

Copado.Copado-Extension-Builder.v2026-07-24.q25

試験コード:	Copado-Extension-Builder
試験名称:	Copado Certified Extension Builder
認定資格:	Copado
無料問題数:	25
バージョン:	v2026-07-24
アクセス数:	118
ページビュー数:	250
https://www.jpnpdf.com/Copado.Copado-Extension-Builder.v2026-07-24.q25-mondaishu.html	

最新問題: 1

現在、Copadoは3つの管理システムとの組み込み統合機能を提供しています。その3つの管理システムは以下のとおりです。

- A. Atlassian Jira、ServiceNOW、ClickUp
- B. Azure DevOps、ServiceNOW、Asana
- C. Jira、Azure DevOps、Wrike
- D. Jira、Azure DevOps、ServiceNOW

Answer: D (メッセージを残す)

Copado Extension Builderトピックからの包括的かつ詳細な100～150語の説明：

Copado は、Jira、Azure DevOps、ServiceNOW とのアジャイルおよびワークマネジメント統合機能を内蔵しています。これらの統合により、チームは計画およびデリバリーシステムを Copado の DevOps プロセスに接続し、ユーザー ストーリーと関連するデリバリーワークを Copado パイプラインを通じて管理できます。Jira はアジャイル プロジェクト追跡によく使用され、Azure DevOps は Microsoft ベースの計画および開発ワークフローをサポートし、ServiceNOW はエンタープライズ サービス管理および変更ガバナンスによく使用されます。オプション D は、サポートされている 3 つのシステムすべてが含まれているため、正解です。他のオプションには、ClickUp、Asana、Wrike などのツールが含まれています。これらは人気のワークマネジメント プラットフォームかもしれませんが、PDF には、Copado に組み込まれている 3 つの管理システム統合として記載されていません。これらの統合は、Copado のより広範なプラットフォーム拡張性の一部です。

最新問題: 2

動的式に関して、次のうちどれが正しいですか？

- A. 動的式は異なるコンテキストで実行され、関数が実行するジョブに応じて異なるコンテキスト ID が必要です。

B. 動的式は単一のコンテキストで実行され、関数が実行するジョブに対して同じコンテキスト ID が必要です。

C. 動的式とは、関数内でコンテキスト値をパラメータとして渡すことを可能にする式です。

D. Copadoには5種類の動的表現があります。

Answer: (解答を表示する)

Copado Extension Builderトピックからの包括的かつ詳細な100～150語の説明：

動的式は、実行時にコンテキスト値を解決し、それらの値をパラメーターとして Copado 関数または Salesforce フロー ステップに渡すために使用されます。オプション A が正しいのは、動的式はさまざまなコンテキストで実行でき、関数が実行される場所に応じて異なるコンテキスト ID が必要になる場合があるためです。たとえば、コンテキストは、デプロイメント、リポジトリ、ユーザー ストーリー、ジョブの実行、または別の Copado レコードに関連付けられる場合があります。オプション C も正しいのは、動的式は実行時の値を取得して自動化に渡す数式のように機能するためです。オプション B は、動的式が1つの固定コンテキストに限定されないため、誤りです。オプション D は、質問が動的式の動作と値の受け渡し方法に焦点を当てているため、ここで期待される正しい記述ではありません。

最新問題: 3

Copado Extensionsツールの機能を使用して生成できるDevOpsレコードの種類はどれですか？（該当するものをすべて選択してください）

A. コパド機能

B. アペックスクラス

C. 鉛源

D. クールな求人テンプレート

Answer: A,D (メッセージを残す)

Copado Extension Builderトピックからの包括的かつ詳細な100～150語の説明：

Copado Extensions ツールは、Copado Functions および Copado Job Templates の拡張レコードを生成できます。オプション A と D は、拡張形式に変換できるサポートされている Copado DevOps レコードであるため、正解です。Functions にはスクリプトベースの自動化ロジックが含まれ、Job Templates は Job Steps で構成される再利用可能な実行プロセスを定義します。これらは Copado データレコードであるため、拡張ツールは必要な JSON 表現を生成することで、これらのレコードのパッケージ化を支援します。Apex Class は、Salesforce メタデータであり、Copado 拡張レコードとして生成されるのではなく、標準のメタデータのパッケージ化またはデプロイによって処理されるため、このコンテキストでは正しくありません。リードソースも、ビジネスデータ値であり、Copado DevOps 拡張レコードではないため、関係ありません。

