PeopleCert DevOps.

PeopleCert DevOps
サイト・リライアビリティ・エンジニア（SRE）

シラバス

January 2022
v1.0 J

PeopleCert®

All talents, certified.

e-mail: info@peoplecert.org, www.peoplecert.org

1. はじめに

DevOpsとは、Development（開発）とOperations（運用）の合成語で、ソフトウェア開発（Dev）と情報技術運用（Ops）を組み合わせたソフトウェア開発手法です。DevOpsの目標は、システム開発ライフサイクルを短縮する一方で、ビジネス目標に密接に整合したソフトウェア・リリースを、より速く、より良く、より安く、頻繁に提供することです。DevOpsのアプローチは、カルチャ的な転換、プラクティスとプロセスの改善、自動化技術のレバレッジという3つの重要成功要因に対応しています。

DevOpsアプローチによる作業の成功が証明されたことにより、DevOpsに精通した人材を必要とする組織の数が常に増加しており、DevOps指向の役割は、業界を問わず、常に多くの組織で求められるリソースとなっています。組織は、組織カルチャの転換を図り、DevOps環境で快適に作業し、コーディング、ビルド、テスト、パッケージ、リリース、構成、モニタリングすることを含む、これに限定されないプロセスのための一連の技術やツール（総称して「ツールチェーン」）をその人材によって管理してほしいと考えています。

このような継続的な進化を遂げる状況の中で、PeopleCertは、多くの組織が直面しているスキルギャップを解消し、ビジネス目標の実現をサポートし、コミュニケーション、標準化、コラボレーション、自動化を向上して、より優れた品質のソフトウェア・プロダクトをより早く、より低いコストで提供するという市場ニーズを反映した一連のDevOps資格を設計しました。

この進化する作業アプローチのニーズを反映し、PeopleCert DevOps資格は次のように構成されています。

- **PeopleCert DevOps Fundamentals - 受験者は、特に確立されたITSMプラクティスに関連した15の必須プラクティスを通じて、有効なDevOpsサービスを提供する方法についてのガイダンスを得ることができます。**
- **PeopleCert DevOpsリーダーシップ - DevOpsプラクティスの導入を主導する方法と、より良いコラボレーションとコミュニケーションのためのカルチャ的な転換を達成する方法に関するプラクティス的なガイダンスを得ることができます。**
- **PeopleCert DevOps Engineer - このモジュールは、DevOps Engineerに必要な主要スキルとツールについて、受験者が深い理解を得られるようにします。**

- **PeopleCert DevSecOps - このモジュールは、典型的なセキュリティ・リスクを軽減し、主要なプラクティスを通じてリスクを効果的に管理するために必要な知識を受験者に提供します。**
- **PeopleCert DevOps Site Reliability Engineer - このモジュールは、システムの信頼性に関する新しい観点を生み出すことで、プラットフォームの設計、開発、および運用の実行の間のギャップを埋めるために必要な知識を受験者に提供します。**

PeopleCert DevOps Site Reliability Engineerは、ソフトウェア・エンジニアリングの側面を取り入れ、インフラストラクチャと運用の問題に適用する一連の原則およびプラクティスに関する詳細な知識をカバーしています。

これらのスキルの基礎となる知識体系は、PeopleCertが認定ATO（教育研修機関）によって提供される公式コースウェアで紹介されます。シラバスの主な目的は、DevOpsに関わる人材の認定のための基礎を提供することです。シラバスは、資格に関連する学習成果をドキュメント化し、特定の資格レベルにおいてこれらの学習成果が達成されていることを証明するために受験者が満たすことを期待される要件を記述しています。

2. PeopleCert DevOps サイト・リライアビリティ・エンジニア (Site Reliability Engineer)

2.1. Site Reliability Engineer資格取得の目的

この資格の目的は、受験者がSite Reliability Engineerの主要なスキルとツールについて十分な知識、理解、応用力を持ち、これらのスキルと知識を分析、適用して、DevOpsチームと効果的に連携し、またはその主要メンバーとして働ける知識を保持しているか確認することです。PeopleCert DevOps Site Reliability Engineer認定の前提条件はありませんが、PeopleCert DevOps Fundamentals認定を取得することが強く推奨されます。

2.2. 対象グループ/オーディエンス

この資格は、PeopleCert DevOps資格スキームの一部で、DevOps環境の効率的な主要メンバーになることを希望する人を対象としています。受験者は、DevOps Site Reliability Engineering用語、原則、ツールおよびプラクティスに関する確かな知識と理解を持ち、実証するとともに、ツールを効率的かつ効果的に使用する方法に関する応用スキルを実証する必要があります。また、この資格は、組織単位での定着を行う利用と共に、個人的な資格取得を目指す受験者にも対応しています。

この資格は、認定資格保有者にDevOps Site Reliability Engineerレベルの知識を提供し、様々なツールを使用したDevOps Site Reliability Engineerのプラクティスをしっかりと理解し、DevOpsのプラクティスを含む日常業務に適用できることを証明するものです。基本的なスキルや知識は、PeopleCertが提供する資格スキームのPeopleCert DevOps Fundamentalsレベルでカバーされています。

3. 学習目標

Site Reliability Engineerレベルのコースでは、受験者は、Site Reliability Engineer標準で使用される概念、用語、原則、ツールを向上させ、DevOps変革の明確化、計画、アプローチ、妥当性確認、持続、組織内でDevOpsフルスタック・アプローチがどのように関与し実装できるか、それがどのように価値提供に連携できるかが学べます。さらに、スクラムの方法論、人材とカルチャの影響、および組織内でDevOpsを適応させるために使用するプラクティス、プロセス、自動化、技術についても説明します。

PeopleCert DevOps Site Reliability Engineer認定資格の保有者は、以下の知識、理解、プラクティスを実証することができます。

- DevOps Site Reliability Engineer の主な特徴と必須スキル
- DevOps の「3 つの道」と「5 つの理想」
- 成功の目標を定量化する方法
- 作業の手順の管理、委任、支援の基本的な方法
- システム管理、問題解決、運用タスクの自動化などのツールとしてのソフトウェアの管理

3.1. 資格制度レベル

上記の学習目標を通じて、受験者は以下の分野で関連するナレッジ・スキルを実証することになります。

主なトピック	
Site Reliability Engineering（SRE）とは？	SRE「オンコール」プロセス
SRE プラクティスの構築	リリース・エンジニアリングにおける SRE
成功の定量化	事業継続のための SRE

4. 試験

PeopleCert DevOps Site Reliability Engineer認定試験は、受験者の知識と理解の妥当性確認をするために設計されており、上記のSite Reliability Engineerの原則に加え、現代企業におけるDevOps変革を支援する実際の状況で、この知識を適用および分析することができるかを確認します。

PeopleCert DevOps Site Reliability Engineer試験では、学習レベル別の基準となるブルーム分類学の認知領域における以下の3つのカテゴリーに重点を置いています。[1]

- **ナレッジ / 知識**
- **理解度**
- **応用**

ブルーム分類学では、認知領域における学習の6つのレベル（知る、理解する、適用する、分析する、評価する、創造する）を定義しています。このように、順次的かつ累積的に、単純なものから複雑なものへと移行していきます。[1] 6段階目の学習を達成するためには、それまでの5段階を確実に習得していなければなりません。

4.1. 評価アプローチ

PeopleCert DevOps Site Reliability Engineer認定で使用される評価アプローチは、知識、理解度、応用の3つの基本カテゴリに重点を置いています。

ナレッジ（Knowledge）とは、事実から理論まで、以前に学習した内容を思い出すことと定義され、認知領域における学習成果の最も低いレベルに相当します。このような学習成果は、次のような知ること、思い出すことを含む評価目標に転化されます。

