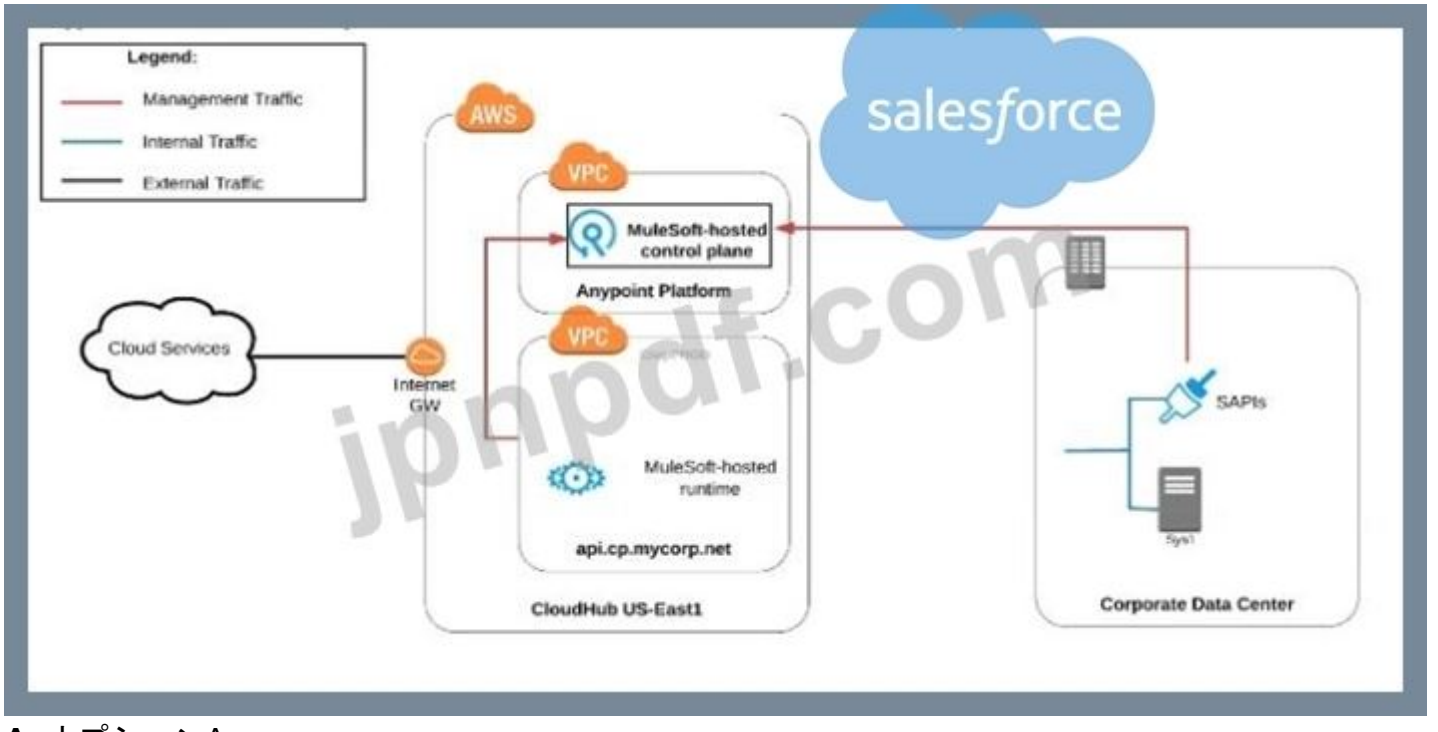
最新問題: **13**

経験豊富な中央IT部門を持つ大規模組織が、MuleSoftの導入を開始しています。サイロ化されたバックエンドシステムを新しい顧客関係管理（CRM）システムに接続するプロジェクトが進行中です。Enablementセンターは、API主導の接続性を活用するよう指導しています。
API 主導の接続性を使用したアプリケーション ネットワークの作成をサポートするアクションは何ですか?
A. ビジネスアナリストに、2つのシステム間の標準データモデルを指定するためのビジネスプロセスモデルの作成を依頼します。
B. 新しい CRM システムがカスタム REST API の作成をサポートし、CloudHub で 4 つのプライベート ネットワークを確立し、GAuth 2.0 認証をサポートしているかどうかを判断します。
C. このプロジェクトを迅速に進めるために、中央IT部門はCRMシステムとバックエンドシステムを拡張し、組み込みの統合インターフェースを使用して相互に接続する必要があります。
D. REST APIを使用してバックエンドシステムのデータをロック解除するためのシステムAPIを作成します。
Answer: D (メーッセージを残す)
API 主導の接続性から始めて、サイロ化されたバックエンド システムを新しい CRM と統合する組織の場合、次のアプローチはベスト プラクティスと MuleSoft の Center for Enablement (C4E) ガイダンスと一致します。
API 主導の接続性: このモデルは、API を個別のレイヤー (システム、プロセス、エクスペリエンス) に編成し、再利用性、モジュール性、管理性を向上させます。
システム API は、コア システム (バックエンド アプリケーションやデータベースなど) からデータを公開およびロック解除するために使用されます。
プロセス API は、複数のシステム間でデータを調整し、必要に応じて変換します。
エクスペリエンス API は、この場合の CRM などのアプリケーションやデバイスで使用するためにデータを特別にフォーマットします。
アプリケーションネットワークをサポートするための手順:
バックエンドシステム用のシステムAPIを作成します。このAPIは、CRMとの統合をサポートするために必要なデータを公開する必要があります。
RESTful インターフェースを備えたシステム API を作成することで、標準化された方法でデータにアクセスできるようになり、他のシステムとの統合が容易になり、将来のスケーラビリティがサポートされます。
オプションDが正しい理由:
システムAPIを作成することは、API主導の接続性の原則に沿っており、データが再利用可能な方法で公開されることを保証します。このAPIは、CRMの要件を満たすために必要に応じてプロセスAPIによってオーケストレーションされ、他のアプリケーションにも容易に拡張できます。
誤った選択肢:
オプション A (ビジネス プロセス モデルの作成) では、接続が直接有効になったり、API を通じてバックエンド データが公開されたりすることはありません。
オプション B はこの段階では不要です。OAuth 2.0 サポートなどの CRM 機能の評価は、システム API を介したアプリケーション ネットワークの作成とは直接関係ありません。
オプション C は、柔軟性と拡張性の欠如により API 主導の接続が回避しようとするポイントツーポイントの統合を提案することで、API 主導のベスト プラクティスと矛盾します。
参照
システム、プロセス、エクスペリエンス API を使用してスケーラブルな統合レイヤーを構築するための詳細なフレームワークについては、MuleSoft の API 主導の接続リソースを参照してください。

最新問題**: 14**
アプリケーションは、在庫の一貫性を保つために、常に1つのプロセスのみを実行して在庫を更新します。このプロセスの実行には200ミリ秒 (0.2秒)かかります。したがって、アプリケーションのスケーラビリティしきい値は1秒あたり5リクエストです。
水平スケーリングを適用して Mule ワーカーの数を増やすと、アプリケーションにどのような影響がありますか?
A. アプリケーションのスケーラビリティしきい値は、水平スケーリングに関係なく、1秒あたり5リクエストです。
B. プロセス実行時間の合計は100ミリ秒 (0.1秒)になりました
C. アプリケーションのスケーラビリティしきい値は、1秒あたり10リクエストになりました。
D. すでに実行中のアプリケーションには水平スケーリングを適用できません
Answer: (解答を表示する**)**
アプリケーションはデータの一貫性を維持するために一度に 1 つのプロセスのみを処理するように設計されているため、水平スケーリングによって処理制限が増加しない理由は次のとおりです。
単一プロセス制約:
アプリケーションは一貫性を重視した設計のため、一度に 1 つのトランザクションを処理するように制限されています。つまり、水平スケーリング (ワーカーの追加) を行っても、この制限を超えて処理速度が上がることはありません。
実行時間:
各リクエストには200ミリ秒かかるため、1秒あたり5リクエストが最大処理しきい値となります。ワーカー数を増やしても、この単一プロセス制限は回避できません。
正解 A):
この制約はアプリケーションの設計に固有のものであるため、スケーラビリティは 1 秒あたり 5 件のリクエストのままとなります。
誤った選択肢:
オプション B は実行時間の変更を示唆しますが、これは水平スケーリングの影響を受けません。
オプション C ではスループットを 2 倍にすることを想定していますが、アプリケーションがシングル スレッドであるため、これは不可能です。
オプション D は、水平スケーリングは適用できないことを示唆していますが、これは誤りです。ただし、このコンテキストではスケーリングによってスループットは向上しません。
参照
Mule アプリケーションのスケーリングと同時実行の詳細については、アプリケーションのパフォーマンスとスケーリングの制限に関する MuleSoft のドキュメントを参照してください。

最新問題**: 15**
ある組織では、様々なクラウドベースのSaaSシステムと複数のオンプレミスシステムを使用しています。オンプレミスシステムは組織のアプリケーションネットワークの重要な部分であり、組織のイントラネット内からのみアクセスできます。
クラウドベースの SaaS システムとオンプレミス システムの両方との統合をサポートするために Anypoint Platform を構成して使用する最適な方法は何ですか?
A) Anypoint Platform Private Cloud Edition コントロール プレーンによって管理される Anypoint VPC で、CloudHub でデプロイされた Mule ランタイムを使用する B) MuleSoft がホストする Anypoint Platform コントロール プレーンによって管理される共有ワーカー クラウドで、CloudHub でデプロイされた Mule ランタイムを使用する C) Anypoint Platform Private Cloud Edition コントロール プレーンによって管理され、外部ネットワーク アクセスのない完全に分離されたオンプレミスの Mule ランタイム インストールを

使用する D) MuleSoft がホストする Anypoint Platform コントロール プレーンによって管理される、Cloud Hub でデプロイされたオンプレミスの Mule ランタイムと手動でプロビジョニングされたオンプレミスの Mule ランタイムの組み合わせを使用する



A. オプションA
B. オプションB
C. オプションC
D. オプションD
Answer: (解答を表示する**)**
正解: MuleSoft がホストする Platform コントロール プレーンによって管理される、CloudHub でデプロイされたオンプレミスの Mule ランタイムと手動でプロビジョニングされたオンプレミスの Mule ランタイムの組み合わせを使用します。
****
与えられたシナリオから得られる重要な詳細:
>> 組織はクラウドベースとオンプレミスの両方のシステムを使用しています
>> オンプレミスのシステムは、組織のイントラネット内からのみアクセスできます。上記の重要な詳細に基づいて、与えられた選択肢を評価してみましょう。
>> CloudHub にデプロイされた Mule ランタイムは、MuleSoft がホストするコントロールプレーンを使用してのみ制御できます。Private Cloud Edition のコントロールプレーンを使用して CloudHub Mule ランタイムを制御することはできません。したがって、この提案は無効です。
>> CloudHubでデプロイされたMuleランタイムを、MuleSoftがホストするAnypoint Platformが管理する共有ワーカークラウドで使用することは、このシナリオとは全く無関係であり、愚かな選択です。したがって、これを示唆する選択肢は無効です。
>> Anypoint Platform Private Cloud Edition コントロールプレーンによって管理され、外部ネットワークへのアクセスが一切ない完全に分離されたオンプレミスの Mule ランタイムを使用すれば、オンプレミス統合は可能です。ただし、外部アクセスが一切ないため、SaaS ベースのアプリケーションとの統合はできません。さらに、CloudHub でホストされるアプリケーションは、SaaS ベースのアプリケーションとの統合に最適です。したがって、この方法をお勧めします。

これらの混合/ハイブリッド統合をサポートするために Anypoint Platform を設定して使用する最適な方法は、MuleSoft がホストする Platform コントロール プレーンによって管理される、CloudHub でデプロイされたオンプレミス Mule ランタイムと手動でプロビジョニングされたオンプレミス Mule ランタイムを組み合わせて使用することです。