最新問題: 4

Copado Platformについて、以下の記述のうち正しいものはどれですか？（該当するものをすべて選択してください）

- A. Copado Platformは、企業がDevOpsプロセスをカスタマイズし、ビジネス要件に合わせてCopadoをパーソナライズすることを可能にします。
- B. Copado Platformは可視性を提供し、変更が本番環境に到達する前に品質基準を満たしていることを保証します。
- C. Copado Platformを使用すると、開発者は環境にコンプライアンスルールを割り当て、リスクを効果的に管理できます。
- D. Copado Platformは、開発者の認知負荷を軽減し、DevOpsライフサイクルにおける自動テストを促進するツールで構成されています。

Answer: A,B,D ([メッセージを残す](#))

Copado Extension Builderトピックからの包括的かつ詳細な100～150語の説明：

Copado Platform は、自動化によって企業が DevOps プロセスをカスタマイズし、可視性を向上させ、品質を確保し、開発者の複雑さを軽減するのに役立ちます。オプション A が正解なのは、プラットフォームが拡張機能、統合、SDK ツール、関数、ジョブ テンプレート、DevOps Exchange アセットによるパーソナライゼーションをサポートしているためです。オプション B が正解なのは、Copado が変更の可視性を提供し、変更が本番環境に反映される前にチームが品質を確保できるように支援するためです。オプション D も正解です。プラットフォームには、再利用可能な自動化、テスト、標準化されたリリース プラクティスを促進することで、手作業を減らし、認知負荷を軽減するツールが含まれているためです。

オプションCは最適な回答ではありません。なぜなら、コンプライアンスルールを環境に割り当てることは、ここでテストされているプラットフォームの要件ではないからです。ここで重要なのは、Copadoの拡張性、品質重視、自動化対応のDevOps機能です。

最新問題: 5

フリーダは、その関数を実行するために、関数イメージにプリインストールされていないライブラリが必要なのですが、どうすればよいでしょうか？

- A. Freida は、関数実行の一部としてライブラリをインストールできます。たとえば、npm install です。
- B. フリーダは代替として既存の図書館をどれでも選択できます。

Answer: ([解答を表示する](#))

Copado Extension Builderトピックからの包括的かつ詳細な100～150語の説明：

Freidaは、関数実行時に必要なライブラリをインストールする必要があります。Copado関数は、事前にインストールされたツール、アプリケーション、ライブラリを含むコンテナ化された環境である関数イメージ内で実行されます。ただし、選択したイメージには、特定の関数に必要なすべての依存関係が含まれていない場合があります。

必要なライブラリが見つからない場合、関数スクリプトは実行中にそれをインストールできます。たとえば、Node ベースの関数の場合は npm install を使用できます。これにより、

開発者は Copado の管理された実行環境を使用しながら、関数の柔軟性を維持できます。オプション B は、別のライブラリを選択しても同じ機能要件を満たせない可能性があるため、最適な解決策ではありません。推奨されるアプローチは、依存関係のインストール手順を関数のロジックに含めることで、実行の再現性と一貫性を確保することです。

最新問題: 6

Copado Job Engineに関して、以下の記述のうち正しいものはどれですか？（該当するものをすべて選択してください）

- A. ジョブエンジンは、関数とフローが動的な式を使用してさまざまなコンテキストの詳細にアクセスできるようにします。
- B. ジョブエンジンとは、ジョブテンプレートの実行結果として実行されるタスクのことです。
- C. ジョブエンジンを実行するたびに、ジョブ実行レコードが生成され、ジョブテンプレートにリンクされて、あらゆるレベルでの実行を追跡します。
- D. ジョブエンジンは、ジョブテンプレート、ジョブ実行、ジョブステップという3つの主要なオブジェクトで構成されています。

Answer: A,C,D (メッセージを残す)

Copado Extension Builderトピックからの包括的かつ詳細な100～150語の説明：

Copado Job Engine は、ジョブ テンプレート、ジョブ ステップ、およびジョブ実行レコードを使用して再利用可能な自動化を実行するフレームワークです。オプション A が正しいのは、Job Engine を介して実行される関数と Salesforce フローは、動的式を使用してコンテキスト値を取得し、それをパラメーターとして渡すことができるためです。オプション D が正しいのは、Job Engine の 3 つの主要オブジェクトが、ジョブ テンプレート、ジョブ ステップ、およびジョブ実行であるためです。オプション C も、想定される試験のコンテキストでは正しいです。ジョブ テンプレートが Job Engine を介して実行されると、Copado はジョブ テンプレートにリンクされたジョブ実行レコードを作成し、ログ、ステータス、および結果を使用して実行を追跡できるようにします。