- 一般的および/または基本的な用語、定義、概念、 原則
- 具体的なプロセス
- プロセス、手順、プロジェクト管理手法

理解度（Comprehension）とは、最も低いレベルの理解であり、その過程には解釈、翻訳、推測の要素も含まれ、教えられた教材の意味を把握する能力を意味します。このような学習成果、ひいては評価目標は、単に情報を思い出すということにとどまらず、以下のようなものが含まれる可能性があります。

- 事実、概念、原則の理解
- 資料の解釈（例：コード、チャート、グラフ、テキスト、ダイアグラム）
- 使用したプロセス、手順、方法の正当化

応用（Application）は、受験者のナレッジと理解・納得を組み合わせ、抽象化することができるレベルです。より具体的には、受験者は特定の具体的な状況に対する抽象化、一般原則、または方法など、知識と理解を適用することが期待されます。このような学習成果、ひいては評価目標は、単に情報を思い出すということにとどまらず、以下のようなものが含まれる可能性があります。

- アイデア、原則、理論を新しい具体的な状況で使うことができる
- 与えられた状況において、適切な手法やツールの選択、原則の適用、特定のアプローチの使用、または選択肢の選択を特定することができる
- 学習したことを新しい状況に適用することができる
- ルール、方法論、概念、原則、理論を適用することができる
- この分野の学習成果は、「理解度」よりも高いレベルの理解が必要

この評価では、上記の認知領域のカテゴリーに対応した評価目標を使用するため、上記の学習成果を取り入れることができます。

4.2. 受講資格／研修要件

この試験の受験者は、PeopleCert DevOps Fundamentals認定を保持すべきですが、研修（トレーニング）に関する要件はありません。

PeopleCert DevOps Site Reliability Engineerレベル試験の受験資格を得るためには、受験者は、DevOps Site Reliability Engineerの基本用語、原則、プロセス、プラクティスに関する知識と理解を示し、現代企業におけるDevOps変革を支援する実際の状況においてこの知識を適用、分析できる必要があり、PeopleCert認定トレーニングパートナーによる関連認定トレーニングを受けていることが推奨されています。

4.3. 試験形式

PeopleCert DevOps Site Reliability Engineer試験の試験形式は、次の表のとおりです。

提供	コンピュータ（Webプロトタイプまたは教室）
タイプ	**40問の多肢選択問題（MCQ）** *各問題には1点が与えられます*
時間	**1時間（60分）** *非ネイティブスピーカーや障がいのある受験者の場合は、さらに15分の延長が認められます*
合格最低点	**70%（40点満点中28点）**
試験官／監督官／プロクター	**はい** *物理的またはオンラインによるプロクタリング*
オープンブック	**いいえ** *試験会場または試験に資料を持ち込むことはできません*
前提条件	**なし**
優等者の区別	**N/A**
認証の効力	**永続的**

テストは、以下のテスト仕様に基づき、定期的に更新される設問テストバンク（QTB）から抽出されます。設問は、テストセット間で互換性を持って使用されています。各試験の全体的な難易度は、他のあらゆる試験と同じです。複数回受験する場合、受験者に同じテストが割り当てられることはありません。

5. シラバス

シラバスは、主要な主題の見出しに関連するセクションで構成され、1桁のセクション番号が付されている。また、シラバスのカテゴリーごとに推奨されるトレーニング時間もこの表に記載されています。合計24時間の認定トレーニングが推奨されています。

カテゴリー	トピック	Ref.	ナレッジ/タスク項目
1.サイト・リライアビリティ・エンジニアリング（SRE）とは？	1.1 サイト・リライアビリティ・エンジニアリング入門	1.1.1	用語の定義：サイト・リライアビリティ・エンジニアリング（SRE）
		1.1.2	SREがDevOpsのデプロイメント・パイプラインにどのように適合するかを説明する
		1.1.3	「SREによるDevOpsの実行」という表現を説明する
	1.2 SREコア原則	1.2.1	リスクの受容: 用語の定義：リスク管理
		1.2.2	リスクの受容：サービス・リスクの測定方法を説明する
		1.2.3	リスクの受容：リスク管理とサービス可用性の概念を比較し、可用性目標がどのように作成されるかを説明する
		1.2.4	サービスレベル目標：用語の定義：サービスレベル指標（SLI）、サービスレベル目標（SLO）、サービスレベルアグリーメント（SLA）
		1.2.5	サービスレベル目標：SLOとサービスレベル目標（SLT）の概念を比較する
		1.2.6	サービスレベル目標：100%の可用性というSLOを求めない根拠を説明する
		1.2.7	トイルの撲滅：用語の定義、トイルと6つの作業属性のリスト
		1.2.8	トイルの撲滅：トイルとオーバーヘッドの違いを再理解
		1.2.9	トイルの撲滅：トイルの上限がSRE時間の50%である理由を説明する
		1.2.10	トイルの撲滅：“グランジ(grunge)“作業の概念と報奨の仕組みを説明する
		1.2.11	トイルの撲滅：どのような作業がエンジニアリングに該当するのかを説明する

カテゴリー	トピック	Ref.	ナレッジ/タスク項目
		1.2.12	トイルの撲滅：用語の定義：リニア・スケーリング
		1.2.13	分散システムのモニタリング：分散システムをモニタリングする理由を説明する
		1.2.14	分散システムのモニタリング：トリガーイベントの4つのゴールデンシグナルを説明する
		1.2.15	分散システムのモニタリング：アラートの3つのレベルを説明する
		1.2.16	自動化：用語の定義、自動化とその価値に関連する主な成果を説明する
		1.2.17	リリース・エンジニアリング：用語の定義：リリース・エンジニアリング
		1.2.18	リリース・エンジニアリング：SREとリリース・エンジニアの役割を比較する
		1.2.19	簡素化：本質的な複雑さと偶発的な複雑さの概念を比較する
		1.2.20	簡素化：システムの安定性とアジリティの概念を比較する
	1.3 SREのコア・プラクティス	1.3.1	エンジニアリングへの永続的な注力の確保：Site Reliability Engineeringのエンジニアリング面を説明する
		1.3.2	SLOに違反しない最大変更ベロシティ：エラーバジェットが最大変更ベロシティを可能にする方法を説明する
		1.3.3	SLOに違反しない最大変更ベロシティ：ほとんどのサービス停止における人間の役割を説明する
		1.3.4	イベントモニタリングとエマージェンシー・レスポンス：用語の定義：AIOps
		1.3.5	イベントモニタリングとエマージェンシー・レスポンス：用語の定義：オンコール
		1.3.6	イベントモニタリングとエマージェンシー・レスポンス：用語の定義：MTTR（Mean Time to Repair）
		1.3.7	需要予測、キャパシティ計画立案、プロビジョニング：用語の定義：キャパシティ計画立案、プロビジョニング、ユーティリゼイション（使用率）
		1.3.8	需要予測、キャパシティ計画立案、プロビジョニング：需要源を説明する

カテゴリー	トピック	Ref.	ナレッジ/タスク項目
		1.3.9	需要予測、キャパシティ計画、プロビジョニング：キャパシティ計画とプロビジョニングの関係を説明する
		1.3.10	需要予測、キャパシティ計画、プロビジョニング：プロビジョニング、キャパシティ目標、サービスレスポンスの関係を説明する
	1.4 SREと顧客リライアビリティ・エンジニアリング（CRE:Customer Reliability Engineering）	1.4.1	用語の定義：顧客リライアビリティ・エンジニア（CRE:Customer Reliability Engineer）
		1.4.2	SREとCREの関係を説明する
2.SREプラクティスの構築	2.1 DevOpsチーム	2.1.1	サイロを壊す: サイロ化思考の結末を説明する
		2.1.2	サイロを壊す：部分最適化と全体最適化を比較する
		2.1.3	DevOpsのチーム化：最適なSREチームのトポロジーを説明する
		2.1.4	DevOpsのチーム化：チームファーストの概念を説明する
		2.1.5	DevOpsのチーム化：用語の定義：コンウェイの法則
		2.1.6	プロダクトチームとプラットフォームチーム：用語の定義：プロダクトチーム
		2.1.7	プロダクトチームとプラットフォームチーム：ストリーム・アラインド・チームの概念を説明する
		2.1.8	プロダクトチームとプラットフォームチーム：用語の定義：プラットフォームチーム
		2.1.9	プロダクトチームとプラットフォームチーム：X-asa-serviceチームのインタラクション・モードを説明する
		2.1.10	プロダクトチームとプラットフォームチーム：ストリーム・アラインド・チームにおけるセルフ・サービスの概念を説明する
	2.2 SREの主な特徴	2.2.1	説明責任と完全性：SREの文脈におけるサービスオーナーシップの概念を説明する
		2.2.2	説明責任と完全性：用語の説明：シングル・ソース・オブ・トゥルース