最新問題**: 16**
ダウンタイムが繰り返し発生することが知られている Order API を呼び出す必要がある API 実装を設計しています。
このため、Order API が利用できない場合はフォールバック API が呼び出されます。
フォールバック API の呼び出しを設計する際に、どのようなアプローチが最高の回復力を提供しますか?
A. Anypoint Exchange で適切な既存のフォールバック API を検索し、注文 API に加えてこのフォールバック API への呼び出しを実装します。
B. API マネージャーで Order API の別のエントリを作成し、プライマリ Order API が利用できない場合にこの API をフォールバック API として呼び出します。
C. Order APIが利用できない場合は、HTTP 307 Temporary Redirectステータスコードを介してクライアントリクエストをフォールバックAPIにリダイレクトします。
D. HTTP リクエスター コンポーネントに、注文 API を呼び出すオプションを設定します。注文 API から HTTP 4xx または 5xx 応答ステータス コードが返されるたびに、代わりにフォールバック API が呼び出されます。
Answer: (解答を表示する**)**
正解: Anypoint Exchangeで適切な既存のフォールバックAPIを検索し、注文APIに加えてこのフォールバックAPIへの呼び出しを実装します。
****
>> API クライアントが HTTP 3xx 一時リダイレクト ステータス コードを受信し、別の API を呼び出すためにフォールバック ロジックを API クライアント側で実装する必要があるという事前承認済みの合意がない限り、これは理想的でも適切なアプローチでもありません。
>> APIマネージャーで同じ注文APIの別のエントリを作成すると、同じAPI実装の上に別のインスタンスが作成されるだけです。そのため、同じAPIのクローンをフォールバックAPIとして使用しても意味がありません。フォールバックAPIは、理想的にはプライマリAPIとは異なるAPI実装である必要があります。
>> 現在、Anypoint HTTP Connector では、応答として特定の HTTP ステータス コードを受信したときにフォールバック API を呼び出すことができるオプションは提供されていません。
指定されたオプションの中で TRUE となる唯一のステートメントは、適切な既存のフォールバック API を Anypoint エクスチェンジで検索し、注文 API に加えてこのフォールバック API への呼び出しを実装することです。

最新問題**: 17**
CloudHub 専用ロードバランサーを使用する必要がある条件は何ですか?

A. 同じ Mule アプリケーションの別々のデプロイメント間でリージョン間の負荷分散が必要な場合
B. 顧客がホストする Mule ランタイムにデプロイされた API 実装にカスタム DNS 名が必要な場合
C. 複数の CloudHub ワーカー間での API 呼び出しを負荷分散する必要がある場合
D. API実装とAPIクライアント間でサーバー側負荷分散TLS相互認証が必要な場合
Answer: D (メーッセージを残す)
正解: API実装とAPIクライアント間でサーバー側負荷分散TLS相互認証が必要な場合
****
事実/メモリのヒント: CloudHub 専用ロードバランサには多くの利点がありますが、検討する際に心に留めておくべき重要な点が 2 つあります。
>> CloudHub にデプロイされたアプリでカスタム DNS 名を持つ URL エンドポイントを持つ
>> HTTPS と双方向 (相互) 認証の両方に対してカスタム証明書を構成します。
この質問に対して提供されているオプションは次のとおりです。
>> DLB を使用して、同じ Mule アプリケーションの個別のデプロイメント間でリージョン間の負荷分散を実行することはできません。
>> 複数のDLB URLを同じMuleアプリにマッピングするルールを設定できます。ただし、その逆（複数のMuleアプリが同じDLB URLを持つ）は不可能です。
>> DLB は Cloudhub にデプロイされた Mule アプリのカスタム DNS 名の設定に役立ちますが、顧客がホストする Mule ランタイムにデプロイされたアプリの場合は役に立ちません。
>> DLB を使用することで、複数の CloudHub ワーカー間で API 呼び出しの負荷分散を行うことは可能ですが、必須ではありません。SLB (共有ロードバランサー) を使用しても同様の負荷分散を実現できます。DLB は必ずしも必要ではありません。
したがって、シナリオに適合し、DLB を使用する必要がある唯一の適切なオプションは、API 実装と API クライアント間で TLS 相互認証が必要な場合です。

最新問題**: 18**
REST API 実装の統合テストの特性として最も可能性が低いものは何ですか?
A. テストでは、すべてのソースシステムおよび/またはターゲットシステムが構成され、アクセス可能である必要があります。
B. テストは、Mule アプリケーションがコンパイルされパッケージ化された直後に実行されます。
C. テストは外部HTTPリクエストによってトリガーされます
D. テストは既知のリクエストペイロードを準備し、レスポンスペイロードを検証します。
Answer: (解答を表示する)
正解: テストは、Mule アプリケーションがコンパイルされパッケージ化された直後に実行されます。
****
>> 統合テストは、完全にカバーするために追加する必要があるテストの最後のレイヤーです。
>> これらのテストは、実際には完全な構成で実行されている Mule に対して実行され、PROD で動作しているときに外部ソースからテストされます。
>> これらのテストは、実際のトランスポートを有効にした状態でアプリケーション全体をテストします。そのため、テストの実行時に外部システムに影響が及ぶ可能性があります。
したがって、これらのテストは、Mule アプリケーションがコンパイルされパッケージ化された直後には実行されません。
参考までに... ユニット テストは、Mule アプリケーションがコンパイルされパッケージ化された直後に実行されるテストです。

最新問題: **19**
多数の REST API を実装する Mule アプリケーションは、組織外部からアクセスできない独自のサブネットに展開されます。
外部のビジネスパートナーはこれらのAPIにアクセスする必要がありますが、これらのAPIはパートナー専用のサブネット（パートナーサブネット）からのみ呼び出すことができます。このサブネットはパブリックインターネットからアクセス可能であり、外部パートナーからのアクセスを可能にします。
Anypoint Platform と Mule ランタイムは既に Partner-subnet にデプロイされており、これらの Mule ランタイムは既に API にアクセスできます。
現在 API を使用している他のアプリケーションへの影響を最小限に抑えながら、これらの要件に準拠するための最もリソース効率の高いソリューションは何ですか?
A. APIごとにAPIプロキシMuleアプリケーションを実装（または生成）し、APIプロキシをMuleランタイムにデプロイします。
B. API を Mule アプリケーションとして複製し、Mule ランタイムにデプロイします。
C. Mule ランタイムを実行している同じサーバーに API 実装を再デプロイします。
D. パートナーによる利用を可能にするために、各 API に追加のエンドポイントを追加します。
Answer: (解答を表示する**)**

最新問題: **20**
API 実装が CloudHub にデプロイされます。
デフォルトの Anypoint Platform 機能を使用すると、どのような条件でアラートを生成できますか。アラート条件は API 実装に対する API 呼び出しによって異なります。
A. API呼び出しがAPI実装の内部DNSレコードに直接送信される場合
B. API呼び出しが安全なTLS/SSL通信チャネルを経由していない場合
C. APL の通知が API とは異なる地域から発信された場合
D. API呼び出し回数が閾値を下回った場合
Answer: D (メーッセージを残す**)**
Anypoint Platform のデフォルトのアラート機能:
Anypoint Platform には、呼び出し回数のしきい値の設定など、API 呼び出し条件を監視するためのすぐに使用できるアラート機能が備わっています。
トラフィックの高低（定義されたしきい値を超える呼び出しや下回る呼び出し）などの条件に対してアラートを設定できます。
オプションの評価:
オプション A: Anypoint Platform は、DNS レコードに基づく直接的なアラートを提供しません。
オプション B: Anypoint Platform では、呼び出しで TLS/SSL が使用されるかどうかに基づいてデフォルトのアラートは提供されません。これにはカスタム構成が必要になります。
オプション C: 地理位置情報に基づくアラートは、Anypoint Platform ではネイティブにサポートされていません。
オプション D (正解): API 呼び出ししきい値 (設定されたしきい値を下回る呼び出しなど) に基づくアラートがサポートされており、デフォルトの Anypoint アラート機能の一部として設定できます。
結論：
オプション D が正解です。Anypoint Platform では、しきい値を下回ったり超えたりした API 呼び出しの数に基づいてアラートを設定できます。
詳細については、MuleSoft の Anypoint モニタリングとアラート構成に関するドキュメントを参照してください。

最新問題**: 21**
APIクライアントが既存のAPI実装から1つのメソッドを呼び出します。その後、API実装が更新されます。API実装にどのような変更を加えると、APIクライアントの呼び出しロジックも更新する必要がありますか？
A. APIクライアントによって呼び出されたメソッドのレスポンスのデータ型が変更された場合
B. APIクライアントが使用するリソースに新しいメソッドが追加されたとき
C. APIクライアントによって呼び出されたメソッドに新しい必須フィールドが追加されたとき
D. APIクライアントによって呼び出されたメソッドに子メソッドが追加された場合
Answer: C (メーッセージを残す**)**
正解: APIクライアントによって呼び出されたメソッドに新しい必須フィールドが追加されたとき
****
>> 一般的に、API 契約が破綻した場合、API クライアントのロジックを更新する必要があります。
>> APIに新しいメソッドまたは子メソッドが追加されても、APIクライアントは既存のメソッドを引き続き使用できるため、動作に支障はありません。したがって、これら2つのオプションは無効です。
>> 応答のデータ型が変更された場合」と 新しい必須フィールドが追加された場合」の 2 つが残っています。
>> レスポンスのデータ型を変更すると、APIの規約に違反することになります。ただし、ここで問題となるのは 呼び出し」ロジックであり、レスポンス処理ロジックではありません。APIクライアントはAPIを正常に呼び出し、レスポンスを受け取ることはできますが、レスポンスの一部フィールドのデータ型は変更されます。
>> 新しい必須フィールドを追加すると、APIの呼び出し規約に違反します。新しい必須フィールドを追加すると、APIの契約は、APIクライアント/APIコンシューマーとAPIプロバイダーの間で締結されているRAMLまたはAPI仕様の合意に違反します。そのため、APIクライアントの呼び出しロジックも更新する必要があります。

最新問題**: 22**
IT セキュリティ コンプライアンス監査人は、セキュリティ対策を満たすためにどの非機能要件 (NFR) がすでに実装されているかを評価しています。
* Web APIにはレート制限SLAがあります
* 基本認証 - LDAP
* JSON脅威保護
* TP許可リストポリシーが適用されました
どの 2 つの NFR が適用されますか?
A. API呼び出しは既知のサブネット範囲から行われています
B. ログイン資格情報を検証するためにサポートされているユーザー名/パスワード
C. 重要な情報の漏洩を防ぐために機密データがマスクされます
D. APIはXML呼び出し攻撃から保護されています
E. パフォーマンスの期待値は、1秒あたり最大1,000リクエストまで許容されます。
Answer: A,B (メーッセージを残す**)**
非機能要件 ℕFR）の理解:
このコンテキストにおける NFR は、レート制限、LDAP ベースの認証、JSON 脅威保護、IP 許可リスト ポリシーなど、Web API に実装されたセキュリティ対策に関連しています。
オプションの評価:

オプション A (正解): IP 許可リスト ポリシーは、既知のサブネットへのアクセスを制限し、API 呼び出しが定義された範囲の IP から行われるようにします。
オプション B (正解): LDAP を使用した基本認証では、ユーザー名/パスワードの検証が強制され、ID 検証の NFR が満たされます。
オプション C: 機密データのマスキングは、記載されているポリシーのいずれもデータのマスキングに対応していないため、記載されている NFR の一部ではありません。
オプション D: XML 脅威保護については言及されていないため、このオプションは正しくありません。
オプション E: レート制限はパフォーマンス制御を意味しますが、特定のパフォーマンス期待を直接強制するものではありません。
結論：
オプション A と B は、IP 範囲の制限とユーザー名/パスワードの認証に関連する実装されたセキュリティ対策に直接対処しているため、正解です。
LDAP、レート制限、許可リスト ポリシーの詳細については、MuleSoft の API セキュリティ ポリシーに関するドキュメントを参照してください。