選択肢Bは誤りです。実行中に実行されるタスクは、ジョブエンジン自体ではなく、ジョブステップに近いからです。

最新問題: 7

Copado Job Templateは、開発者がCopadoが管理するインフラストラクチャから直接スクリプトを構築、デプロイ、実行できるようにするCopado SDKツールです。

A. 真

B. 偽

Answer: B (メッセージを残す)

Copado Extension Builderトピックからの包括的かつ詳細な100～150語の説明：

この記述は、Copado ジョブ テンプレートではなく Copado 関数について説明しているため、誤りです。Copado ジョブ テンプレートは、ジョブ ステップを含む再利用可能な自動

化構造です。これらのステップには、関数、Salesforce フロー、または手動タスクが含まれる場合がありますが、ジョブ テンプレート自体は、Copado 管理インフラストラクチャから直接スクリプトを構築、デプロイ、および実行するために使用されるツールではありません。Copado 関数はスクリプト実行機能を提供し、サポートされているプログラミング言語と関数イメージを使用して自動化を実行できます。ジョブ テンプレートは再利用可能なプロセスを整理して実行しますが、関数はスクリプトベースの実行を実行します。この違いは、ジョブ テンプレートがオーケストレーションを定義するのに対し、関数が実行可能な自動化ロジックを定義するため、拡張機能ビルダーのトピックでは重要です。したがって、正解は「誤り」です。

最新問題: 8

Copado拡張ツールを使用してパッケージ化可能な形式に変換できるレコードの種類は何ですか？（該当するものをすべて選択してください）

- A. ユーザーストーリーの記録
- B. アペックスクラス
- C. ジョブテンプレート
- D. 機能

Answer: C,D (メッセージを残す)

Copado Extension Builderトピックからの包括的かつ詳細な100~150語の説明：

Copado Extensions ツールは、特定の Copado DevOps レコードをパッケージ化可能な拡張フォーマットに変換するために使用されます。適切なレコードはジョブ テンプレートと関数であるため、オプション C と D が正解です。これらのレコードは、再利用可能な自動化ロジックとスクリプトベースの動作を定義し、バンドル、配布、および別の Copado 組織へのインストールが可能です。ユーザー ストーリー レコードは、再利用可能な拡張構成ではなく、配信作業項目と環境固有のプロジェクト データを表すため、正解ではありません。Apex クラスも、このコンテキストでは正解ではありません。Apex クラスは Salesforce のメタデータであり、通常の Salesforce パッケージングまたはデプロイ メカニズムで処理され、このツールによって Copado 拡張レコードとして変換されるわけではないためです。

この拡張ツールは、Copadoの関数およびジョブテンプレートのデータをパッケージ化することに重点を置いています。

最新問題: 9

Copado Extensionツールの最も重要な2つの用途は何ですか？

- A. インストールされているCopado拡張機能のリストを取得します。オープンソースの拡張機能を作成し、DevOps Exchangeで公開します。
- B. Copado拡張機能をインストールし、拡張機能レコードを生成します。関数とジョブテンプレートをパッケージ化可能な形式に変換します。

- C. レコードがデータではなくメタデータであるセットアップオブジェクトを作成します。クローズドソースの拡張機能を作成し、DevOps Exchangeで公開します。
- D. さまざまな形式の拡張ファイルをアップロードします。メタデータコンポーネントをバンドルしてパッケージを作成します。

Answer: (解答を表示する)

Copado Extension Builderトピックからの包括的かつ詳細な100～150語の説明：

Copado Extension ツールの最も重要な 2 つの用途は、Copado Extension のインストールと拡張レコードの生成、および関数とジョブ テンプレートをパッケージ化可能な形式に変換することです。オプション B が正しいのは、このツールが拡張ライフサイクルの両側面、つまりパッケージ化可能な拡張コンテンツの作成とインストール後のレコード生成の両方をサポートしているためです。これは、関数とジョブ テンプレートが通常の Salesforce メタデータ コンポーネントではなく、Copado データ レコードであるため、特に重要です。このツールは、これらのレコードを静的リソースに保存してパッケージに含めることができる JSON ベースの形式に変換します。オプション A は不完全で、リスト化または公開に重点を置きすぎています。オプション C はカスタム メタデータを一般的に説明しており、オプション D は Copado 拡張レコードではなく、一般的なメタデータのバンドルに誤って焦点を当てています。

最新問題: 10

ケ빈は、ユーザーストーリーの展開手順の一部を実行する関数を作成しています。彼は、特定のオブジェクトレコード内の特定のフィールドを検索したり、関連する親レコード内のフィールドを検索したりできる動的な式を参照したいと考えています。