カテゴリー	トピック	Ref.	ナレッジ/タスク項目
		2.2.3	問題解決：サービスオーナーシップがどのように問題オーナシップにつながるかを説明する
		2.2.4	問題解決：用語の定義：根本原因分析
		2.2.5	フレキシブルかつオープン：SREと専門チームとのチーム間コラボレーションの目的を説明する
		2.2.6	フレキシブルかつオープン：SREにおける実験の目的を説明する
		2.2.7	トイルの撲滅：SREが自動化に注力する目的を説明する
		2.2.8	「ヒーローはいない」：SREの文脈でのヒーローという言葉を説明する
		2.2.9	「ヒーローはいない」：用語の定義：共有されたチームの説明責任
	2.3 求められる主なスキルの種類	2.3.1	テクニカル：用語の定義：ソフトウェア・エンジニアリング、ネットワーク・エンジニアリング、プロダクト・エンジニアリング
		2.3.2	分析的：システムの信頼性をモニタリングやレポートと関連付ける
		2.3.3	分析的：ピア・コードレビューを説明する
		2.3.4	「ソフト」スキル： SREにとってのコミュニケーションの重要性を説明する
		2.3.5	「ソフト」スキル：コラボレーションとチームワークがどのようにSREの結果を高めるかを説明する
		2.3.6	時間管理：SRE の優先順位付けの判断に影響を与える要因を説明する
		2.3.7	時間管理：SRE が時間をどのように使うかについて決定することを説明する
	2.4 SREプラクティスの成熟化	2.4.1	客観的なメトリクス：データ駆動型測定の価値を説明する
		2.4.2	客観的なメトリック：明確な目標によってコミュニケーションがどのように強化されるかを説明する
		2.4.3	長期的なベネフィットへのフォーカス：長期的な解決策と短期的な教訓の全体的な影響を比較する
		2.4.4	インクリメンタル改善：継続的改善の原則を説明する

カテゴリー	トピック	Ref.	ナレッジ/タスク項目
		2.4.5	インクリメンタル改善：用語の定義：改善/KAIZEN
		2.4.6	ランディング対ローンチ：用語の定義：ランディング・ゾーン（着陸地帯）とローンチ
		2.4.7	ランディング対ローンチ：デプロイメント・パイプラインとサービス・ライフサイクルを比較する
3.成功の定量化	3.1 サービスレベル指標（SLI）	3.1.1	定量的なサービス測定基準：用語の定義：可用性、信頼性、耐久性
		3.1.2	定量的なサービス測定基準：SLI の共通定義の重要性を説明する
		3.1.3	アグリゲーションと標準化：共通のSLIを使用した様々なタイプのサービスを説明する
	3.2 サービスレベル目標（SLO）	3.2.1	サービスレベルの目標値：サービスレベルの収穫逓減の概念を説明する
		3.2.2	サービスレベルの目標値：SLOの設定におけるプロダクト管理の役割を説明する
		3.2.3	顧客の期待：SLOで顧客の期待を設定するプロセスを説明する
		3.2.4	顧客の期待：SREがSLOを超える努力をしない理由を説明する
	3.3 エラーバジェット	3.3.1	"信頼性の欠如" の限界：エラーバジェットとSLOを比較する
		3.3.2	"信頼性の欠如 "の限界：エラーバジェットが、DevとOpsの間の緊張関係にどのような影響を与えるかを説明する
		3.3.3	イノベーションと信頼性のバランス：エラーバジェットがイノベーションと信頼性のバランスをとる方法を説明する
		3.3.4	イノベーションと信頼性のバランス：エラーバジェットがどのようにサービス信頼性を駆動するかを説明する
		3.3.5	イノベーションと信頼性のバランス：エラーバジェットを超えた場合のSREのアクションを説明する
		3.3.6	イノベーションと信頼性のバランス：エラーバジェットの計算方法を説明する
	3.4 サービスレベルアグリーメント（SLA）	3.4.1	SLOとSLAを比較する

カテゴリー	トピック	Ref.	ナレッジ/タスク項目
		3.4.2	SLAに内在する潜在的な影響とペナルティを説明する
	3.5 コンプライアンスのためのモニタリング	3.5.1	サービスレビューとポストモーテムの概念を比較する
		3.5.2	自動化されたモニタリングとレポートが、どのように可用性の向上に貢献するかを説明する
4.SRE「オンコール」プロセス	4.1 SREの作業負荷とプレッシャーの軽減	4.1.1	割り込みへの対処：用語の定義：SREの文脈における割り込み
		4.1.2	割り込みへの対処：割り込みとトイルとの関係を説明する
		4.1.3	心理的安全性：SREに過度なプレッシャーを与える可能性があることを説明する
		4.1.4	心理的安全性：非難のないカルチャのパフォーマンス上の便益を説明する
	4.2 インシデントのエスカレーションパスの明確化	4.2.1	サポートレベル：従来のインシデント解決/エスカレーションのプロセスを説明する
		4.2.2	サポートレベル：用語の定義：インシデントの文脈におけるオーナーシップ
		4.2.3	サポートレベル：従来のエスカレーションによってオーナーシップがどのように変更されるか/されないかを説明する
		4.2.4	スウォーミング 対 古典的なエスカレーション：用語の定義：インシデントレスポンスの文脈におけるスウォーミング
		4.2.5	スウォーミング 対 古典的なエスカレーション：スウォーミングと従来のエスカレーションの手順を比較する
		4.2.6	スウォーミング 対 古典的なエスカレーション：スウォーミングと改善/カイゼンチームを比較する
	4.3 明確に定義されたインシデント管理手順	4.3.1	インシデント・コマンド：用語の定義：インシデント・コマンダー
		4.3.2	インシデント・コマンド：よく設計されたインシデント管理プロセスにおける明確な役割を説明する
		4.3.3	インシデント・コマンド：シフト間の明確な稼働中のハンドオフの目的を説明する
		4.3.4	運用作業：インシデント解決における運用作業の目的を説明する

カテゴリー	トピック	Ref.	ナレッジ/タスク項目
		4.3.5	運用作業：トイルの文脈での運用作業と、インシデント対応の文脈での運用作業を比較する
		4.3.6	コミュニケーション：インシデント対応の文脈で、単一コミュニケーションポイントの概念を説明する
		4.3.7	コミュニケーション：インシデント対応におけるフォーカスされたコミュニケーションの重要性を説明する
		4.3.8	計画立案：より長期的なインシデントにおける計画立案の役割を説明する
		4.3.9	計画立案：計画立案における稼働中インシデント状態ドキュメントの役割を説明する
	4.4 非難のないポストモーテム	4.4.1	教訓という用語を説明し、「学んだ」ことの尺度として、どれだけの教訓がアクション／クローズされたかを説明する
		4.4.2	教訓とナレッジ管理の関係を説明する
		4.4.3	ポストモーテム・カルチャの概念を説明する
		4.4.4	非難のないポストモーテムの目的を説明する
		4.4.5	ゲーミフィケーションという用語の定義と、それがどのように現場のサイト・リライアビリティ・エンジニアリングのスキルと知識を高めることができるかを説明する
5.リリース・エンジニアリングにおけるSRE	5.1 ローンチ・コーディネーション	5.1.1	SREとローンチ・コーディネーション・エンジニア (LCE:Launch Coordination Engineers)を比較する
		5.1.2	LCEの主な特徴を説明する
		5.1.3	インターネットを利用した企業と従来の企業の立ち上げサイクルを比較
	5.2 プロダクション・レディネス・ レビュー (Product Readiness Review)	5.2.1	用語の定義： プロダクション・レディネス・レビュー (Product Readiness Review)
		5.2.2	プロダクト・レディネスの重要な側面を説明する
		5.2.3	ランブックの概念と実験を比較する
		5.2.4	早期エンゲージメント・モデルを説明する
	5.3 リリース管理	5.3.1	用語の定義：カナリア・リリースとリリース・トレイン
		5.3.2	A/Bテストとブルーグリーン・デプロイメント、カナリア・リリースとの違いを説明する