最新問題**: 23**
大企業は、データとメタデータがローカルに保存されるという企業の IT ポリシー要件に基づいて、独自のデータ センターに IT インフラストラクチャを実装したいと考えています。
Mule コントロール プレーンと Mule ランタイム プレーンのどの組み合わせが要件を満たしていますか?
A. コントロールプレーンとMuleSoftがホストするランタイムプレーン用のAnypoint Platform Private Cloud Edition
B. MuleSoft がホストするコントロールプレーンとランタイムプレーン用の Anypoint Runtime Fabric
C. MuleSoft がホストするコントロールプレーンと、ランタイムプレーン用の顧客がホストする Mule ランタイム
D. コントロールプレーン用のAnypoint Platform Private Cloud Editionと、ランタイムプレーン用の顧客ホスト型Muleランタイム
Answer: (解答を表示する**)**
制御プレーンとランタイムプレーンの理解:
コントロールプレーン :コントロールプレーンは、Muleアプリケーションの管理、監視、およびデプロイを担います。プライベートクラウドエディション（PCE）では、このコントロールプレーンは顧客のインフラストラクチャ内にオンプレミスでデプロイされ、データレジデンシーとセキュリティ要件を満たします。
ランタイムプレーン :ランタイムプレーンは、Mule アプリケーションを実行する Mule ランタイムで構成されています。これらのランタイムをお客様のインフラストラクチャ内でホストすることで、データとメタデータをローカルに保持でき、データレジデンシーに関する企業ポリシーに準拠できます。
オプションの評価:
オプション A: コントロール プレーンに Anypoint Platform Private Cloud Edition を使用し、MuleSoft がホストするランタイム プレーンを使用すると、ランタイム プレーンが MuleSoft によってホストされ、データがローカルに保持されないため、要件を満たせません。
オプション B: ランタイム プレーン用の Anypoint Runtime Fabric を備えた MuleSoft がホストするコントロール プレーンでは、メタデータは依然として MuleSoft のクラウドで管理されるため、データとメタデータをオンプレミスで保持するという要件に準拠しません。
オプション C: MuleSoft がホストするコントロール プレーンと顧客がホストする Mule ランタイムでは、メタデータがオンプレミスではなくクラウドに存在するため、常駐要件を満たしません。

オプション D (正解): コントロール プレーンと顧客がホストする Mule ランタイム用の Anypoint Platform Private Cloud Edition (PCE) は、コントロール プレーンとランタイム プレーンの両方が顧客のデータ センター内でホストされるため、両方の要件を満たします。
結論：
オプション D が正解です。コントロール プレーンとランタイム プレーンの両方がオンプレミスでホストされ、企業の IT ポリシーに従ってデータとメタデータがローカルに保存されるようになります。
詳細については、MuleSoft の Private Cloud Edition の展開とオンプレミスのランタイム構成に関するドキュメントを参照してください。

最新問題**: 24**
ある組織は、Azure 環境で MuleSoft がホストするランタイム プレーン機能（HTTP ロードバランシング、ゼロダウンタイム、水平および垂直スケーリングなど）を導入したいと考えています。これらの機能を実現するための組織の労力を最小限に抑えるランタイム プレーンはどれでしょうか。
A. Anypoint ランタイムファブリック
B. Pivotal Cloud Foundry 向け Anypoint プラットフォーム
C. クラウドハブ
D. 顧客ホスト型と MuleSoft ホスト型の Mule ランタイムのハイブリッドな組み合わせ
Answer: (解答を表示する**)**
正解: Anypoint Runtime Fabric
****
>> お客様が既に Azure 環境をご利用の場合、一部の Mule ランタイムを Azure でホストし、残りの Mule ランタイムを MuleSoft でホストするハイブリッドモデルを採用するのは、決して理想的なアプローチではありません。これは不要であり、無駄です。
>> CloudHub は Mulesoft がホストするランタイムプレーンであり、AWS 上に存在します。CloudHub をお客様の Azure 環境に向けるようにカスタマイズすることはできません。
>> Pivotal Cloud Foundry 向け Anypoint Platform は、Pivotal Cloud Foundry が提供するインフラストラクチャ専用です。
>> Anypoint Runtime Fabric は、Mule アプリケーションと API ゲートウェイのデプロイメントとオーケストレーションを自動化するコンテナサービスであるため、正解です。Runtime Fabric は、AWS、Azure、仮想マシン (VM)、ベアメタルサーバー上の顧客管理インフラストラクチャ内で実行されます。
-Anypoint Runtime Fabric の機能には次のようなものがあります。
- アプリケーションごとに個別の Mule ランタイムを実行することで、アプリケーション間の分離を実現します。
- 同じリソース セットで複数のバージョンの Mule ランタイムを実行する機能。
- 複数のレプリカにわたってアプリケーションをスケーリングします。
- 自動化されたアプリケーションフェイルオーバー。
-Anypoint Runtime Manager によるアプリケーション管理。

最新問題**: 25**
プロセス API に適用される可能性が最も低い API ポリシーは何ですか?
A. カスタム回路ブレーカー
B. クライアントIDの強制
C. レート制限

D. JSON脅威保護
Answer: D (メーッセージを残す**)**
正解: JSON脅威保護
****
事実：技術的には、どのレイヤーにどのポリシーを適用できるかについて制限はありません。どのレイヤーのAPIにも、どのポリシーを適用できます。ただし、APIにポリシーを盲目的に適用する前に、コンテキストを適切に考慮する必要があります。
そのため、この質問では、プロセス API に適用される可能性が最も低いポリシーを求めました。
指定されたオプションから:
>> 「JSON 脅威保護」を除くすべてのポリシーは、プロセス層の API にためらうことなく適用できます。
>> JSON脅威保護ポリシーは、外部APIクライアントからの疑わしいJSONペイロードを阻止するエクスペリエンスAPIに最適です。エクスペリエンスAPIを呼び出す外部クライアントからの悪意のある、または有害な可能性のあるJSONペイロードを回避することで、セキュリティ面をより強化します。
外部 API クライアントがプロセス API を直接呼び出すことは決して許可されず、また、このような悪意のある有害な JSON ペイロードは常にこのポリシーを使用するエクスペリエンス API レイヤーでのみ停止されるため、同じポリシーがプロセス レイヤー API に再度適用される可能性は最も低くなります。

最新問題**: 26**
別紙をご参照ください。ある組織では、Mule スタンドアロンランタイムを実行しており、Anypoint Platform の外部 ID プロバイダとして Active Directory を設定しています。組織には、他のシステムコンポーネントに充てる予算がありません。
特定の内部ユーザー グループへのアクセスを最も効果的に制限するには、組織内のすべての API インスタンスにどのようなポリシーを適用する必要がありますか?
A. 基本認証 - LDAP ポリシーを適用します。内部 Active Directory がユーザー認証用の LDAP ソースとして設定されます。
B. クライアントID強制ポリシーを適用します。特定のユーザーグループは、特定のクライアント資格情報を使用するようにクライアントアプリケーションを構成します。
C. IPホワイトリストポリシーを適用します。特定のユーザーのワークステーションのみがホワイトリストに登録されます。
D. OAuth 2.0アクセストークン強制ポリシーを適用します。内部Active DirectoryがOAuthサーバーとして構成されます。
Answer: A (メーッセージを残す**)**
正解: 基本認証 - LDAP ポリシーを適用します。内部 Active Directory がユーザー認証用の LDAP ソースとして構成されます。
****
>> IPホワイトリストはこの目的には適していません。さらに、ユーザーのワークステーションはネットワーク内で必ずしも固定IPアドレスを持つとは限りません。
>> OAuth 2.0 の適用には、組織のシステム コンポーネントに含まれていないクライアント プロバイダーが必要です。
>> すべてのユーザーが個別のクライアント資格情報を作成し、使用に合わせて構成できるようにすることは効果的なアプローチではありません。
効果的な方法は、基本認証 - LDAP ポリシーを適用することです。内部 Active Directory は、ユーザーを認証するための LDAP ソースとして構成されます。

最新問題**: 27**
下流 API の応答時間と需要を改善するという目標を達成するために、サーキットブレーカー戦略が計画されています。

* サーキットオープン: 3分間に1分あたり10件以上のエラー
* 回路半開: 1分あたり1回のエラー
* 回路閉鎖: 5分間、1分あたり1件未満のエラー
エンジニアリング チームからのいくつかの提案のうち、どのオプションがこの目標を満たすでしょうか?
A. サーキットブレーカーを実装し、必要な設定のポリシーテンプレート式を含むカスタムポリシーを作成します。
B. Circuit Open/Closed 構成の Anypoint Monitoring アラートを作成し、Circuit Half-Open 構成の再試行戦略を実装します。
C. APIインスタンスにサーキットブレーカーポリシーを追加し、必要な設定を行います。
D. Muleアプリケーションで戦略を実装し、YAML構成で設定を提供します。
Answer: C (メーッセージを残す)
サーキットブレーカーポリシーの理解:
サーキットブレーカーは、障害を検出し、アプリケーションが失敗した操作を繰り返し実行しようとするのを防ぐために使用される設計パターンです。この場合、応答時間の改善と下流APIへの負荷軽減に役立ちます。
指定された構成には、時間の経過に伴うエラー率に基づいて回路を開く、半開にする、閉じる条件が含まれます。
回路オープン: 3 分連続で 1 分あたり 10 件を超えるエラーが発生した場合にトリガーされます。
回路半開: 1 分間に 1 つのエラーが発生すると、回路は半開状態に移行します。
回路が閉じます: エラー率が 5 分間に 1 分あたり 1 件未満になると、回路が閉じます。
オプションの評価:
オプションA :テンプレート式を使用してカスタムポリシーを作成することは可能ですが、カスタム開発が必要になります。Anypoint PlatformにはすでにCircuit Breakerポリシーが用意されているため、このソリューションは効率性が低く、より複雑になります。
オプションB :Anypoint MonitoringアラートはAPIの監視に使用できますが、サーキットブレーカー機能は提供されません。また、ハーフオープン状態に対する再試行戦略を実装するだけでは、必要なサーキットブレーカーの動作を実現できません。
オプションC（正解）Anypoint Platform上のAPIインスタンスにCircuit Breakerポリシーを追加すると、サーキットブレーカーの条件を直接設定できます。このアプローチでは、組み込みのCircuit Breakerポリシーを使用し、エラーしきい値や時間間隔などのパラメータを要件に合わせて設定できます。このソリューションは効率的で信頼性が高く、Anypointのすぐに使用できる機能を活用します。
オプションD :YAML設定を使用してMuleアプリケーション内で戦略を実装すると、複雑になり、管理が困難になる可能性があります。さらに、このシナリオにはより適しているAnypoint Platformの組み込みCircuit Breakerポリシーを活用できません。
結論：
オプションCはAnypoint PlatformのCircuit Breakerポリシーを活用しているため、正しい選択です。このソリューションでは、しきい値と時間間隔を指定どおりに設定できるため、Anypointのマネージドポリシー機能を活用しながら、応答時間を改善し、下流APIへの負荷を軽減できます。
詳細な構成ガイダンスについては、API Manager での Circuit Breaker ポリシーの実装に関する MuleSoft のドキュメントを参照してください。

最新問題**: 28**
ある組織は、様々なAPIレイヤーを用いてモバイルクライアントとバックエンドシステムを統合するAPI主導のアーキテクチャを構築しました。バックエンドシステムは複数の専用コンポーネントで構成され、REST API経由でアクセスできます。プロセスAPIとエクスペリエンスAPIは、バックエンドデータモデルとは異なる境界付きコンテキストモデルを共有しています。バックエンドシステムから消費されるデータの処理を支援するために、このアーキテクチャにどのような標準モデル、境界付きコンテキストモデル、または破損防止レイヤーを追加するのが最適でしょうか？