ケ빈は以下のどの表現タイプを選択すべきでしょうか？

- A. CObjectField 式は、
- B. レコード表現
- C. SObject式
- D. ユーザー表現

Answer: A (メッセージを残す)

Copado Extension Builderトピックからの包括的かつ詳細な100～150語の説明：

ケ빈は、CObjectField Expression を選択する必要があります。要件は、オブジェクトレコードまたは関連する親レコードから特定のフィールド値を取得することです。動的式は、Copado でコンテキスト値を動的に解決し、それらの値を関数または Salesforce フローのステップに渡すために使用されます。自動化でレコード全体を参照するのではなく、特定の値を取得する必要がある場合は、フィールドに焦点を当てた式タイプが適切です。SObject Expression は Salesforce オブジェクトコンテキストに広く関連していますが、質問ではレコードまたは親レコードのフィールドを具体的に求めています。ユーザー式は、オブジェクトフィールドの取得ではなく、ユーザー関連のコンテキストに適用されます。したがって、CObjectField Expression が正解です。

最新問題: 11

Copadoのジョブ実行詳細タブにはどのような情報が表示されますか？

- A. 詳細タブには、ジョブ実行タイプ、API名、説明など、ジョブテンプレートに関する情報が含まれています。
- B. 詳細タブに「実行の再開」というチェックボックスがあります。これをクリックすると、すべてのジョブステップが再実行されます。
- C. 詳細タブには、ジョブテンプレートの一部としてトリガーされたジョブステップのリストが表示されます。
- D. 詳細タブでは、ジョブテンプレートの実行結果（ステータスや、実行が失敗した場合のエラーメッセージなど）を確認できます。

Answer: D ([メッセージを残す](#))

Copado Extension Builderトピックからの包括的かつ詳細な100～150語の説明：

ジョブ実行の詳細タブには、ジョブテンプレートの実行結果に関する情報が表示されません。オプションDが正解です。このタブには、実行が失敗した場合のステータスやエラーメッセージなど、実行関連の詳細が含まれています。この情報は、管理者やリリースチームが失敗した自動化のトラブルシューティングを行い、特定の実行中に何が起こったかを理解するのに役立ちます。ジョブテンプレートは再利用可能な自動化を定義しますが、ジョブ実行レコードはそれらのテンプレートの実際の実行を表します。オプションAは、API名や説明などのテンプレートレベルの詳細に誤って焦点を当てています。オプションBは、詳細タブが主に実行の再開チェックボックスで説明されているわけではないため、不正確です。オプションCは実行に関連する手順について言及していますが、質問は詳細タブについて具体的に尋ねています。したがって、実行結果と失敗の詳細が正しい焦点です。

最新問題: 12

カスタムメタデータ型を作成する必要があるのはなぜですか？

- A. データレコードを保存し、JSONファイルを生成します。
- B. 新しいStandardResourceレコードを作成し、静的リソースをリンクします。
- C. カスタムDevOpsパイプラインを作成し、変更をHerokuのみにデプロイします。
- D. ジョブテンプレートのカスタムメタデータレコードを設定します。

Answer: ([解答を表示する](#))

Copado Extension Builderトピックからの包括的かつ詳細な100～150語の説明：

Copado Extensionの作成プロセスでは、拡張パッケージを静的リソースにリンクする構成レコードを作成するために、カスタムメタデータタイプが必要です。オプションBは正しいですが、表現に問題があるようで、必要な静的リソース関連の構成を作成するものと理解する必要があります。Copado Extensionsでは、選択された関数またはジョブテンプレートレコードがJSONファイルに生成され、静的リソースに保存されます。その後、カスタムメタデータレコードがその静的リソースを指すため、Copadoはインストール後にどのパッケージ化された拡張データを処理する必要があるかを認識できます。オプションAはJSON生成の手順を説明していますが、これはカスタムメタデータタイプを作成する理由

ではなく、Copado Extensions ツールによって実行されます。オプション C と D は正しい目的ではありません。

最新問題: 13

これらの機能のうち、Copado Platformを他のDevOpsプラットフォームと区別するものはどれですか？

- A. Copado Platformの機能を使用すると、APIを介して外部システムからユーザーストーリーの変更リストをコミットできます。
- B. Copado Platformを使用すると、1つまたは複数のテストツールを使用してテストを実行し、即座に結果を取得できます。
- C. Copado Platform を使用すると、自動化スクリプトに高度なパラメーターとカスタマイズを追加できます。
- D. Copado Platform は CI/CD パイプラインの作成を容易にします。