カテゴリー	トピック	Ref.	ナレッジ/タスク項目
		5.3.3	ブルー/グリーン・デプロイメントの基本的な概念を説明する
		5.3.4	リリース・トレインの一部としてフィーチャー・フラグがどのように使用されるかを説明する
	5.4 デプロイメント・パイプラインの構築	5.4.1	Gitのアーキテクチャと機能を説明する
		5.4.2	Mavenのアーキテクチャと機能を説明する
		5.4.3	Junitのアーキテクチャと機能を説明する
		5.4.4	Jenkinsのアーキテクチャと機能を説明する
		5.4.5	Puppetのアーキテクチャと機能を説明する
	5.5 構成管理	5.5.1	コンテナ型構造の利点を説明する
		5.5.2	コンテナと仮想マシンを比較する
		5.5.3	Dockerのアーキテクチャと機能を説明する
		5.5.4	用語の定義：コンテナ・クラスタ
		5.5.5	Kubernetesのアーキテクチャと機能を説明する
6.事業継続におけるSRE	6.1 アフター・デプロイメント	6.1.1	DevOpsの継続的モニタリングのケイパビリティとスコープを説明する
		6.1.2	可観測性とシステムモニタリングの違いを説明する
		6.1.3	継続的モニタリングがシステム停止のMTTRにどのような影響を与えるかを説明する
		6.1.4	Nagios®とELK Stackのアーキテクチャと機能を説明する
	6.2 サービス継続性	6.2.1	マイクロ・サービス・アーキテクチャ：マイクロ・サービス・アーキテクチャとモノリシック・アプリケーションを比較する
		6.2.2	災害復旧：災害復旧 / デザイリカバリーという用語を説明する
		6.2.3	災害復旧：回復がバックアップより難しい理由を説明する
		6.2.4	災害復旧：事業継続計画を説明する
		6.2.5	災害復旧：DiRT基本演習を説明する
		6.2.6	災害復旧：DiD(Defense in depth)の概念を説明する
		6.2.7	システム・レジリエンス：用語の定義：システム・レジリエンス

カテゴリー	トピック	Ref.	ナレッジ/タスク項目
		6.2.8	システム・レジリエンス：高可用性の概念を説明する
		6.2.9	カオス・エンジニアリング：Chaos Monkeyのようなツールがどのように事業継続性を高めるかを説明する
		6.2.10	カオス・エンジニアリング：NetflixのSimian Armyを説明する

6. テスト仕様

PeopleCert DevOps Site Reliability Engineer試験は、以下の構成で6つのセクションから構成されます。

カテゴリー	説明	試験(%)
1.0	Site Reliability Engineering（SRE）とは？	25.0%
2.0	SRE プラクティスの構築	20.0%
3.0	成功の定量化	15.0%
4.0	SRE「オンコール」プロセス	15.0%
5.0	リリース・エンジニアリングにおける SRE	15.0%
6.0	事業継続における SRE	10.0%
	合計	**100.0%**

7. 推薦 参考文献等

1. A/B testing image created by Maxime Lorant, CC BY-SA 4.0<https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons, October 29, 2015. https://commons.wikimedia.org/wiki/File:A-B_testing_simple_example.png
2. Adams, Rich. “Incident Response Training.” PagerDuty University. Response.pagerduty.com. Accessed December 23, 2021. https://response.pagerduty.com/training/courses/incident_response/
3. Altexsoft. “DevOps vs Site Reliability Engineering: Concepts, Practices, and Roles,” Altexsoft.com (blog), last modified November 12, 2020, accessed October 4, 2021. https://www.altexsoft.com/blog/devops- vs- site- reliability- engineering/
4. Atlassian. “Incident Communication Best Practices.” Atlassian.com. Accessed November 1, 2021. https://www.atlassian.com/incident- management/incident-communication
5. Apache Maven Project. “Maven – Introduction.” Mavenapache.org. Last modified January 5, 2022. Accessed July 15, 2021. https://maven.apache.org/what- is-maven.html
6. Arrington, Eric. “What’s the Difference between VMs and Containers?” Akfpartners.com (blog), last modified May 29, 2019, accessed July 7, 2021. https://akfpartners.com/growth- blog/vms vs- containers
7. AXELOS. “ITIL ® 4 Foundation Glossary.” Axelos.com. Last modified 2021. Accessed November 17, 2021. https://www.axelos.com/getmedia/5896d51f- ab6c- 4843-992b- 4f045eab0875/ITIL- 4 Foundation- glossary_v0_22.aspx
8. ——. ITIL ® Foundation, ITIL 4 edition. London: TSO, 2019.
9. ——. ITIL ® Glossary. Axelos.com. London: TSO, 2011. https://www.axelos.com/getmedia/d746e4ed- 1d1c- 4644- b99d b760b5946452/ITIL_2011_Glossary_GB- v1- 0.aspx
10. ——. ITIL ® 4: High Velocity IT. London: TSO, 2020.
11. ——. *ITIL ® 4 Practice Guide: Availability Management.*AXELOS, 2020.
12. ——. ITIL ® 4 Practice Guide: Problem Management. AXELOS, 2020.
13. ——. *ITIL ® 4 Practice Guide: Risk Management.*AXELOS, 2019.
14. ——. ITIL ® 4 Practice Guide: Service Continuity Management. AXELOS, 2020.
15. ——. ITIL ® 4 Practice Guide: Service Level Management. AXELOS, 2020

16. Barnett, Taylor. “Writing Runbook Documentation When You’re an SRE.” Transposit.com (blog), last modified January 30, 2020. Accessed December 22, 2021. https://www.transposit.com/blog/2020.01.30- writing- runbook- documentation- when- youre- an sre/

17. Benčević, Marin. “Accidental and Essential Complexity – Programming Word of the Day.” Background Thread. Medium.com. Last modified July 10, 2018. Accessed November 11, 2021. https://medium.com/background- thread/accidental- and- essential- complexity programming- word- of- the- day- b4db4d2600d4

18. “Beware of Local Optimisation.” Sketchplanations.com. Accessed October 18, 2021. https://sketchplanations.com/beware- of- local- optimisation

19. Beyer, Betsy, Chris Jones, Jennifer Petoff, and Niall Richard Murphy. “Communication and Collaboration” in “ Site Reliability Engineering” How Google Runs Production Systems. Sebastopol, CA: O’Reilly Media, 2016. Accessed December 24, 2021. https://sre.google/sre book/communication- and- collaboration/

20. ——. “Service Level Objectives” in “ Site Reliability Engineering” How Google Runs Production Systems. Sebastopol, CA: O’Reilly Media, 2016. Accessed December 24, 2021. https://sre.google/sre- book/service- level- objectives/

21. ——. “Release Engineering” in “ Site Reliability Engineering” How Google Runs Production Systems. Sebastopol, CA: O’Reilly Media, 2016. Accessed December 24, 2021. https://sre.google/sre- book/release- engineering/

22. ——. “On-Call” in “ Site Reliability Engineering” How Google Runs Production Systems. Sebastopol, CA: O’Reilly Media, 2016. Accessed October 14, 2021. https://sre.google/workbook/on- call/