A. 各レイヤーに境界コンテキストモデルを作成し、境界コンテキストが重なる場合は重ね合わせることで、API開発者が上流と下流のデータモデルの違いを認識できるようにします。
B. バックエンド モデルと API 主導のモデルを組み合わせた標準モデルを作成し、データ モデルを簡素化および統合し、データ変換を最小限に抑えます。
C. システム層の境界付きコンテキストモデルを作成し、バックエンドデータモデルと密接に一致させ、破損防止層を追加して、異なる境界付きコンテキストがシステム層とプロセス層全体で連携できるようにします。
D. すべての API に破損防止レイヤーを作成し、すべてのデータ モデルが互いに一致するように変換し、データが API 間で簡単に移動できるようにして、標準モデルを構築する複雑さとオーバーヘッドを回避します。
Answer: C (メーッセージを残す**)**
正解: システム層の境界付きコンテキスト モデルを作成してバックエンド データ モデルと密接に一致させ、破損防止層を追加して、異なる境界付きコンテキストがシステム層とプロセス層全体で連携できるようにします。
****
>> 組織はすでに努力を重ね、エクスペリエンス API とプロセス API 用の境界付きコンテキスト モデルを作成しているため、ここでは標準モデルは選択できません。
>> エクスペリエンスAPIとプロセスAPIは同じ境界コンテキストモデルを共有しているため、すべてのAPIに破損防止レイヤーを設けるのは不要かつ無効です。現在、適切なアプローチを選択すべきなのはシステム層APIのみです。
>> したがって、プロセス層とシステム層の間に破損防止層を設けるのが効果的です。また、このアプローチを高速化するために、システムAPIはバックエンドシステムのデータモデルを模倣できます。

最新問題**: 29**
ある組織には、HTTP POST経由でJSONデータを受け入れる複数のAPIがあります。これらのAPIはすべて公開されており、複数のモバイルアプリケーションやWebアプリケーションに関連付けられています。
組織はこれらの API に対して認証やコンプライアンス ポリシーを使用することを望んでいませんが、同時に、悪意のある人物が、API 実装を実行しているアプリケーションやサーバーを何らかの形で侵害する可能性のあるペイロードを送信する可能性があることを懸念しています。
この脅威にさらされることに対処できる、すぐに使用できる Anypoint Platform ポリシーは何ですか?
A. すべてのAPI呼び出しにHTTPS相互認証を使用して、悪意のある行為者をシャットアウトします。
B. すべてのAPIにIPブラックリストポリシーを適用します。ブラックリストにはすべての悪質な行為者が含まれます。
C. 悪意のあるデータが使用される前にそれを検出するヘッダー挿入および削除ポリシーを適用する
D. すべてのAPIにJSON脅威保護ポリシーを適用して、潜在的な脅威ベクトルを検出します。
Answer: D (メーッセージを残す**)**
正解: 潜在的な脅威ベクトルを検出するために、すべてのAPIにJSON脅威保護ポリシーを適用する
****
>> 通常、API が特定の消費者 (既知の消費者/顧客) 向けに設計および開発されている場合は、トラフィックがその消費者/顧客からのみ発生するように、同じものを IP ホワイトリストに登録します。
>> ただし、このシナリオでは、API が公開されており、非常に多くのモバイル アプリケーションや Web アプリケーションで使用されているため、すべての悪意のある行為者を特定してブラックリストに登録することは不可能です。
>> したがって、JSON 脅威保護ポリシーは、そのような悪意のある行為者による不正な JSON ペイロードを防ぐ最良の方法です。

最新問題: **30**
プラットフォームアーキテクトは、クライアントに属するすべての保険証券の表示など、様々なタスクを実行する、レガシーなモノリシックなSOAPベースのWebサービスを継承しています。このサービスは、生命保険管理システムと損害保険管理システムという2つのバックエンドシステムに接続し、各システム内で保険証券情報を照会し、結果を集約して、SOAPベースの応答をユーザーインターフェース (UI) に表示します。
アーキテクトは、API 主導の規則に従うためにモノリシック Web サービスを分割したいと考えています。
サービスのどの部分をプロセス層に配置する必要がありますか?
A. 管理システムからの保険契約情報を統合する
B. UI に SOAP ベースの応答を表示する
C. 各バックエンド管理システムへの接続を認証および維持する
D. 管理システムからのデータのクエリ
Answer: (解答を表示する**)**
API 主導の接続アプローチでは、各レイヤー (システム、プロセス、エクスペリエンス) には明確な目的があります。
システム API: これらの API はバックエンド システムに直接接続し、標準化された方法でデータを公開およびロック解除します。
プロセス API: さまざまなシステム間でデータをオーケストレーションおよび処理し、必要に応じて情報を組み合わせます。
エクスペリエンス API: 特定のユーザー インターフェイスまたはアプリケーション向けに設計されており、多くの場合、各コンシューマー アプリケーションのニーズに合わせてデータ形式を変換します。
オプションAが正しい理由:
プロセスAPIは、複数のシステムからデータを統合するように設計されており、生命保険システムと損害保険システムの両方から保険契約情報を集約する機能と整合しています。この集約ロジックは、理想的にはプロセス層に配置され、データ取得とデータオーケストレーションを分離します。
この機能をプロセス レイヤーに移動すると、再利用性とモジュール性が実現され、必要に応じて他のエクスペリエンス API やサービスでも結合されたポリシー データを活用できるようになります。
誤った選択肢:
オプション B (SOAP ベースの応答の提示) は、エクスペリエンス レイヤーによって管理されます。このレイヤーは、特定のインターフェースに合わせてデータ形式を適応させるためです。
オプション C (バックエンド接続の認証と維持) は通常、バックエンドの統合とセキュリティ処理が行われるシステム層内で処理されます。
オプション D (データのクエリ) はシステム API の機能であり、バックエンド システムに直接アクセスし、追加の処理なしで生データを公開します。
参照
API 主導のアーキテクチャと各レイヤーの役割の詳細については、MuleSoft の API 主導の接続性と API レイヤーに関するドキュメントを参照してください。

最新問題: **31**
ある組織は、顧客の住所情報を取得するための顧客住所APIを実装しました。このAPIは複数の環境にデプロイされており、すべての環境でクライアントIDを適用するように設定されています。
開発者は、ユーザーが住所を更新できるようにするクライアントアプリケーションを作成しています。開発者はAnypoint ExchangeのCustomer Address APIを見つけ、それをクライアントアプリケーションで使用したいと考えています。

Anypoint Platform によって自動的に実行できる API へのアクセスのどの手順ですか?
A. 選択したSLA層に対するクライアントアプリケーション要求を承認します
B. クライアントアプリケーションの資格情報を使用して、複数の環境にデプロイされた適切なAPIインスタンスへのアクセスを要求します。
C. クライアント アプリケーションの資格情報を使用して API を呼び出すようにクライアント アプリケーションを変更します。
D. APIへのアクセスを要求するためにAnypoint Exchangeで新しいアプリケーションを作成します
Answer: A (メーッセージを残す)
正解: 選択したSLA層のクライアントアプリケーション要求を承認する
****
>> 選択したSLA層に対するクライアントアプリケーションリクエストの承認のみを自動化できます
>> 提供された残りのオプションは無効です

最新問題: **32**
既存のQuoting APIはRAMLで定義されており、RESTクライアントが見積りエンジンとやり取りするために使用されています。現在、見積りの作成を可能にするリソースがRAMLで定義されていますが、既存の見積りの更新を可能にするための新たな要件が最近受領されました。
この変更を容易に処理できるようにするには、どの 2 つのアクションを実行する必要がありますか?
2つの回答を選択してください
A. 新しい更新リクエストに対応するためにAPI実装を更新します
B. 古いクライアントアプリケーションを削除し、変更を反映した新しいクライアントアプリケーションを作成します。
C. 更新リクエストの新しいメソッドの詳細で RAML を更新します。
D. Exchange の API の既存のバージョンを廃止します
E. 更新されたエンドポイントへのアクセスを許可するために、API Manager に新しい API ポリシーを追加します。
Answer: A,C (メーッセージを残す)
既存の見積りの更新を許可するという新しい要件に対応するには、次のアクションを実行する必要があります。
RAML 定義を更新します (オプション C):
RAML仕様はAPIの構造と動作を定義します。相場情報を更新するための新しいメソッド（PUTやPATCHなど）を追加するには、この新しいエンドポイントを含めるようにRAMLを変更する必要があります。これにより、API仕様が最新の状態になり、新しい機能を正確に反映できるようになります。
API 実装を更新する (オプション A):
RAMLが更新されたら、新しい更新リクエストを処理できるようにバックエンドAPI実装も変更する必要があります。これには、更新リクエストを処理・検証し、必要なバックエンドリソースに接続し、既存の見積りに変更を適用するロジックを追加することが含まれる場合があります。

誤った選択肢:
オプション B (クライアントの削除と新規作成) は不要です。クライアント アプリケーションはそのまま残すことができ、完全に置き換える必要はありません。
下位互換性が維持されている場合、オプション D (既存のバージョンの廃止) は必要ない場合もあります。
オプション E (新しいポリシーの追加) は機能の変更を容易にせず、更新機能の実装とは無関係です。
参照
RAML 定義と API 実装の更新の詳細については、RAML および RESTful API のプラクティスに関する MuleSoft の API 設計ドキュメントを参照してください。

最新問題: **33**
統合テストではなく MUnit テストに適したシナリオはどれですか?
A. 依存関係（他のWeb APIなど）への読み取り専用のやり取り用
B. テストに実装の詳細に関する知識が必要ない場合
C. モックが許可されていない場合
D. SoapUIを使用して実装されたテストの場合
Answer: A (メーッセージを残す**)**
MUnitは、Anypoint Studio内で自動テストを作成・実行するためのMuleSoftのテストフレームワークです。Muleアプリケーションのユニットテスト用に特別に設計されており、テスト対象コンポーネントの内部動作や実装の詳細を理解する必要がないテストに最適です。
MUnit の理想的な使用例:
MUnitは、個々のフロー、関数、またはコンポーネントを個別にテストする場合に最適です。このタイプのテストは、システム全体を理解する必要がなく、各ユニットの動作を検証することに重点を置いています。
ユニット テストでは外部統合や依存関係が稼働している必要がないため、MUnit では外部サービスや API の動作をシミュレートするためにモックがよく使用されます。
オプションBが正しい理由:
オプションBは、システム統合よりも機能テストに重点を置くユニットテストの概念に沿っています。一方、統合テストは実装に関する知識と実際のエンドポイントを必要とするため、MUnitのスコープには適していません。
誤った選択肢:
MUnit は依存関係をシミュレートするためのモック機能を備えて設計されているため、オプション A (読み取り専用インタラクション) とオプション C (モックなし) は MUnit の一般的なテスト環境には適していません。
オプション D (SoapUI ベースのテスト) は外部テスト ツールを提案しますが、MUnit は MuleSoft に固有です。
参照
MUnit のベスト プラクティスの詳細については、MuleSoft の MUnit ドキュメントを参照してください。

最新問題: **34**
ある組織では、InfoSec チームが Anypoint Platform 関連のデータ トラフィックを調査しています。
Anypoint Platform で監視およびアラートに使用できるデータのほとんどはどこから発生するのでしょうか?
A. デプロイメントモデルに応じて、Mule ランタイムまたは API 実装から
B. 共有ロードバランサ、VPC、Muleランタイムなど、Anypoint Platformのさまざまなコンポーネントから
C. データの種類に応じて、Mule ランタイムまたは API マネージャーから取得されます。

D. デプロイメントモデルに関係なく、Mule ランタイムから
Answer: D (メーッセージを残す**)**
正解: デプロイメントモデルに関係なく、Mule ランタイムから
****
>> モニタリングとアラートのメトリックは、デプロイメント モデルに関係なく、常に Mule ランタイムから生成されます。
>> 一部のメトリクス（Runtime Manager）はMule Runtimeから生成され、一部のメトリクス（API呼び出し/API分析）はAPI Managerから生成されているように見えるかもしれません。しかし、実際にはそうではありません。API ManagerはAPIインスタンスの管理ツールに過ぎませんが、APIに適用されたすべてのポリシーは最終的にMule Runtime（組み込みまたはAPIプロキシ）でのみ実行されるためです。
>> 同様に、すべての API 実装も Mule ランタイム上で実行されます。
したがって、デプロイメント モデルが MuleSoft ホスト型、顧客ホスト型、またはハイブリッド型であるかどうかに関係なく、監視とアラートに必要な 1 日の大半は Mule Runtimes からのみ生成されます。