Answer: C (メッセージを残す)

Copado Extension Builderトピックからの包括的かつ詳細な100~150語の説明：

この質問において、Copado Platformを最も際立たせる機能は、自動化スクリプトに高度なパラメータとカスタマイズを追加できる機能です。Copado Platformは、単なる標準的なCI/CDパイプラインツールではなく、関数、動的式、システムプロパティ、ジョブテンプレート、SDKツールを通じて拡張およびパーソナライズできるように設計されています。これらの機能により、チームは自動化を再利用可能で、コンテキストを認識し、特定のビジネス要件に適応させることができます。オプションAはActions APIの使用方法について説明していますが、これは有用ではあるものの、範囲が狭くなります。オプションBは品質とテストの統合について言及しており、これも重要ですが、この質問における最も強力な差別化要因ではありません。オプションDは、CI/CDパイプラインの作成は一般的なDevOps機能であり、Copado Platformの拡張重視の価値を独自に説明するものではないため、誤りです。

最新問題: 14

Copadoの拡張機能作成プロセスの一環として、静的リソースを作成する必要があるのはなぜですか？

- A. Copadoでは、大きなファイルを拡張子にバンドルすることはできません。静的リソースを作成することで、レコードのファイルサイズを小さくすることができます。
- B. データレコードをパッケージに直接含めることはできません。データレコードは静的レコードを使用して保存し、拡張機能として使用できます。
- C. Copado を使用して拡張機能を作成する場合、レコードをクローズド ソースとしてバンドルすることをお勧めします。

静的リソースの作成は、ファイルをクローズドソースにするのに役立ちます。

- D. 上記のすべて

Answer: B (メッセージを残す)

Copado Extension Builderトピックからの包括的かつ詳細な100～150語の説明：

Copado Extension の関数やジョブ テンプレートなどのレコードはデータ レコードであり、標準の Salesforce メタデータ コンポーネントではないため、静的リソースが必要です。データ レコードは、メタデータと同じように管理パッケージまたはロック解除パッケージに直接含めることはできません。この問題を解決するために、Copado は生成された拡張機能データをファイル形式 (通常は JSON) に保存し、そのファイルを静的リソースとして含めます。インストール中に、Copado は静的リソースを読み取り、対象組織で対応するレコードを生成できます。オプション B は、拡張機能の作成プロセスで静的リソースを使用する本当の理由を説明しているため正しいです。オプション A は、主な目的がファイル サイズの削減ではないため誤りです。オプション C は、拡張機能をクローズド ソースにすることが目的ではないため誤りです。

最新問題: 15

Copadoでは、3つの標準ジョブステップのうち、どれを作成できますか？

- A. 機能、Salesforceフロー、手動タスク
- B. Salesforceフロー、手動タスク、品質ゲート
- C. 関数、アクションAPI、手動タスク
- D. 機能、手作業、品質統合フレームワーク

Answer: ([解答を表示する](#))

Copado Extension Builderトピックからの包括的かつ詳細な100～150語の説明：

Copado ジョブテンプレートでは、ジョブステップは再利用可能な自動化プロセスの一部として実行される個々のアクションを定義します。Copado は、関数、Salesforce フロー、手動タスクの 3 つの標準ジョブステップタイプを提供します。関数ステップを使用すると、外部コマンドの実行、検証、統合など、スクリプトベースの自動化を実行できます。Salesforce フローステップを使用すると、チームはジョブテンプレートプロセス内で宣言型の Salesforce 自動化を実行できます。手動タスクステップは、実行中に人間の介入または確認が必要な場合に使用します。品質ゲート、アクション API、品質統合フレームワークは Copado の重要な概念ですが、このコンテキストでは 3 つの標準ジョブステップタイプではありません。したがって、オプション A がサポートされている標準ジョブステップタイプを正しく識別しています。

最新問題: 16

Copado Job Engineの主な構成要素を3つ挙げてください。

- A. ジョブテンプレート、ジョブステップ、ジョブ拡張機能
- B. ジョブテンプレート、ジョブステップ、ジョブ実行
- C. 求人テンプレート、求人プロモーション、求人配置
- D. 職務ステップ、職務昇進、職務配置

Answer: B ([メッセージを残す](#))