23. ——. “Engagement Model” in “ Site Reliability Engineering” How Google Runs Production Systems. Sebastopol, CA: O’Reilly Media, 2016. Accessed December 24, 2021. https://sre.google/workbook/engagement- model/

24. ——. “Introduction” in “ Site Reliability Engineering” How Google Runs Production Systems. Sebastopol, CA: O’Reilly Media, 2016. Accessed December 24, 2021. https://sre.google/sre book/introduction/

25. ——. “Dealing with Interrupts” in “ Site Reliability Engineering” How Google Runs Production Systems. Sebastopol, CA: O’Reilly Media, 2016. Accessed December 24, 2021. https://sre.google/sre- book/dealing- with- interrupts/

26. ——. “Evolving SRE Engagement Model” in “ Site Reliability Engineering” How Google Runs Production Systems. Sebastopol, CA: O’Reilly Media, 2016. Accessed December 24, 2021. https://sre.google/sre- book/evolving- sre- engagement- model/

27. ——. “Managing Incidents” in “ Site Reliability Engineering” How Google Runs Production Systems. Sebastopol, CA: O’Reilly Media, 2016. Accessed December 24, 2021. https://sre.google/sre- book/managing- incidents/

28. ——. “Postmortem Culture” in “ Site Reliability Engineering” How Google Runs Production Systems. Sebastopol, CA: O’Reilly Media, 2016. Accessed December 24, 2021. https://sre.google/sre- book/postmortem- culture/

29. ——. “Reliable Product Launches” in “ Site Reliability Engineering” How Google Runs Production Systems. Sebastopol, CA: O’Reilly Media, 2016. Accessed December 24, 2021. https://sre.google/sre- book/reliable- product- launches/

30. Bigelow, Stephen J. “Release Management.” Techtarget.com. Last modified June 12, 2009. Accessed December 24, 2021. https://searchitchannel.techtarget.com/definition/release management

31. Binette, Elisa. “Best Practices for Setting SLOs and SLIs for Modern, Complex Systems,” Newrelic.com (blog), last modified October 31, 2018, accessed October 14, 2021. https://newrelic.com/blog/best- practices/best- practices- for- setting-slos- and- slis- for- modern complex- systems

32. Bird, Jim. DevOpsSec. Sebastopol, CA: O’Reilly Media. June 2016. https://www.oreilly.com/library/view/devopssec/9781491971413/ch04.html

33. Bose, Shreya. “What Is Continuous Monitoring in DevOps?” Browserstack.com. Last modified December 8, 2020/ Accessed December 31, 2021. https://www.browserstack.com/guide/continuous- monitoring- in- devops

34. Bramley, Alex. “Are we there yet? Thoughts on assessing an SRE team’s maturity,” Google.com (blog). Last modified June 18, 2021. Accessed October 22, 2021. https://cloud.google.com/blog/products/devops- sre/evaluating- where- your-team- lies- on- the sre- spectrum

35. Branson, Julie. “What Is a Product Team?” Crema.us (blog), last modified July 28, 2020, accessed November 11, 2021. https://www.crema.us/blog/what- is- a-product- team

36. Bravo, Marco. “Top Five Configuration Management Tools.” Opensource.com. Last modified December 12, 2018. Accessed September 7, 2021. https://opensource.com/article/18/12/configuration- management- tools

37. Buchanan, Ian. “Team Topologies: How Four Fundamental Topologies Influence a DevOps Transformation.” Atlassian.com. Accessed December 7, 2021. https://www.atlassian.com/devops/frameworks/team- topologies

38. Catalano, Thomas. “Theory of Constraints (TOC).” In Application of Project Management Principles to the Management of Pharmaceutical R&D Projects, 13–15. Cham: Springer International Publishing, 2020.

39. Casperson, Matthew. “Runbooks Best Practices,” Octopus.com (blog), last modified March 9, 2020, accessed December 23, 2021. https://octopus.com/blog/runbooks- best- practices

40. CAST. “What Is Software Engineering?” Castsoftware.com. Accessed November 9, 2021. https://www.castsoftware.com/glossary/what- is- software- engineering- definition- types- of basics- introduction

41. CBS Denver. “Cyber Security Students Learn Strategy with Video Games,” Youtube.com, last modified May 24, 2019, accessed September 8, 2021. https://www.youtube.com/watch?v=5R8Gq0wxdg4

42. Chris. “Maven Overview.” Dev.vividbreeze.com. Last modified March 28, 2018. Accessed July 6, https://dev.vividbreeze.com/maven- overview/

43. CISQ. “DevOps Software Quality.” IT-cisq.org. Accessed July 15, 2021. https://www.it cisq.org/use- cases/devops- code- quality.htm

44. Civil Service College. “Understanding the Differences between Teamwork and Collaboration.” Civilservicecollege.org.uk. Last modified August 30, 2018. Accessed November 9, 2021. https://www.civilservicecollege.org.uk/news- understanding- the- differences- between teamwork- and- collaboration- 203

45. “Continuous Delivery for PE Architecture.” Puppet.com. Accessed January 3, 2022. https://puppet.com/docs/continuous- delivery/3.x/cd_architecture.html

46. Cowling, James. “Engineering for Data Durability.” In Seeking SRE: Conversations about Running Production Systems at Scale. Sebastopol, CA: O’Reilly Media, 2018.

47. Das, Abhishek Dey and Sudhanshu Iyer. “Git Introduction, PowerPoint, SlideServe.com. Uploaded September 4, 2014. Accessed June 2021. PPT - Git Introduction PowerPoint Presentation, free download - ID:3927183 (slideserve.com)

48. DevMountain. “Git vs. GitHub: What’s the Difference?” Devmountain.com (blog), accessed July 6, 2021. https://blog.devmountain.com/git- vs- github- whats- the- difference/

49. “DevOps Tech: Version Control.” Google Cloud Architecture Center. Accessed January 7, 2022. https://cloud.google.com/architecture/devops/devops- tech- version- control

50. “DevOps Topologies.” Devopstopologies.com. Accessed June 17, 2021. https://web.devopstopologies.com/. Licensed under CC BY-SA.

51. Dharmalingam, N. “Chef vs Puppet vs Ansible,” Whizlabs.com (blog), last modified February 11, 2020, accessed January 4, 2022. https://www.whizlabs.com/blog/chef- vs- puppet- vs- ansible/

52. Docker docs. “Docker Overview.” Docs.docker.com. Accessed July 16, 2021. https://docs.docker.com/get- started/overview/

53. Donahue, Jamie. “NUS-ISS Learning Day 2019-Site Reliability Engineering – the Modern Method for Digital Infra/Ops Management.” Institute of Systems Science, National University of Singapore, August 13, 2019. Slideshare.net. Accessed December 17, 2021. https://www.slideshare.net/ISS- NUS/nusiss- learning- day-2019site- reliability- engineering- the modern- method- for- digital- infraops-management

54. DuMoulin, Troy. “Configuration Management – A Rose by Any Other Name,” Pinkelephant.com (blog), January 10, 2020. https://blog.pinkelephant.com/blog/configuration- management- a rose- by- any-other- name

55. “EC2: High Availability with EBS Snapshots,” N2w.com (blog), last modified October 22, 2013, accessed December 23, 2021. https://n2ws.com/blog/aws- ec2-backup/ec2- poor- mans- high availability- ebs- snapshots

56. “Engineering.” Dictionary. com. Accessed November 20, 2021. https://www.dictionary.com/browse/engineering

57. Everitt, Jessica. “Understanding Capacity Utilization Rates,” Wrike.com,(blog), last modified January 20, 2020, accessed October 13, 2021. https://www.wrike.com/blog/why- capacity utilization- rates- key- profitability/

58. “Feature Flags.” Microsoft.com. Last modified May 11, 2021. Accessed January 10, 2022. https://docs.microsoft.com/en- us/dotnet/architecture/cloud- native/feature-flags

59. Feliciano, Dan. “Why Are Goals and Objectives Important?” Fastcompany.com. Last modified April 9, 2008. Accessed November 16, 2021. https://www.fastcompany.com/795028/why- are goals- and- objectives- important

60. Fowler, Adam. “Root Cause Analysis.” Techtarget.com. Last modified April 27, 2018. Accessed October 21, 2021. https://searchitoperations.techtarget.com/definition/root- cause- analysis

61. Fowler, Martin. “Software Architecture Guide.” MartinFowler.com. Last modified August 1, 2019. Accessed May 5, 2021. https://martinfowler.com/architecture/

62. Fowler, Susan. Production-Ready Microservices: Building Standardized Systems across an Engineering Organization. Sebastopol, CA: O’Reilly Media, 2016.