最新問題**: 35**
ヨーロッパ全土に顧客を持つあるヨーロッパ企業のIT部門は、旧プラットフォームからMuleSoftへの移行を進めています。新プラットフォームの主な要件は、ダウンタイムなしでの再デプロイメント、複数のランタイムバージョンへのアプリケーションのデプロイメント、セキュリティとスピードの確保、そしてメッセージサービスとしてAnypoint MQの利用です。
追加のネットワーク構成なしで、要件に基づいてどのランタイム プレーンを選択する必要がありますか?
A. ランタイムプレーン用の VM / ベアメタル上のランタイムファブリック
B. 顧客ホストのランタイムプレーン
C. MuleSoft がホストするランタイム プレーン (CloudHub)
D. ランタイムプレーン用のセルフマネージドKubernetes上のAnypoint Runtime Fabric
Answer: C (メーッセージを残す**)**
ゼロダウンタイムの再デプロイメント、複数のランタイム バージョンへのデプロイメント、安全で高速なパフォーマンス、追加のネットワーク構成なしでの Anypoint MQ の使用などの要件を持つヨーロッパの企業にとって、次の理由から CloudHub が最適な選択肢です。
ゼロダウンタイムの再デプロイメント: CloudHub はゼロダウンタイムのデプロイメントをサポートしており、可用性に影響を与えることなくアプリケーションをシームレスに再デプロイメントできます。
複数のランタイム バージョンのサポート: CloudHub を使用すると、さまざまな Mule ランタイム バージョンにわたってアプリケーションをデプロイできるため、必要に応じてアプリケーションをテストおよび移行する柔軟性が得られます。
統合されたAnypoint MQ :CloudHubと完全に統合されたAnypoint MQは、アプリケーション間で信頼性の高いメッセージングを提供します。CloudHubを選択すると、このホスト環境からAnypoint MQに直接アクセスできるため、追加のネットワーク設定は不要になります。
セキュリティとパフォーマンス :CloudHubは、複雑な設定を必要とせず、安全なネットワーク、自動スケーリング、最適化されたパフォーマンスを提供します。これらはMuleSoftのインフラストラクチャによって管理され、最小限のオーバーヘッドで速度とセキュリティの要件を満たします。
誤った選択肢:

オプション A および D (VM 上の Runtime Fabric またはセルフマネージド Kubernetes): Runtime Fabric は柔軟性を提供しますが、より複雑なネットワークとインフラストラクチャの構成が必要になるため、企業がシンプルさを求めている場合は理想的ではありません。
オプション B (顧客ホスト): この場合、追加のネットワークおよびセキュリティ構成が必要になりますが、セットアップの複雑さを最小限に抑えるという要件には一致しません。
参照
ゼロダウンタイムのデプロイメントと Anypoint MQ との統合に関する CloudHub の機能の詳細については、CloudHub の MuleSoft ドキュメントを参照してください。

最新問題: **36**
大手融資会社が、データベースサーバーとウェブサーバーからデータを取得するためのAPIを開発しました。このAPIは、CloudHub 1.0上のAnypoint仮想プライベートクラウド（VPC）にデプロイされています。
データベースサーバーとウェブサーバーはお客様のセキュアネットワーク内に設置されており、パブリックインターネットからはアクセスできません。データベースサーバーはお客様のAWS VPC内に設置され、ウェブサーバーはお客様のオンプレミスの企業データセンター内に設置されています。
API がデータベース サーバーおよび Web サーバーに接続できるようにアクセスを有効にするにはどうすればよいでしょうか?
A. AWS VPC との VPC ピアリングと、顧客のオンプレミス企業データセンターへの VPN トンネルを設定します。
B. AWS VPCと顧客のオンプレミス企業データセンターとのVPCピアリングを設定する
C. AWS VPC を介して顧客のオンプレミス企業データセンターへのトランジットゲートウェイをセットアップする
D. 顧客のオンプレミス企業データセンターとのVPCピアリングとAWS VPCへのVPNトンネルを設定します。
Answer: (解答を表示する**)**
シナリオの概要:
API は CloudHub 1.0 上の Anypoint Virtual Private Cloud (VPC) に常駐し、AWS がホストするデータベース サーバーとオンプレミスの Web サーバーの両方への接続が必要です。
両方のサーバーはパブリックインターネットから分離されています。データベース サーバーは顧客の AWS VPC 内にあり、Web サーバーは顧客のオンプレミスの企業データセンター内にあります。
接続要件:
Anypoint VPC の API から AWS データベースサーバーに接続するには、VPC ピアリングが最適です。これにより、MuleSoft Anypoint VPC とお客様の AWS VPC 間のプライベートネットワーク接続が可能になり、データベースへの安全で直接的なアクセスが可能になります。
オンプレミスのウェブサーバーへの接続には、VPNトンネルが適しています。これにより、Anypoint VPCからお客様の企業データセンターへの安全で暗号化された接続が確立され、APIとオンプレミスのウェブサーバー間の安全なデータフローが実現します。
オプションの分析:
オプションA（正解）AWS VPCとのVPCピアリングを設定することで、データベースサーバーとのプライベートネットワーク接続が可能になり、オンプレミスデータセンターへのVPNトンネルによってウェブサーバーへの安全なアクセスが可能になります。この組み合わせは、両方のリソースへの安全かつ制御されたアクセスの要件を満たします。
オプションB :VPCピアリングだけでは不十分です。Anypoint VPCからオンプレミスネットワークへの直接接続をサポートしていないためです。オンプレミスアクセスにはVPNが必要です。

オプション C: トランジットゲートウェイを設定すると、AWS 内での接続は可能になりますが、CloudHub からオンプレミスネットワークへの直接接続は有効になりません。
オプション D: VPC ピアリングは通常、VPC とオンプレミス ネットワークを接続するのではなく、2 つの VPC を接続するために使用されるため、オンプレミス ネットワークとの VPC ピアリングは不可能です。
結論：
オプションAは正しい選択です。AWS VPC接続にはVPCピアリングを使用し、オンプレミスへの安全な接続にはVPNトンネルを使用することで、包括的なソリューションを提供します。この設定は、Anypoint VPCをAWSホストシステムとオンプレミスシステムの両方に接続するためのAnypoint Platformのベストプラクティスに準拠しており、データベースとWebサーバーの両方への安全で制御されたアクセスを保証します。
さらに詳しい情報については、Anypoint VPC ピアリングと VPN 接続に関する MuleSoft のドキュメントを参照してください。ハイブリッド ネットワーク インフラストラクチャ内でこれらの接続を設定するためのベスト プラクティスに関する追加のコンテキストが提供されています。

最新問題**: 37**
一部のHTTPリクエストへのレスポンスは、リクエストで使用されたHTTP動詞に応じてキャッシュされることがあります。HTTP仕様によると、どのHTTP動詞に対してキャッシュしても安全でしょうか？
A. PUT、POST、DELETE
B. GET、HEAD、POST
C. GET、PUT、オプション
D. GET、オプション、HEAD
Answer: D (メーッセージを残す**)**
正解: GET、OPTIONS、HEAD
http://restcookbook.com/HTTP%20Methods/idempotency/

最新問題**: 38**
あるeコマース企業が、ウェブサイトに新しい商品詳細機能を追加しています。顧客が商品カタログページを開くと、商品ごとに新しい商品詳細リンクが表示され、クリックすることで商品の詳細説明を取得できます。商品詳細データは、年に1～2回の製品アップデートリリースに合わせて更新されますが、現在、アクセス数の増加によりデータベースの応答時間が非常に遅くなっています。
応答時間が最も短く、フォールト トレランスと一貫性のあるデータで製品の詳細を取得するアクションは何ですか。
A. キャッシュスコープ内のデータベースから製品の詳細を選択し、APIレスポンス内で返します。
B. データベースから製品の詳細を選択し、Anypoint MQに格納します。Anypoint MOサブサーバは製品の詳細を受け取り、APIレスポンスで返します。
C. オブジェクトストアを使用して、データベースから読み取った製品の詳細を保存および取得し、APIレスポンス内で返します。
D. データベースから製品の詳細を選択し、APIレスポンス内で返します。
Answer: C (メーッセージを残す**)**
シナリオ分析:
電子商取引会社の製品詳細機能では、データがほとんど変更されない (年に 1 回か 2 回のみ) ため、応答時間が短く、データの一貫性が求められます。

データベースの応答時間はボリュームが大きいために遅くなります。そのため、リクエストごとにデータベースを直接クエリすると、パフォーマンスが低下し、応答時間が長くなります。
最適なソリューションの要件:
応答時間が短い: データの取得は高速で、データベースのパフォーマンスに依存しません。
フォールト トレランスとデータの一貫性: 製品の詳細データは頻繁に変更されないため、キャッシュまたは保存されたデータは、データベースが利用できない場合でも一貫性と回復力を備えている必要があります。
オプションの評価:
オプション A: キャッシュ スコープを使用すると、製品の詳細が一時的にメモリに保存され、パフォーマンスが向上する可能性がありますが、キャッシュの有効期限ポリシーでは通常、より短い期間が要求されるため、頻度の低い更新 (年に 2 回のみ) には適さない可能性があります。
オプションB :Anypoint MQに製品の詳細を保存し、サブスクライバーを介して取得する方法は、このユースケースには適していません。Anypoint MQは、データストレージメカニズムとしてよりも、メッセージングに適しています。
オプションC（正解）：製品詳細保存と取得にはオブジェクトストアを使用するのが理想的です。MuleSoftのオブジェクトストアは、キーと値のペアを永続的に保存するように設計されており、データベースから最初に取得したデータを保存できます。これにより、リクエストごとにデータベースにクエリを実行することなく、迅速かつ一貫したアクセスが可能になり、応答時間の短縮、フォールトトレランス、データの一貫性といった要件を満たします。
オプション D: 各リクエストに対してデータベースから直接データを選択すると、データベースからの応答時間が遅いことが知られているため、パフォーマンス要件を満たせません。
結論：
オプションCが最適な回答です。オブジェクトストアを使用することで、更新頻度の低い製品情報をキャッシュできます。このアプローチにより、データベースへの依存度が低減され、応答時間が大幅に改善され、データの一貫性が確保されます。
このソリューションを効果的に実装するには、Object Store v2 に関する MuleSoft のドキュメントとデータ キャッシュのベスト プラクティスを参照してください。

最新問題: **39**
Mule アプリケーションのデプロイメントを自動化する 3 つのツールはどれですか?
3つの回答を選択してください
A. ランタイムマネージャー
B. Anypoint Platform CLI
C. プラットフォームAPI
D. エニーポイントスタジオ
E. Mule Mayen プラグイン
F. API コミュニティ マネージャー
Answer: (解答を表示する**)**
MuleSoftは、Muleアプリケーションのデプロイメントを自動化するための様々なツールを提供しており、デプロイメントと管理プロセスを効率化できます。各ツールがどのように自動デプロイメントをサポートするかを以下に示します。
ランタイムマネージャー:
Anypoint Runtime Managerは、MuleSoftのWebベースインターフェースで、ユーザーがアプリケーションを直接デプロイ、管理、監視できます。ユーザーフレンドリーなインターフェースを通じて、デプロイの自動化を実現します。
Anypoint プラットフォーム CLI:

Anypoint CLI は、デプロイメントと管理タスクのスクリプト化を可能にし、コマンドラインスクリプトによるデプロイメントの自動化を可能にします。このツールは自動化プロセスと統合できるため、CI/CD パイプラインに最適です。
プラットフォーム API:
MuleSoftのプラットフォームAPIは、デプロイメント機能へのプログラム的なアクセスを可能にし、外部の自動化ツールやCI/CDシステムとの統合を可能にします。これらのAPIは、RESTful呼び出しによるデプロイメントを容易にし、継続的デリバリーの自動化を可能にします。
誤った選択肢:
オプション D (Anypoint Studio) は主に開発用であり、デプロイメントの自動化はサポートされていません。
オプション E (Maven プラグイン) は、Mule アプリケーションの構築とデプロイに使用できますが、デプロイ用のプラットフォームツールとしては分類されません。
オプション F (API コミュニティ マネージャー) はデプロイメントとは関係なく、API コミュニティの管理に重点を置いています。
参照
これらのツールを使用してデプロイメントを自動化する詳細な手順については、Runtime Manager、CLI、および Platform API に関する MuleSoft のドキュメントを参照してください。