Copado Extension Builderトピックからの包括的かつ詳細な100～150語の説明：

Copado Job Engine の 3 つの主要オブジェクトは、ジョブ テンプレート、ジョブ ステップ、およびジョブ実行です。ジョブ テンプレートは、再利用可能な自動化プロセスを定義します。これは、実行する作業を整理する親構造として機能します。ジョブ ステップは、関数の実行、Salesforce フローの実行、手動タスクの割り当てなど、テンプレート内の個々のアクションです。ジョブ実行は、ジョブ テンプレートの特定の実行を表し、実行ステータス、ログ、および結果を格納します。オプション B は、3 つのコア オブジェクトすべてが含まれているため、正解です。ジョブ拡張、ジョブ プロモーション、およびジョブ展開は、ジョブ エンジン モデルを定義する 3 つの標準オブジェクトではありません。これらのオブジェクトを理解することは、Copado 自動化の構築とトラブルシューティングに不可欠です。

有効な **Copado-Extension-Builder** 問題集は GoShiken.com が提供された合格しやすい Copado-Extension-Builder 試験問題集！ GoShiken.com が最新の **Copado-Extension-Builder** 試験問題集を提供しています。GoShiken.com Copado-Extension-Builder 試験問題は最新で、解答が正確でございます。最新の GoShiken.com Copado-Extension-Builder 問題集をゲットする人はこちら：

<https://www.goshiken.com/Copado/Copado-Extension-Builder-mondaishu.html>

(**7030%OFF**問題集溶と正解付きで **30%w** 特別割引コード: **Freepdfdumps**)

最新問題: 17

次のうち、ジョブ実行を最も適切に定義しているのはどれですか？

- A. ジョブ実行とは、Salesforce Flowのデプロイ手順で使用される、事前定義された変数のセットです。
- B. ジョブ実行を使用すると、ジョブテンプレートで使用する Copado アクションを確認できます。
- C. ジョブ実行機能を使用すると、ジョブテンプレートの実行結果を検証できます。
- D. ジョブ実行は、ジョブテンプレート実行の一部として実行されるタスクです。

Answer: C ([メッセージを残す](#))

最新問題: 18

動的式を正確に説明する定義を選択してください。

- A. テストしたいコードの品質を判断するのに役立つテスト指標。
- B. コミット、デプロイ、ブランチ検証、Seleniumテストケースを環境に依存しないようにする機能。
- C. 関数またはジョブステップで参照される値を格納するコンテナ。
- D. 異なるID、変数、文字を含む値要素の軌跡。

Answer: ([解答を表示する](#)**)**

Copado Extension Builderトピックからの包括的かつ詳細な100~150語の説明：

動的式は、実行時にコンテキスト情報を解決する値要素の軌跡またはシーケンスとして理解できます。オプション D が最も適切な定義です。動的式は、さまざまな Copado コンテキストから値を取得し、それらの値を関数または Salesforce フローに渡すように構築されています。これらの値には、レコード ID、変数、オブジェクト フィールド、リポジトリ情報、ユーザー ストーリー データ、またはその他のコンテキスト要素が含まれる場合があります。これにより、環境固有またはレコード固有の値をハードコーディングするのではなく、自動化を再利用可能な状態に保つことができます。オプション A は、動的式がテストメトリックではないため、不適切です。オプション B は、コンテキスト認識型自動化の考えられる利点を説明していますが、正確な定義ではありません。オプション C は、動的式ではなく、再利用可能な値を格納するシステム プロパティに近いものです。

最新問題: 19

Copadoでは、関数、Salesforce Flow、手動タスクに加えて、カスタムジョブステップを作成することもできます。

A. 真

B. 偽

Answer: A ([メッセージを残す](#))

Copado Extension Builderトピックからの包括的かつ詳細な100～150語の説明：

その通りです。Copadoは、関数、Salesforceフロー、手動タスクなどの標準的なジョブステップタイプを提供していますが、プラットフォームは拡張性も考慮して設計されています。つまり、標準オプションでは自動化要件を完全に満たせない場合、チームはカスタムジョブステップの動作を作成できます。カスタムジョブステップは、組織がCopadoを外部プラットフォームと連携させたり、特別なビジネスプロセスを適用したり、再利用可能な自動化をより広範な拡張機能の一部としてパッケージ化したりする必要がある高度なDevOpsシナリオで役立ちます。

これは、Copado Extension Builder の目的、つまり標準機能を超えて Copado の機能を拡張し、カスタマイズするという目的に合致しています。標準ステップは一般的なシナリオに対応し、カスタムステップを使用すると、組織は特定の DevOps プロセスに合わせてジョブ テンプレートをカスタマイズできます。