63. Frame, Jess, Lenton, Anthony, Thurgood, Steven, Tolchanov, Anton, and Trdin, Nejc. “Monitoring.” Sre.Google. Accessed January 7, 2022. https://sre.google/workbook/monitoring

64. Fronczak, Sylvia. “Service Ownership: What It Really Means and How to Achieve It,” Opslevel.com (blog), last modified April 9, 2021, accessed December 8, 2021. https://www.opslevel.com/blog/service- ownership- what- it- really- means- and- how- to- achieve it/

65. Gaba, Ishan. “Managing Your CI/CD Continuous Improvement With Pipelines” Simplilearn.com. Last modified November 3, 2021. Accessed July 16, 2021. https://www.simplilearn.com/tutorials/jenkins- tutorial/ci- cd- pipeline

66. Git. “What Is Git?” Git-scm.com. Accessed July 15, 2021. https://git scm.com/book/en/v2/Getting- Started- What- is- Git%3F

67. Goossaert, Emmanuel. “Service Ownership Checklist,” Codecapsule.com (blog), last modified November 12, 2017, accessed December 6, 2021. https://codecapsule.com/2017/11/12/service- ownership- checklist/

68. Gosset, Stephen. “What Is a Site Reliability Engineer? What Does an SRE Do?” Builtin.com. Last modified October 25, 2021. Accessed November 9, 2021. https://builtin.com/software engineering- perspectives/site- reliability- engineer

69. Grey, Matthew. “Site Reliability Engineering” Servian.dev. Last modified January 7, 2019. Accessed October 20, 2021. https://servian.dev/site- reliability- engineering- 8505e3daa069

70. Guernion, Yann. “Automating Site Reliability Engineering,” Broadcom.com (blog), last modified August 4, 2020, accessed December 20, 2021. https://community.broadcom.com/enterprisesoftware/blogs/yann guernion1/2020/08/04/automating- site- reliability- engineering

71. Gunja, Saif. “What Is Site Reliability Engineering? And What Do Site Reliability Engineers Do?” Dynatracenews.com (blog), last modified February 4, 2021, accessed December 20, 2021. https://www.dynatrace.com/news/blog/what- is- site- reliability- engineering/

72. Güntaş, İbrahim. “5 Common Incident Response Problems and Their Solutions.” Medium.com. Last modified November 16, 2017. Accessed December 17, 2021. https://medium.com/@ibrahimguntas/5- common- incident- response- problems- and- their solutions- 6130ca7b43c0

73. Gupta, Pankaj. “A Thousand Salutes to SREs: Unsung Heroes of the Pandemic.” Devops.com. Last modified June 22, 2020. Accessed October 22, 2021. https://devops.com/a- thousand salutes- to- sres- unsung- heroes- of- the- pandemic/

74. Hertvik, Joe. “Service Availability: Calculations and Metrics, Five 9s, and Best Practices.” BMC.com. Last modified July 8, 2020. Accessed November 18, 2021. https://www.bmc.com/blogs/service- availability- calculation- metrics/

75. Heslin, Kevin. “How to Avoid Outages: Try Harder!” Journal. Uptimeinstitute.com. Last modified September 23, 2019. Accessed October 12, 2021. https://journal.uptimeinstitute.com/how- to avoid- outages- try- harder/

76. “High Availability.” Avinetworks.com. Last modified April 21, 2017. Accessed November 18, 2021. https://avinetworks.com/glossary/high- availability/

77. Humble, Jez. “Architecture.” Continuousdelivery.com. Modified 2017. Accessed May 5, 2021. https://continuousdelivery.com/implementing/architecture/ All content licensed under the Creative Commons Attribution-Share Alike 3.0 United States License (CC BY-SA 3.0 US).

78. Humble, Jez and David Farley. “What Is a Deployment Pipeline?” in Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Boston, MA: AddisonWesley Professional, 2010. Accessed via Informit.com, November 15, 2021. https://www.informit.com/articles/article.aspx?p=1621865&seqNum=2

79. Hunter, Tatum. “How to Manage Overzealous ‘Hero’ Engineers.” Builtin.com. Updated October 22, 2021. Accessed October 22, 2021. https://builtin.com/software- engineering perspectives/engineering- team- heroing

80. IBM Cloud Education. “Aiops.” Ibm.com. Accessed October 29, 2021. https://www.ibm.com/cloud/learn/aiops

81. ——. “Infrastructure as Code.” Ibm.com. Last modified December 2. 2019. Accessed July 8, https://www.ibm.com/cloud/learn/infrastructure- as- code

82. IT Revolution. “The Post-Incident Review.” Itrevolution.com. Last modified June 21, 2021. Accessed October 26, 2021. https://itrevolution.com/post- incident- review/

83. Jenkins. “Delivery Pipeline.” Jenkins.io. Accessed July 16, 2021. https://plugins.jenkins.io/delivery- pipeline- plugin/

84. Jewkes, Henry. “Site Reliability and Experimentation: They Go Hand in Hand,” Split.io (blog), last modified October 15, 2020, accessed October 22, 2021. https://www.split.io/blog/sre- feature flags- experimentation/

85. Jones, Mike. “What Are Microservices and Why Would I Want Them? Microservices in Three Minutes,” Youtube.com, last modified May 4, 2016, accessed September 14, 2021. https://www.youtube.com/watch?v=CKL3fV5UR8w

86. Kareem, Kameerath. “Reliability Monitoring for Improved Digital Experience,” Catchpoint.com (blog), accessed November 9, 2021. https://www.catchpoint.com/blog/reliability- monitor improved- digital- experience

87. “Kubernetes Controllers and Services.” Edureka. Youtube.com, June 14, 2021. Last modified June 14, 2021. Accessed August 19, 2021. https://www.youtube.com/watch?v=fblsY6uQ8Hs

88. Lock, Daniel. “4 Principles of Continuous Improvement,” Daniel Lock Consulting (blog), last modified November 15, 2016, accessed December 20, 2021. https://daniellock.com/4 principles- of- continuous- improvement/

89. Loewengruber, Joe. “Automate Application Monitoring.” IBM Garage Methodology. Accessed October 21, 2021. https://www.ibm.com/garage/method/practices/manage/practice_automated_monitoring

90. LogicManager Analyst Team. “What Does a Business Continuity Plan Typically Include? [Complete Guide].” Logicmanager.com. Last modified May 27, 2021. Accessed November 17, 2021. https://www.logicmanager.com/erm-software/2021/05/27/what- does- a- business continuity- plan- typically- include/

91. Lowe, Hunter. “5 Areas Supply Chain Optimization Improves” Selecthub.com (blog), last modified June 11, 2021. Accessed October 18, 2021. https://www.selecthub.com/supply- chain management/5- areas- supply- chain-optimization- improves/

92. MacVittie, Lori. “Automation versus Orchestration.” Devops.com. Last modified April 10, 2014. Accessed September 3, 2021. https://devops.com/automation- versus-orchestration/

93. Mandel, Mark and Francesc Campoy Flores. “Customer Reliability Engineering with Luke Stone.” Google Cloud Platform Podcast. May 10, 2017. Accessed November 29, 2021. https://www.gcppodcast.com/post/episode- 72- customer- reliability-engineering- with- luke stone/