最新問題**: 40**
API 主導の接続性と呼ばれるフレームワーク内で作成された API の機能ではないものは何ですか?
A. 基盤となるバックエンド システムの上に追加の耐障害性レイヤーを提供し、それによってクライアントをこれらのシステムの長期的な障害から保護します。
B. バックエンド システムからデータがどのように抽出されるかを意識せずに、基礎となる資産を消費することで、ユーザー インターフェイス レベルでのイノベーションを可能にします。
C. バックエンド システムからデータを再利用可能かつ消費可能な方法でロック解除できるようにすることで、基盤となるバックエンド システムへの依存を軽減します。
D. さまざまなソースからデータを作成し、オーケストレーション ロジックと組み合わせて、より高いレベルの価値を生み出すことができます。
Answer: A (メーッセージを残す**)**
正解: 基盤となるバックエンド システムの上に回復力の追加レイヤーを提供し、それによってクライアントをこれらのシステムの長期的な障害から保護します。
****
API主導の接続性では、
>> エクスペリエンス API - バックエンド システムからデータがどのように抽出されるかを意識せずに、基盤となるアセットを消費することで、ユーザー インターフェイス レベルでのイノベーションを可能にします。
>> プロセスAPI - さまざまなソースからデータを作成し、オーケストレーションロジックと組み合わせて、より高いレベルの価値を生み出します
>> システム API - バックエンド システムからデータを再利用可能かつ消費可能な方法でロック解除できるようにすることで、基盤となるバックエンド システムへの依存を軽減します。
しかし、基盤となるバックエンド システムの上に追加の回復力レイヤーを提供し、それによってクライアントをこれらのシステムの長期的な障害から保護することを決して約束しません。
https://dzone.com/articles/api-led-connectivity-with-mule

最新問題**: 41**
次の順序のうち正しいものはどれですか?
A. APIクライアントはAPIを呼び出すロジックを実装します >> APIコンシューマーはAPIへのアクセスを要求します >> API実装はリクエストをAPIにルーティングします
B. APIコンシューマがAPIへのアクセスを要求 >> APIクライアントがAPIを呼び出すロジックを実装 >> APIがリクエストをルーティング >> API実装
C. API コンシューマーは API を呼び出すロジックを実装します >> API クライアントは API へのアクセスを要求します >> API 実装はリクエストを API にルーティングします
D. APIクライアントはAPIを呼び出すロジックを実装します >> APIコンシューマーはAPIへのアクセスを要求します >> APIはリクエストをルーティングします >> API実装
Answer: B (メーッセージを残す**)**
正解: API コンシューマーが API へのアクセスを要求 >> API クライアントが API を呼び出すロジックを実装 >> API がリクエストをルーティング >> API 実装
****
>> APIコンシューマーはAPIを呼び出すロジックを実装しません。単なるロールです。したがって、APIコンシューマーはAPIを呼び出すロジックを実装します」というオプションは無効です。
>> API実装はリクエストをルーティングしません。これは、対象システムの機能が公開されるロジックの最終段階です。したがって、リクエストは別のエンティティによってAPI実装にルーティングされる必要があります。したがって、API実装がリクエストをAPIにルーティングする」というオプションは無効です。
>> 選択肢の1つの記述は正しいですが、順序が間違っています。順序は「APIクライアントがAPIを呼び出すロジックを実装 >> APIコンシューマーがAPIへのアクセスを要求 >> APIがリクエストをAPI実装にルーティング」となっています。選択肢の記述は正しいのですが、順序が間違っています。
>> 正しいオプションとシーケンスは、APIコンシューマーがまずAnypoint Exchange上のAPIへのアクセスを要求し、クライアント資格情報を取得するというものです。次に、APIクライアントは、APIコンシューマーが要求したアクセスクライアント資格情報を使用してAPIを呼び出すロジックを記述し、そのリクエストはAPI Managerによって管理されるAPIを介してAPI実装にルーティングされます。

最新問題**: 42**
プラットフォームアーキテクトは、クライアントに属するすべての保険証券の表示など、様々なタスクを実行する、レガシーなモノリシックなSOAPベースのWebサービスを継承しています。このサービスは、生命保険管理システムと損害保険管理システムという2つのバックエンドシステムに接続し、各システム内で保険証券情報を照会し、結果を集約して、SOAPベースの応答をユーザーインターフェース（UI）に表示します。
アーキテクトは、API 主導の規則に従うためにモノリシック Web サービスを分割したいと考えています。
サービスのどの部分をプロセス層に配置する必要がありますか?
A. 管理システムからのデータのクエリ
B. UI に SOAP ベースの応答を表示する
C. 各バックエンド管理システムへの接続を認証および維持する
D. 管理システムからの保険契約情報を統合する

Answer: D (メーッセージを残す)

最新問題**: 43**
ある企業は、データセンター内のオンプレミス クラスターをランタイム プレーンと MuleSoft がホストするコントロール プレーンとして使用しています。
コントロールプレーンからクラスターにデプロイされた Mule アプリケーションの詳細なパフォーマンス メトリックをどのように監視できますか?
A. キャプチャするメトリックの詳細なログを有効にするには、コントロールプレーンの監視セクションの設定を更新する必要があります。
B. クラスタ内の各ノードに監視エージェントをインストールする必要があります
C. ランタイムノードにパフォーマンスに影響を与える可能性があるため、監視エージェントは別のサーバーにインストールする必要があります。
D. オンプレミスのランタイムがパフォーマンスデータをコントロールプレーンに自動的に送信するため、アクションは必要ありません。
Answer: (解答を表示する)
オンプレミス Mule アプリケーションの監視:
オンプレミスクラスタにデプロイされた Mule アプリケーションの場合、詳細なパフォーマンスメトリクスを監視するには、MuleSoft がホストするコントロールプレーンとの通信が必要です。オンプレミスランタイムで使用する場合、コントロールプレーンは Anypoint Monitoring に依存しており、詳細なパフォーマンスメトリクスを収集して送信するには Monitoring Agent が必要です。
監視の設定:
詳細なメトリクスを有効にするには、Mule アプリケーションがデプロイされているクラスタ内の各ノードにモニタリングエージェントをインストールする必要があります。このエージェントは、メモリ使用量、CPU 負荷、応答時間などのメトリクスに関するデータを収集し、コントロールプレーンに送信して集計および可視化します。
オプションの評価:
オプション A: コントロール プレーンの設定を更新するだけでは詳細な監視は有効になりません。詳細なメトリックを取得するには、各ノードにエージェントをインストールする必要があります。
オプション B (正解): 各ノードにモニタリング エージェントをインストールすると、クラスター内の各ランタイム ノードがメトリックをコントロール プレーンに送信できるようになり、詳細なモニタリングが可能になります。
オプション C: 完全な可視性を確保するために、クラスター内の各ノードが個別にメトリックを報告する必要があるため、エージェントを別のサーバーにインストールすることは効果的ではありません。
オプション D: オンプレミスのランタイムは、モニタリング エージェントがインストールされていない場合、詳細なメトリックをコントロール プレーンに自動的に送信しません。
結論：
クラスター内のオンプレミス アプリケーションの詳細なパフォーマンス監視には、各ノードに監視エージェントをインストールすることが不可欠であるため、オプション B が正解です。
オンプレミス展開用の Anypoint Monitoring の構成と Monitoring Agent の使用については、MuleSoft のドキュメントを参照してください。

最新問題**: 44**

あるビジネスプロセスが組織のアプリケーションネットワーク内に実装されています。アーキテクチャグループは、より細分化された設計と比較して、アプリケーションネットワークにデプロイされるAPIの数を比較的少なくし、より粗い粒度のアプリケーションネットワーク設計の使用を提案しています。
全体的に、このビジネス プロセスの実装と展開において、よりきめ細かい設計を使用する場合と比較して、より粗い設計を使用すると、通常どの要因が増加しますか。
A. 各API実装の複雑さ
B. アプリケーションネットワークに展開されたAPIに関連する検出可能な資産の数
C. アプリケーションネットワーク内のAPI実装間の可能な接続数
D. アプリケーションネットワークによるネットワークインフラストラクチャリソースの使用状況
Answer: A (メーッセージを残す**)**
粗粒度と細粒度の API 設計の違いを理解する:
粗粒度設計では、複数の操作を単一のAPIに統合するため、APIの数は少なくなりますが、実装は複雑になります。一方、細粒度設計では、機能をより小さく、より具体的なAPIに分割するため、実装はシンプルになりますが、APIの数は多くなります。
オプションの評価:
オプション A (正解): 粗粒度設計では、各 API が処理する機能が増え、より多くのユースケースとロジックを管理する必要があるため、各 API 実装の複雑さが増します。
オプション B: 粗粒度設計では通常、API の数が減るため、検出可能なアセットの数が少なくなります。
オプション C: API が少ないということは、通常、アプリケーション ネットワーク内での API 間の接続が少なくなることを意味します。
オプション D: API 間の呼び出しが少なくなるため、API が少なくなるとネットワーク インフラストラクチャの使用量が実際に減少する可能性があります。
結論：
オプション A が正解です。粗粒度設計では、複数の機能が単一の API に統合されるため、各 API 実装の複雑さが増します。
API 設計原則と、粗粒度と細粒度の API 実装のベスト プラクティスについては、MuleSoft のドキュメントを参照してください。

最新問題**: 45**
チームはエクスペリエンス API 仕様の拡張を計画しており、API 主導の接続設計原則に従っています。
API を強化する動機は何ですか?
A. 主要なAPIコンシューマーは、特定の種類のエンドポイントをCenter for Enablement標準からコンシューマーシステム標準に変更することを望んでいます。
B. 基盤となるシステムAPIが更新され、頻繁に使用されるいくつかのリソースについてより詳細なデータを提供するようになりました。
C. 開発環境とステージング環境のAPIインスタンスにIP許可リストポリシーが追加されています
D. APIに含まれるいくつかの種類のデータに影響を与える標準データモデルが採用されています。
Answer: D (メーッセージを残す**)**
API主導設計では、エクスペリエンスAPIが拡張され、エンドユーザーアプリケーションへのデータ配信方法が改善されます。エクスペリエンスAPIを拡張する主な理由の一つは、標準データモデルなどの新しいデータ標準が採用された場合です。その理由は以下のとおりです。
標準データモデル (CDM):

CDM を導入すると、組織全体のデータ表現が標準化され、API の一貫性が高まり、さまざまなサービスやアプリケーションで使いやすくなります。
Experience API を更新すると、この標準化された形式でデータが提供されるようになり、相互運用性と再利用性が向上します。
正解 D）
CDM は、API が提供するデータの構造とタイプに影響を与えます。また、この更新は、アプリケーションの主要なインタラクションポイントである Experience API に直接関係します。
誤った選択肢:
オプション A では、消費者固有の標準に適応することになり、これは API 主導の設計原則に反します。
オプション B にはシステム API の変更が含まれますが、データ形式の調整が必要な場合を除き、エクスペリエンス API への変更は直接的には必須ではありません。
オプション C (IP 許可リスト) は、API 設計ではなくセキュリティに関連し、API の機能強化を促すものではありません。
参照
API 主導のアーキテクチャでの標準データ モデルの使用の詳細については、MuleSoft のデータ標準化に関するガイドラインと Experience API のベスト プラクティスを参照してください。