最新問題: 20

次のうち、Copado DevOps Exchange拡張機能の種類ではないものはどれですか？

A. コパド・ラボ拡張機能

B. コパドエクステンション

C. パートナー&コミュニティ拡張機能

D. 顧客拡張機能

Answer: A ([メッセージを残す](#))

Copado Extension Builderトピックからの包括的かつ詳細な100～150語の説明：

Copado Labs Extensionsは、Extension Builderの資料に記載されている標準的なCopado DevOps Exchange拡張機能タイプには含まれていません。認識されている拡張機能カテゴリは、Copado Extensions、Partner & Community Extensions、Customer Extensionsです。Copado Extensionsは通常、Copadoによって提供されます。

パートナーおよびコミュニティ拡張機能は、パートナーまたはコミュニティメンバーによって提供され、Copado に追加の統合機能や再利用可能な DevOps 機能を追加します。顧客拡張機能は、顧客固有のユースケースや社内のビジネス要件に合わせて構築されます。

Copado Labs Extensions」はこれらの拡張機能カテゴリのいずれにも該当しないため、オプション A が正解です。DevOps Exchange は、チームが Copado のプラットフォーム機能を強化する拡張機能を発見、インストール、再利用できるマーケットプレイスとして機能するため、これらのカテゴリを理解することが重要です。

最新問題: 21

Copado DevOps Exchangeマーケットプレイスで利用可能な拡張機能の種類を3つ挙げてください。

- A. Copado拡張機能、カスタムメイド拡張機能、およびユーザー定義拡張機能。
- B. Copado拡張機能、パートナー&コミュニティ拡張機能、顧客拡張機能。
- C. DevOps拡張機能、データテンプレート拡張機能、テストスクリプト拡張機能。
- D. Copado DevOps拡張機能、パートナー拡張機能、および顧客拡張機能。

Answer: B ([メッセージを残す](#))

Copado Extension Builderトピックからの包括的かつ詳細な100~150語の説明：

Copado DevOps Exchange マーケットプレイスで利用できる拡張機能には、Copado 拡張機能、パートナー & コミュニティ拡張機能、顧客拡張機能の 3 種類があります。オプション B が正解です。これは、Copado エコシステムで拡張機能がどのように提供され、利用されるかを説明するために使用される拡張機能のカテゴリと一致しているためです。Copado 拡張機能は Copado によって作成または提供され、パートナー & コミュニティ拡張機能はパートナーまたはより広範なコミュニティによって提供され、顧客拡張機能は顧客固有のニーズに合わせて作成されます。この分類は、DevOps Exchange がチームが Copado の機能を拡張する再利用可能なソリューションを見つけることができるマーケットプレイスとして機能するため重要です。オプション A、C、および D は、カスタムメイド、ユーザー定義、データ テンプレート、テスト スクリプト拡張機能など、不正確または不完全なカテゴリ名を使用しており、これらは想定される DevOps Exchange 拡張機能のカテゴリではありません。

最新問題: 22

次のうち、ジョブ実行を最も適切に定義しているのはどれですか？

- A. ジョブ実行機能を使用すると、ジョブテンプレートの実行結果を検証できます。
- B. ジョブ実行を使用すると、ジョブテンプレートで使用する Copado アクションを確認できます。

C. ジョブ実行は、ジョブテンプレート実行の一部として実行されるタスクです。

D. ジョブ実行とは、Salesforce Flowのデプロイ手順で使用する、事前定義された変数のセットです。

Answer: ([解答を表示する](#))

Copado Extension Builderトピックからの包括的かつ詳細な100～150語の説明：

ジョブ実行とは、ジョブテンプレートの実行結果を検証および確認するために使用される記録と定義するのが最も適切です。ジョブテンプレートが実行されると、Copadoはその特定の実行に対してジョブ実行記録を作成します。

このレコードは、実行が失敗した場合のステータス、ログ、結果、およびエラーメッセージを表示します。オプション A は、実行結果の検証に焦点を当てているため、正解です。オプション C は、テンプレートの一部として実行されるタスクはジョブ実行ではなくジョブステップとして説明する方が適切であるため、不正解です。オプション B は、ジョブ実行が Copado アクションの検証に使用されないため、無関係です。オプション D は、実行追跡ではなく、変数または動的値について説明します。ジョブ実行は、主に実行時の追跡とトラブルシューティングに使用されます。

最新問題: 23

システムプロパティを使用すると、Copadoでは機密性の高い値のみを保存でき、それらの値はシステムプロパティオブジェクトに格納されます。

A. 真

B. 偽

Answer: ([解答を表示する](#))