94. Mark Mandel, Francesc Campoy Flores, Luke Stone. “Customer Reliability Engineering with Luke Stone.” Google Cloud Platform Podcast, April 12, 2017. Accessed November 29, 2021. https://www.gcppodcast.com/post/episode- 72-customer- reliability- engineering- with- luke stone/

95. Matrix Management Wiki. “2P. How Accountability Works on a Team.” MMI Inc. Accessed December 21, 2021. https://wiki.matrixmanagementinstitute.com/accountability- key/2p- how accountability- works- on- a- team/ Matrix Management Wiki. “2G. Team Accountability.” MMI Inc. Last modified October 8, 2020. Accessed December 21, 2021. https://wiki.matrixmanagementinstitute.com/accountability key/2g- team-accountability/

96. Mesz, Sol. “Law of Diminishing Returns, Design, and Decision-Making,” SolMesz.com (blog), last modified November 12, 2020, accessed December 17, 2021. https://www.solmesz.com/blog en/law- diminishing- returns- design- decision-making

97. Mohan, Mukund. “The Difference between Metrics-Driven and Data-Driven Startups.” Bestengagingcommunities.com. Last modified September 22, 2015. Accessed November 15, 2021. https://bestengagingcommunities.com/2015/09/22/the- difference- between-metrics driven- and- data- driven- startups/

98. Mukkara, Uma. “Chaos Engineering for Cloud Native.” Thenewstack.io. Last modified October 14, 2021. Accessed December 20, 2021. https://thenewstack.io/chaos-engineering- for- cloud native/

99. Müller, Michael. “What Is CRE, and What Does It Have to Do with SRE?” ContainerSolutions.com (blog), last modified August 9, 2020, accessed October 13, 2021. https://blog.container- solutions.com/what- is- cre- and- what- does- it-have- to- do- with- sre

100. Murphy, Niall, John Looney, and Michael Kacirek. “The Evolution of Automation”. Last modified 2017. Accessed December 20, 2021. https://sre.google/sre-book/automation- at- google/

101. N-able. “Key Differences between a Disaster Recovery Plan versus a Business Continuity Plan,” N-able.com (blog), last modified September 23, 2020, accessed December 18, 2021. https://www.n- able.com/blog/disaster- recovery- plan- vs-business- continuity- plan

102. Nolle, Tom. “The 4 Rules of a Microservices Defense-in-Depth Strategy.” SearchAppArchitecture, TechTarget. Last modified December 18, 2020. Accessed December 23, https://searchapparchitecture.techtarget.com/tip/The- 4- rules- of-a- microservices defense- in- depth- strategy

103. Nukala, Shylaja and Vivek Rau. “Why SRE Documents Matter.” ACMQUEUE. Last modified October 4, 2018. Accessed December 22, 2021. https://queue.acm.org/detail.cfm?id=3283589

104. Overby, Stephanie, Lynn Greiner, and Lauren Gibbons Paul. “What Is an SLA? Best Practices for Service-Level Agreements.” CIO.com. Last modified July 5, 2017. Accessed October 21, 2021. https://www.cio.com/article/2438284/outsourcing- sla-definitions- and- solutions.html

105. Patrizio, Andy. “The Biggest Risk to Uptime? Your Staff.” Networkworld.com. Last modified October 9, 2019. Accessed November 11, 2021. https://www.networkworld.com/article/3444762/the- biggest- risk- to- uptime-your- staff.html

106. Perveez, Sayeda Haifa. “What Is Git: Features, Command and Workflow in Git.” Simplilearn.com. Simplilearn, May 4, 2020. Last modified September 18, 2021. Accessed July 9, 2021. https://www.simplilearn.com/tutorials/git- tutorial/what- is-git

107. Pink Elephant. Lean IT Foundation certification course. Burlington, ON: Pink Elephant, 2021.

108. Plutora. “What Is an Agile Release Train?” Plutora.com (blog), last modified November 23, 2020. https://www.plutora.com/blog/agile- release- train

109. “Production Readiness Review (PRR).” Acqnotes.com. Last modified June 22, 2021. Accessed December 22, 2021. https://acqnotes.com/acqnote/tasks/production-readiness- review- 2

110. “Puppet Enterprise – All about,” Itsysintegration.net (blog), accessed January 3, 2022. https://blog.itsysintegration.net/configuration- management- cm- tools- and-solution/puppet enterprise

111. Puricica, Cristian-Antonio. “Demystifying Recovery Objectives,” Veeam.com (blog), last modified October 9, 2017, accessed December 23, 2021. https://www.veeam.com/blog/rto- rpo definitions- values- common- practice.html

112. “R/Sre – How Does Your Team Handle Interrupt Work?” Reddit.com. Accessed October 28, 2021. https://www.reddit.com/r/sre/comments/p2572v/how_does_your_team_handle_interrupt_w ork/

113. RabIT software engineering. “Peer Code Review: What Is It and Why Do You Need It?” RabITse.com (blog), last modified October 31, 2018, accessed November 5, 2021. https://www.rabitse.com/blog/peer- code- review/#

114. Raina, Ajeet. “Test-Drive Continuous Integration Pipeline Using Docker, Jenkins, and GitHub under $0.” Collabnix.com. Last modified March 3, 2018. Accessed July 6, 2021. https://collabnix.com/5- minutes- to- continuous- integration- pipeline-using- docker- jenkins github- on- play- with- docker- platform/

115. Raza, Muhammad. “Reliability vs Availability: What’s the Difference?” Bmc.com (blog), last modified May 13, 2020, accessed December 6, 2021. https://www.bmc.com/blogs/reliability- vs availability/

116. Rendell, Mark. “Team Topologies Book Summary – Part 2 of 3: Topologies and Interaction Modes.” Markosrendell.wordpress.com. Last modified February 4, 2020. Accessed December 7, https://markosrendell.wordpress.com/2020/02/04/team-topologies- book- summary- part 2- of- 3- topologies- and- interaction- modes/

117. Rensin, Dave. “Introducing Google Customer Reliability Engineering (CRE),” CloudGoogle.com (blog), last modified October 10, 2016, accessed November 29, 2021. https://cloud.google.com/blog/products/devops- sre/introducing- a- new-era- of- customer support- google- customer- reliability- engineering

118. Rizqi, Ashar. “How SRE Creates a Blameless Culture.” Devops.com. Last modified February 27, 2019. Accessed October 29, 2021. https://devops.com/how- sre-creates- a- blameless- culture/

119. Rowe, Sandra F. and Sharon Sikes. “Lessons Learned: Taking It to the Next Level.” Project Management Institute. Pmi.org. Last modified January 2006. Accessed December 17, 2021. https://www.pmi.org/learning/library/lessons- learned- next-level- communicating- 7991

120. Ruqayya, Noor-Ul-Anam. “Complete Guide to Service Level Objectives (SLOs) that Work.” Blameless.com. Last modified June 11, 2021. Accessed October 20, 2021. https://www.blameless.com/sre/service- level- objectives

121. Schiemann, Wulf. “Cloud Landing Zone Life Cycle Explained!” Meshcloud.io. Last modified June 8, 2020. Accessed December 20, 2021. https://www.meshcloud.io/2020/06/08/cloud- landing zone- lifecycle- explained/

122. Schnepp, Rob, Ron Vidal, and Chris Hawley. Incident Management for Operations. Sebastopol, CA: O’Reilly Media, 2017.