最新問題**: 46**
週に数回、API実装のAnypoint Monitoringダッシュボードに1分あたり数千件のリクエストが表示されることがあります。これらのバーストの間には、ダッシュボードには1分あたり2〜5件のリクエストが表示されます。API実装は、2つの非クラスタレプリカ（予約済みvCPU 1.0とvCPU Limit 2.0）を備えたAnypoint Runtime Fabric上で実行されています。
API利用者から応答時間の遅さについて苦情があり、ダッシュボードでは苦情発生時の99パーセンタイルが120秒を超えていることが示されています。また、この時間帯のCPU使用率は90%を超えています。
QA環境での手動テストでは、APIコンシューマーは一貫して遅い応答時間と高いCPU使用率を再現しており、この時点では他のAPIリクエストはありませんでした。ブレインストーミングセッションにおいて、エンジニアリングチームはリクエストの応答時間を短縮するためのいくつかの提案を作成しました。
どの提案を最初に追求すべきでしょうか?
A. API実装のvCPUリソースを増やす
B. APIクライアントを変更して、問題のあるリクエストをより小さく、要求の少ないリクエストに分割します。
C. API実装のレプリカ数を増やす
D. APTクライアントをスロットルして、1分あたりのリクエスト数を減らします。
Answer: A (メーッセージを残す**)**
シナリオ分析:
API 実装では、リクエストのバースト時に CPU 使用率が高くなります (90% 以上)。これは、99 パーセンタイル応答時間が 120 秒を超えていることで示されるように、応答時間が遅くなることに関連しています。
API 実装は、2 つの非クラスター化レプリカを持つ Anypoint Runtime Fabric 上で実行されており、予約済みの vCPU は 1.0、vCPU 制限は 2.0 です。
バースト時の CPU 使用率が高い場合、現在のリソースがピーク負荷を処理するのに十分でない可能性があります。
オプションの評価:

オプションA（正解） :各プリカのvCPUリソースを増やすことで、高トラフィック量を処理するための処理能力が向上し、スパイク時の応答時間を短縮できる可能性があります。バースト時にはCPU使用率が常に高くなるため、このオプションはリソースのボトルネックに直接対処します。
オプション B: API クライアントを変更してリクエストを分割すると、個々のリクエストの負荷が軽減される可能性がありますが、クライアント側での実装が複雑になり、CPU 使用率の高い問題が完全に解決されない可能性があります。
オプション C: レプリカの数を増やすと負荷を分散できる可能性があります。ただし、各レプリカの CPU 負荷が高い場合、CPU リソースを増やさずにレプリカを追加しても、問題が完全に解決されない可能性があります。
オプションD :クライアントのスロットリングによってリクエスト数は減少しますが、クライアントが高リクエストレートを維持する必要がある場合は、この方法は適切ではない可能性があります。また、API実装のCPU制限に直接対処するものではありません。
結論：
オプションAは、vCPUの割り当てを増やすことでCPU使用率の上昇の根本原因に対処し、APIがより多くのリクエストを効率的に処理できるようにするため、最適な選択肢です。他のオプションを検討する前に、まずはこれを検討する必要があります。
需要の高い環境での API パフォーマンスの最適化の詳細については、MuleSoft の Runtime Fabric と vCPU リソース割り当てに関するドキュメントを参照してください。

最新問題: **47**
Anypoint Exchange では、API プロデューサーによって、承認されたセマンティック バージョン管理の慣行に従って API がバージョン 3.1.1 から 3.2.0 に更新され、その変更は API のパブリック ポータルを通じて通知されました。
新しいバージョンでは API エンドポイントは変更されません。
API クライアントの開発者はこの変更にどのように対応すべきでしょうか?
A. APIクライアントコードは、新しい機能を利用する必要がある場合にのみ変更する必要があります。
B. 更新はプロジェクトリスクとして識別され、このAPIを使用する機能の完全な回帰テストを実行する必要があります。
C. APIプロデューサーは、新しいバージョンと並行して古いバージョンを実行するように要求される必要があります。
D. 既存の機能の変更を理解するために、APIプロデューサーに連絡する必要があります。
Answer: A (メーッセージを残す**)**

最新問題: **48**
システムAPIはプライマリ環境と災害復旧（DR）環境にデプロイされており、各環境で異なるDNS名が使用されています。プロセスAPIはシステムAPIのクライアントであり、システムAPIによってレート制限を受けており、各環境で異なる制限が適用されます。システムAPIのDR環境では、プライマリ環境が提供するレート制限の20%しか提供されません。これらの条件と制約を考慮した場合、プロセスAPIの全体的なエラーを削減するための最適なAPIフォールトトレラント呼び出し戦略は何でしょうか？
A. プライマリ環境にデプロイされたシステム API を呼び出します。断続的な障害を回避するために、プロセス API にタイムアウトと再試行のロジックを追加します。それでも障害が発生する場合は、DR 環境にデプロイされたシステム API を呼び出します。

B. プライマリ環境にデプロイされたシステム API を呼び出します。DR 環境にデプロイされたシステム API を呼び出すことで、断続的な障害を処理するためにプロセス API に再試行ロジックを追加します。
C. プライマリ環境にデプロイされたシステムAPIとDR環境にデプロイされたシステムAPIを並行して呼び出し、断続的な障害を回避するためにプロセスAPIにタイムアウトと再試行のロジックを追加し、結果を結合するためのロジックをプロセスAPIに追加します。
D. プライマリ環境にデプロイされたシステム API を呼び出します。断続的な障害を回避するために、プロセス API にタイムアウトと再試行ロジックを追加します。それでも失敗する場合は、DR 環境にデプロイされたプロセス API のコピーを呼び出します。
Answer: (解答を表示する**)**
正解: プライマリ環境にデプロイされたシステム API を呼び出します。断続的な障害を回避するために、プロセス API にタイムアウトと再試行のロジックを追加します。それでも障害が発生する場合は、DR 環境にデプロイされたシステム API を呼び出します。
****
この質問には、DR環境のシステムAPIがプライマリ環境が提供するレート制限の20%しか提供しないという重要な考慮事項が1つあります。つまり、DR環境のAPIへの呼び出しは、プライマリ環境に比べて非常に少なくなります。この点を念頭に置き、適切かつ最適なフォールトトレラントな呼び出し戦略を分析してみましょう。
1. DR環境には20%という制限があるため、両方のシステムAPIを並列に呼び出すことは絶対に現実的なアプローチではありません。毎回並列に呼び出すと、DR環境のレート制限をあっという間に使い果たしてしまい、必要な時に真に断続的なエラーシナリオを許容する機会を失ってしまう可能性があります。
2. もう一つの選択肢として、プライマリ環境のシステムAPIを呼び出す際に、プロセスAPIにタイムアウトと再試行ロジックを追加するというものがあります。これは今のところ良いアイデアです。しかし、すべての再試行が失敗した場合、DR環境のプロセスAPIのコピーを呼び出すという選択肢がありますが、これは適切ではなく、推奨もされません。フォールバックの対象とすべきはシステムAPIのみであり、プロセスAPI全体ではありません。プロセスAPIは通常、他の多くのAPIを呼び出すための高度なオーケストレーションを多く備えており、DRのプロセスAPIを呼び出すことでこれらのAPIを再度実行することは望ましくありません。したがって、このオプションは適切ではありません。
3. もう1つの選択肢として、プロセスAPIに再試行（タイムアウトなし）ロジックを追加し、プライマリ環境のシステムAPIを最初に再試行するのではなく、DR環境のシステムAPIを直接再試行するという提案があります。これは全く適切なフォールバックではありません。適切なフォールバックは、プライマリ環境ですべての再試行が実行され、それが使い果たされた後にのみ発生するべきです。しかし、ここでは、最初の障害発生時にメインAPIを試行せずにフォールバックAPIを直接再試行することを提案しています。したがって、このオプションも正しくありません。
これにより、適切かつ最適なオプションが 1 つ残ります。
- プライマリ環境にデプロイされたシステムAPIを呼び出す
- プロセス API にタイムアウトと再試行のロジックを追加します。
- すべての再試行後も失敗した場合は、DR 環境にデプロイされたシステム API を呼び出します。

最新問題**: 49**
Anypoint Platform の API ポリシーを使用して効果的に適用できないものは何ですか?
A. サービス拒否攻撃に対する防御
B. API間の改ざん防止認証情報の維持
C. HTTPリクエストとレスポンスのログ記録
D. バックエンドシステムの過負荷
Answer: A (メーッセージを残す**)**

正解: サービス拒否攻撃に対する防御
****
>> バックエンドシステムの過負荷は、「スパイク制御ポリシー」を適用することで対処できます
>> HTTPリクエストとレスポンスのログ記録は、「メッセージログポリシー」を適用することで実行できます。
>> 認証情報は、「セキュリティ」および「コンプライアンス」ポリシーを使用して改ざん防止できます。ただし、残念ながら、現在Anypoint Platform には DOS 攻撃から保護するための適切な方法はありません。

最新問題: **50**
4 ある運輸組織の開発者は、乗客記録を処理・保存するための処理機能を1つだけ予約Muleアプリケーションに実装しています。この予約アプリケーションは、複数のCloudHubワーカー／レプリカにデプロイされます。複数の外部システムから予約アプリケーションに重複した乗客記録が送信される可能性があります。
予約アプリケーションが各乗客レコードを可能な限り正確に1回処理できるよう、適切なストレージメカニズムを選択する必要があります。選択されたストレージメカニズムは、デプロイされた予約ミュールアプリケーションが各乗客レコードを1回だけ処理できるよう、状態情報を同期するために、すべてのCloudHubワーカー/レプリカで共有される必要があります。
Anypoint Platform のどのタイプのシンプルなストレージ メカニズムにより、Reservation Mule アプリケーションは最小限の開発労力で、CloudHub ワーカー/レプリカ間でデータを 1 回だけ更新および共有できるようになりますか?
A. 永続オブジェクトストア
B. ランタイムファブリックオブジェクトストア
C. 非永続オブジェクトストア
D. インメモリ Mule オブジェクト ストア
Answer: A (メーッセージを残す**)**
処理要件と保存メカニズム:
Reservation Mule アプリケーションは複数の CloudHub ワーカー／レプリカにデプロイされるため、各ワーカーはレコードを一度だけ処理するために状態情報を共有する必要があります。そのため、同じレコードの重複処理を回避するために、複数のインスタンスから状態を保存・アクセスできる共有ストレージメカニズムが必要です。
Anypoint Platform の永続オブジェクト ストアを使用すると、複数のワーカー間でアクセス可能な方法でレコードを保存でき、「正確に 1 回」の処理を実行するための信頼性の高いメカニズムが提供されます。
オプションの評価:
オプションA（正解）：永続オブジェクトストアは、異なるアプリケーションインスタンス間でデータを保持し、CloudHub上のすべてのワーカーで共有できるように設計されており、レコードが正確に1回処理されることを保証することで、冪等性を実現します。
オプションB：CloudHubではなくAnypoint Runtime Fabricにデプロイされたアプリケーションには、Runtime Fabricオブジェクトストアが使用されます。このオプションはCloudHubのデプロイメントとは互換性がありません。
オプション C: 非永続オブジェクト ストアでは、アプリケーションの再起動や異なるインスタンス間でデータが保持されないため、正確に 1 回の処理のための同期ストレージの要件には適していません。
オプション D: インメモリ Mule オブジェクト ストアは各ワーカーに対してローカルであり、インスタンス間で共有されないため、すべての CloudHub ワーカーがアクセスできる共有ストレージ メカニズムの要件を満たしません。
結論：
オプション A が正解です。永続オブジェクト ストアを使用すると、複数の CloudHub ワーカー間でデータを共有できるため、最小限の開発労力で乗客レコードを同期して「正確に 1 回」処理できます。

分散インスタンス間で状態を処理するためのベスト プラクティスについては、MuleSoft のオブジェクト ストアの構成と使用に関するドキュメントを参照してください。