Copado Extension Builderトピックからの包括的かつ詳細な100～150語の説明：

この記述は誤りです。システムプロパティは機密値のみに限定されるものではありません。Copado Extension Builder の概念では、システムプロパティは再利用可能な値であり、パラメーターで参照したり、パイプライン、ユーザー、またはオブジェクトレコードにリンクしたりできます。これらは、関数やジョブステップでハードコードされた値を回避するのに役立ちます。システムプロパティには、資格情報、トークン、構成シークレットなどの機密値が含まれる場合がありますが、自動化が実行時に必要とする機密性の低い構成値も保存できます。この柔軟性により、同じ関数またはテンプレートを異なる環境やレコード間で動作させることができ、適切なプロパティ値を動的に解決できます。したがって、システムプロパティには機密値しか保存できないというのは、制限が厳しすぎ、誤りです。

最新問題: 24

Copado Extensionsのカスタムメタデータタイプは、以下のように定義できます。

A. メタデータコンポーネントの動作を決定し、要件に応じてカスタマイズするのに役立つメタデータタイプ。

B. データレコードの代わりにメタデータレコードを追加できるメタデータタイプで、アプリやソリューションを構築する機能を提供します。

- C. パイプラインに接続されている組織上の不要なメタデータの変更を監視し、デプロイメントからそれらを削除できるメタデータタイプ。
- D. プロファイルと権限セット以外のカスタムベースのメタデータコンポーネントを含むメタデータタイプ。

Answer: B ([メッセージを残す](#))

Copado Extension Builderトピックからの包括的かつ詳細な100～150語の説明：

Copado Extensions カスタムメタデータ型は、通常のデータレコードではなくメタデータとして構成のようなレコードをパッケージ化するために使用されます。オプション B が正解です。カスタムメタデータ型を使用すると、開発者はパッケージに含めて組織全体に展開できるレコードを定義できます。これは、拡張機能関連の構成が別の Copado 環境に移行可能でインストール可能である必要があるため、Copado Extensions にとって重要です。拡張機能を構築する際、拡張機能構成の詳細などのレコードはカスタムメタデータで表現できるため、パッケージ化可能なセットアップデータのように動作します。オプション A は一般的すぎて、メタデータレコードとデータレコードの重要な違いを捉えていません。オプション C はメタデータ監視について説明していますが、これは関係ありません。オプション D も誤りです。カスタムメタデータ型は、プロファイル以外のメタデータコンポーネントの単なるコンテナではないからです。

最新問題: 25

提示された選択肢の中から、ジョブエンジンに関する正しい記述を選択してください。（該当するものをすべて選択してください）

- A. ジョブエンジンを使用すると、失敗したジョブの実行を再開したり、最後に失敗したステップから開始したりできます。
- B. ジョブエンジンは、関数とフローが動的な式を使用してさまざまなコンテキストの詳細にアクセスできるようにします。
- C. ジョブエンジンを実行するたびに、ジョブ実行レコードが作成され、対応するジョブテンプレートに関連付けられます。
- D. ジョブエンジンは、実行ログを再計算し、プロジェクトの一部として実行されたジョブテンプレートを追跡するための結果ダッシュボードを準備するCopadoの機能です。

Answer: ([解答を表示する](#))

Copado Extension Builderトピックからの包括的かつ詳細な100～150語の説明：

Copado Job Engine は、ジョブテンプレートとジョブステップを通じて再利用可能な自動化を実行するために使用されます。重要な機能として、失敗した実行を再開できます。これは、実行シナリオに応じて、実行全体を再度実行するか、未完了または失敗したステップのみを実行するかのいずれかです。オプション A は、自動化の失敗時の復旧動作を反映しているため、正解です。オプション B も正解です。Job Engine は関数と Salesforce フローと連携し、動的式により、これらのステップは実行中にコンテキスト値にアクセスできます。オプション C は不正確です。通常の実行単位は「Job Engine の実行」ではなく、ジョブ

テンプレートだからです。オプション D は、Job Engine は単なるダッシュボードや再計算機能ではないため、不正解です。

Valid Copado-Extension-Builder Dumps shared by GoShiken.com for Helping Passing Copado-Extension-Builder Exam! GoShiken.com now offer the **newest Copado-Extension-Builder exam dumps**, the GoShiken.com Copado-Extension-Builder exam **questions have been updated** and **answers have been corrected** get the **newest** GoShiken.com Copado-Extension-Builder dumps with Test Engine here: <https://www.goshiken.com/Copado/Copado-Extension-Builder-mondaishu.html> (70 Q&As Dumps, **30%OFF** Special Discount: **Freepdfdumps**)