最新問題**: 51**
Anypoint Platform で API ポリシーが定義される場所と、それが API インスタンスにどのように適用されるかについて正しいのはどれですか。
A. API ポリシーは、Mule ランタイムへの API デプロイメントの一部として Runtime Manager で定義され、特定の API インスタンスにのみ適用されます。
B. API ポリシーは特定の API インスタンスに対して API Manager で定義され、特定の API インスタンスにのみ適用されます。
C. APIポリシーはAPI Managerで定義され、すべてのAPIインスタンスに自動的に適用されます。
D. APIポリシーはAPI Managerで定義され、指定された環境内のすべてのAPIインスタンスに適用されます。
Answer: B (メーッセージを残す**)**
正解: API ポリシーは、特定の API インスタンスに対して API Manager で定義され、その特定の API インスタンスにのみ適用されます。
****
>> API 仕様が準備され、Exchange に公開されたら、API Manager にアクセスして、各 API の API インスタンスを登録する必要があります。
>> API マネージャーは、ポリシーを適用して NFR に対処するなど、API の側面の管理が行われる場所です。
>> 同じ API に対して複数のインスタンスを作成し、目的に応じて異なる方法で管理できます。
>> 1 つのインスタンスに API ポリシー セットを適用し、同じ API の別のインスタンスに他の目的で異なるポリシー セットを適用することができます。
>> これらのAPIとそのインスタンスは環境ごとに定義されます。そのため、各環境で個別に管理する必要があります。
>> プラットフォーム機能を使用して上位環境に昇格する際に、APIインスタンスの設定 $LA、ポリシーなど）が常に同じになるように設定できます。ただし、これはあくまでオプションです。必要に応じて、環境ごとに設定を変更することも可能です。
>> ランタイムマネージャは、API実装とそのMuleランタイムを管理する場所であり、API自体を管理する場所ではありません。APIポリシーはMuleランタイムで実行されますが、ランタイムマネージャでAPIポリシーを適用することはできません。環境内の厳選されたインスタンスに対してのみ、APIマネージャを介してポリシーを適用する必要があります。
したがって、これらの事実に基づくと、与えられた選択肢の中で正しい記述は、「API ポリシーは特定の API インスタンスに対して API Manager で定義され、その特定の API インスタンスにのみ適用されます」です。

最新問題**: 52**
コード中心の API ドキュメント環境では、API コンシューマーが、代表的なシナリオの一部として 1 つ以上の API の呼び出しを示す API クライアント ソース コードを調査および実行できるようにする必要があります。
Anypoint Platform を使用してこのようなコード中心の API ドキュメント環境を提供する最も効果的な方法は何ですか?
A. 関連するAPIごとにモックサービスを有効にし、Anypoint Exchangeエントリ経由で公開します。
B. APIがAnypoint ExchangeエントリとAPIコンソールを通じて適切に文書化されていることを確認し、これらのページをすべてのAPIコンシューマーと共有します。
C. APIノートブックを作成し、関連するAnypoint Exchangeエントリに含めます。
D. Anypoint Exchange エントリを介して関連する API を検出可能にする

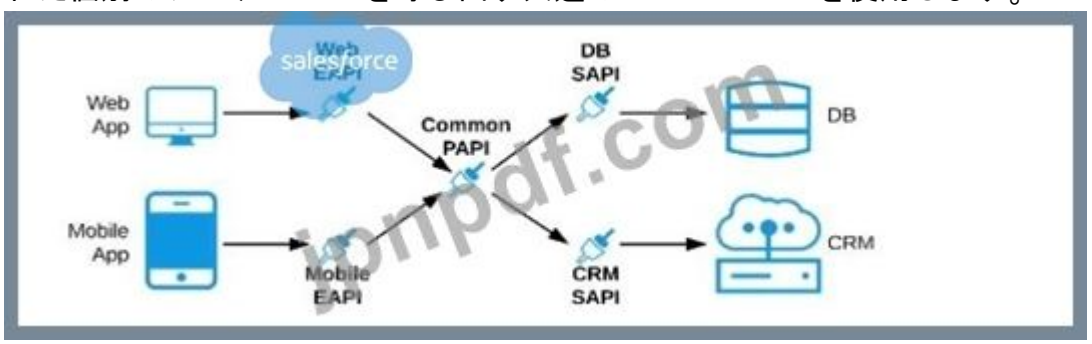
Answer: C (メーッセージを残す)
正解: API ノートブックを作成し、関連する Anypoint Exchange エントリに含める
****
>> API ノートブックは、コード中心の API ドキュメントを提供できる Anypoint Platform 上のものです。参考:
フォームの下部
フォームの上部

最新問題**: 53**
ある企業は、EUコントロールプレーンと顧客ホストのMuleランタイムを組み合わせたハイブリッドAnypoint Platformデプロイメントモデルを採用しています。ステージング環境でMule API実装のテストに成功した後、Mule API実装は環境固有のプロパティが設定され、本番環境に昇格する必要があります。MuleSoftが推奨するMule API実装の設定方法と、本番環境への昇格の自動化方法を教えてください。
A. API マネージャーのプロパティ タブで Mule API 実装のプロパティを変更し、API マネージャーを使用して Mule API 実装を本番環境に昇格します。
B. API ポリシーを使用してステージング環境にデプロイされた Mule API 実装のプロパティを変更し、別の API ポリシーを使用して Mule API 実装を本番環境にデプロイします。
C. 各環境のプロパティ ファイルを Mule API 実装のデプロイ可能なアーカイブにバンドルし、Anypoint CLI または Anypoint Platform REST API を使用して Mule API 実装を本番環境に昇格します。
D. Anypoint Exchange で Mule API 実装のプロパティを変更し、Runtime Manager を使用して Mule API 実装を本番環境に昇格します。
Answer: (解答を表示する)

最新問題**: 54**
添付資料を参照してください。組織は、モバイルアプリとWebアプリケーションの両方から顧客データにアクセスできるようにする必要があります。これらのアプリケーションは、共通フィールドだけでなく、特定の固有フィールドにもアクセスする必要があります。
データは部分的にデータベースで利用でき、部分的にサードパーティの CRM システムで利用できます。
これらの設計要件に最適な API を作成する必要がありますか?

A) ウェブ アプリとモバイル アプリの両方に必要なデータを含むプロセス API。これにより、これらのアプリケーションはプロセス API を直接呼び出して必要なデータにアクセスできるため、将来的に API を変更することなくフィールドを追加できる柔軟性が得られます。B) ウェブ アプリ用に 1 セットの API (エクスペリエンス API、プロセス API、システム API)、モバイル アプリ用に別のセッ

ト C) モバイル アプリとウェブ アプリ用に別々のエクスペリエンス API を使用しますが、データベースと CRM システム用に作成された個別のシステム API を呼び出す共通のプロセス API を使用します。



D) Webアプリとモバイルアプリの両方で使用される共通のエクスペリエンスAPIですが、データベースやCRMシステムと対話するWebアプリとモバイルアプリには別々のプロセスAPIがあります。

A. オプションA
B. オプションB
C. オプションC
D. オプションD
Answer: (解答を表示する**)**

正解: モバイルとウェブアプリには別々のエクスペリエンス API がありますが、データベースと CRM システム用に作成された別々のシステム API を呼び出す共通のプロセス API があります。

****

MuleSoft の API 主導の接続性について:

>> エクスペリエンス API は、各消費者のニーズとエクスペリエンスに応じて構築する必要があります。
>> プロセス API には、ビジネス機能を実現するためのすべてのオーケストレーション ロジックが含まれている必要があります。
>> 各バックエンド システムのデータをロック解除するには、システム API を構築する必要があります。

最新問題**: 55**

ある企業は、Runtime Manager を通じて、顧客がホストする Mule ランタイムにデフォルト設定の Mule アプリケーションをデプロイします。各 Mule アプリケーションは、API クライアントに RESTful インターフェースを公開する API 実装です。Mule ランタイムは、MuleSoft がホストするコントロールプレーンによって管理されます。ペイロードは、どの Logger コンポーネントによっても使用されることはありません。

API クライアントが顧客がホストする Mule アプリケーションに HTTP リクエストを送信すると、どのメタデータまたはデータ (ペイロード) が MuleSoft がホストするコントロール プレーンにプッシュされますか?

A. データのみ
B. データなし
C. データとメタデータ
D. メタデータのみ
Answer: (解答を表示する**)**

Mule ランタイムとコントロール プレーン間のデータ フローを理解する:

Mule アプリケーションがお客様ホストの Mule ランタイムにデプロイされている場合、MuleSoft がホストするコントロールプレーン (Anypoint Platform) はこれらのアプリケーションを監視および管理できます。ただし、データのプライバシーとセキュリティ上の理由から、コントロールプレーンは特定の種類の情報のみを収集します。

通常、リクエストとレスポンスに関するメタデータ（ヘッダー、ステータスコード、タイムスタンプなど）のみがMuleSoftがホストするコントロールプレーンに送信されます。実際のペイロードデータは明示的に設定されない限り送信されないため、機密データはお客様のネットワーク内に留まります。
オプションの評価:
オプション A (データのみ): デフォルト構成ではペイロード データ自体がコントロール プレーンに自動的に送信されないため、これは正しくありません。
オプション B (データなし): これも誤りです。ペイロードは送信されませんが、メタデータは収集され、コントロール プレーンに送信されます。
オプション C (データとメタデータ): デフォルトではデータ (ペイロード) がコントロール プレーンに送信されないため、このオプションは正しくありません。
オプション D (正解): デフォルトではメタデータのみが MuleSoft がホストするコントロール プレーンに送信され、顧客がホストするランタイムのセキュリティとデータ プライバシーを優先するという MuleSoft の設計と一致しています。
結論：
オプションDが正解です。デフォルトでは、MuleSoftがホストするコントロールプレーンにはメタデータのみが送信され、ペイロードは送信されません。この構成は、機密データがお客様のホスト環境の外部に漏洩するのを防ぐために設計されています。
詳細については、顧客がホストする Mule ランタイムと MuleSoft コントロール プレーンで収集されるテレメトリ データに関する MuleSoft のドキュメントを参照してください。

最新問題**: 56**
Anypoint Platform 組織は、ID 管理とクライアント管理のために外部 ID プロバイダ (IdP) を設定しています。Anypoint Platform API に対してコマンドを実行するには、Anypoint CLI にどのような認証情報またはトークンを提供する必要がありますか？
A. ID管理のためにIdPによって提供される資格情報
B. クライアント管理のためにIdPによって提供される資格情報
C. クライアント管理用にIdPから提供された資格情報を使用して生成されたOAuth 2.0トークン
D. ID管理のためにIdPによって提供された資格情報を使用して生成されたOAuth 2.0トークン
Answer: A (メーッセージを残す**)**
正解: ID管理のためにIdPによって提供される資格情報
****
参照：
>> Anypoint CLI 経由の認証において、クライアント/ID プロバイダからの OAuth 2.0 トークンはサポートされていません。使用可能なトークンは「ベアラートークン」のみで、これも https://anypoint.mulesoft.com/accounts/login から取得できる Anypoint 組織/環境クライアント ID とシークレットを使用してのみ生成されます。クライアントプロバイダのクライアント資格情報ではありません。そのため、OAuth 2.0 は使用できません。さらに、このトークンは主に API Manager 用であり、ユーザーに関連付けられていません。Mulesoft ナレッジ記事に記載されているように、ほとんどの API（Cloudhub など）の呼び出しには使用できません。
>> Anypoint CLI で許可されているもう一つのオプションは、クライアント資格情報を使用することです。クライアントプロバイダのクライアント資格情報を使用することは可能ですが、クライアント管理で接続アプリケーションを設定する必要があります。しかし、質問で説明されているシナリオでは、そのような詳細は提供されていません。
>> 残された唯一の選択肢は、IDプロバイダーからのユーザー認証情報を使用することです

Valid Mule-Arch-201 Dumps shared by GoShiken.com for Helping Passing Mule-Arch-201 Exam! GoShiken.com now offer the **newest Mule-Arch-201 exam dumps**, the GoShiken.com Mule-Arch-201 exam **questions have been updated** and **answers have been corrected** get the **newest** GoShiken.com Mule-Arch-201 dumps with Test Engine here: https://www.goshiken.com/Salesforce/Mule-Arch-201-mondaishu.html (**154** Q&As Dumps, **30%OFF** **Special Discount:** **Freepdfdumps**)