123. Simon, Becky. “A Winning Combination: Collaborative Teamwork Equals Teamwork and Collaboration.” Smartsheet.com. August 28, 2017. Last modified on July 19, 2021. Accessed October 21, 2021. https://www.smartsheet.com/collaborative-teamwork

124. Singh, Rhandeev, Sebastian Kirsch, Vivek Rau, and Betsy Beyer. “Reliable Product Launches at Scale.” Sre.Google. Last modified 2017. Accessed December 22, 2021. https://sre.google/sre book/reliable- product- launches/

125. “Single Source of Truth.” Dragon1.com. Accessed October 20, 2021. https://www.dragon1.com/concepts/single- source- of- truth

126. “Site Reliability Engineering.” Capgemini.com. Last modified August 7, 2020. Accessed October 22, 2021. https://www.capgemini.com/2020/08/site- reliability-engineering- 2/

127. Skelton, Matthew. “SRE in Practice: 5 Insights from Google’s Experience.” TechBeacon.com. December 16, 2019. Last modified December 16, 2019. Accessed October 13, 2021. https://techbeacon.com/enterprise- it/sre- practice- 5- insights googles- experience

128. Skelton, Matthew and Chris O’Dell. Continuous Delivery with Windows and.NET. Sebastopol, CA: O’Reilly Media, 2016. https://www.oreilly.com/library/view/continuous- delivery with/978 1492042327/

129. Skelton, Matthew and Manuel Pais. Team Topologies: Organizing Business and Technology Teams for Fast Flow. Portland, OR: IT Revolution Press, 2019.

130. Slimmon, Dan. “MTTR: Lower Isn’t Always Better,” Danslimmon.com (blog), last modified September 3, 2014, accessed October 14, 2021. https://blog.danslimmon.com/2014/09/03/mttr- lower- isnt- always- better/

131. Sloss, Benjamin Treynor. “The History of SRE.” Google Cloud Tech, July 15, 2020. https://youtu.be/1NF6N2RwVoc

132. Splunk. A Beginner’s Guide to Observability. Resources.enterprisetalk.com. Accessed July 14, https://resources.enterprisetalk.com/ebook/Splunk- TCO3- EN- 2.pdf

133. ——. “Resilience First: SRE and the Four Golden Signals of Monitoring.” Splunk.com. Accessed December 6,2021. https://www.splunk.com/en_us/observability/resources/guide- to- sre- and the-four- golden- signals- of- monitoring.html

134. Sridharan, Cindy. “The Three Pillars of Observability” in Distributed Systems Observability. O’Reilly Media, 2018, chapter 4. Accessed July 14, 2021. https://www.oreilly.com/library/view/distributed- systems observability/9781492033431/ch04.html

135. Srivastava, Anurag. Kibana 7 Quick Start Guide: Visualize Your Elasticsearch Data with Ease. Birmingham, England: Packt Publishing, 2019.

136. Sumo Logic Glossary. “Continuous Monitoring.” Sumologic.com. Accessed December 31, 2021. https://www.sumologic.com/glossary/continuous- monitoring/

137. Taylor, David. “Nagios Tutorial: What Is Nagios Tool? Architecture and Installation.” Guru99.com. Last modified December 25, 2021. Accessed December 31, 2021. https://www.guru99.com/nagios- tutorial.html

138. Team DiligenceVault. “Optimizing Returns on a Technical Debt Portfolio,” Diligencevault.com (blog), last modified November 30, 2017, accessed December 24, 2021. https://diligencevault.com/optimizing- returns- on- a- technical- debt- portfolio/

139. TestingXperts. “A Beginner’s Guide to Test Automation (2020),” Testingxperts.com (blog), last modified October 28, 2021, accessed January 11, 2022. https://www.testingxperts.com/blog/test- automation- guide

140. “The Elastic Stack: Elasticsearch, Kibana, Beats & Logstash” Elastic.co. Accessed August 18, 2021. https://www.elastic.co/elastic- stack/

141. TutorialsPoint. “Maven – Overview.” Tutorialspoint.com. Accessed July 15, 2021. https://www.tutorialspoint.com/maven/maven_overview.htm

142. ——. “Junit – Overview.” Tutorialspoint.com. Accessed July 16, 2021. https://www.tutorialspoint.com/junit/junit_overview.htm

143. Watts, Stephen. “Error Budgets Explained: Risk & Reliability in One Metric,” BMC (blog), Bmc.com, last modified November 12, 2020, accessed October 12, 2021. https://www.bmc.com/blogs/error- budgets/

144. ——. “Orchestration in SDLC for DevOps,” Bmc.com (blog, March 15, 2019. https://www.bmc.com/blogs/devops- orchestration/

145. “What Are Distributed Systems?” Splunk.com. Accessed November 11, https://www.splunk.com/en_us/data- insider/what- are- distributed- systems.html

146. “What Is an Incident Postmortem?” Pagerduty.com. Last modified May 8, 2017. Accessed October 25, 2021. https://www.pagerduty.com/resources/learn/incident- postmortem/

147. What Is Gamification? BiWorldwide.com. https://www.biworldwide.com/gamification/what- is gamification/

148. “What Is Golden Record Management and Why It Is Important?” Dataclarity.uk.com. Last modified June 22, 2020. Accessed October 20, 2021. https://www.dataclarity.uk.com/2020/06/22/what- is- golden- record- management- and- why- it is- important/

149. “What Is a Kubernetes Cluster?” Redhat.com. Last modified January 15. https://www.redhat.com/en/topics/containers/what- is- a- kubernetes- cluster

150. “What Is Kubernetes?” Kubernetes.io. Last modified July 23, 2021. Accessed August 18, 2021. https://kubernetes.io/docs/concepts/overview/what- is- kubernetes/

151. “What Is Root Cause Analysis (RCA)?” In Root Cause Analysis, 35–44. CRC Press, 2006. https://asq.org/quality- resources/root- cause- analysis

152. “What Is System Resilience.” Igi-Global.com. Accessed November 18, 2021. https://www.igi global.com/dictionary/cyber- threats- to- critical- infrastructure-protection/51260

153. Wildpaner, Michael and Gráinne Sheerin, Daniel Rogers, and Surya Prashanth Sanagavarapu (New York Times) with Adrian Hilton and Shylaja Nukala “SRE Engagement Model” Sre.google. Accessed October 20, 2021. https://sre.google/workbook/engagement- model/

154. Wilsenach, Rouan. “Running the Gauntlet: Setting Up Your First Deployment Pipeline.” TechBeacon.com. Accessed July 14, 2021. https://techbeacon.com/app-dev- testing/running gauntlet- setting- your- first- deployment- pipeline

155. Wilson, Bibin. “Jenkins Architecture Explained – Beginners Guide,” Devopscube.com (blog), last modified August 31, 2021, accessed July 6, 2021. https://devopscube.com/jenkins- architecture explained/

156. van der Heijden, Beatrice and André de Waal, Michael Weaver, and Tammy Day. “Silo-Busting: Overcoming the Greatest Threat to Organizational Performance.” ResearchGate. Sustainability 11, no. 23 (December 2019). Accessed November 29, 2021. https://www.researchgate.net/publication/337714303_Silo-Busting_Overcoming_the_Greatest_Threat_to_Organizational_Performance

157. von Grünberg, Kaspar. “What Is an Internal Developer Platform?” Humanitec.com (blog), last modified July 29, 2021, accessed October 20, 2021. https://humanitec.com/blog/what- is- an internal- developer- platform 用語集 （未査読バージョン：試験と未連動）

8. 用語集

Term	用語	Definition	定義
A/B Testing	**A/B テスト**	This is a technique used to test which of two alternative solutions is the best. It could be an alternative graphical design or functionality. The consumers are randomly split into two groups and test two different versions of the same feature.	これは、2 つの代替案のうちどちらがベストかをテストするために使用される手法です。それは、代替のグラフィックデザインであったり、機能性であったりします。消費者はランダムに 2 つのグループに分けられ、同じフィーチャーの 2 つの異なるバージョンをテストします。
Adaptive development model	**適合型開発モデル**	This is iterative and incremental in its approach to planning, developing, and delivering. These models use collaboration as a foundation to their success. These models are seen as circular and repetitive, because they break a large deliverable into smaller, more manageable deliverables, and they also circulate the development through the repeated activity of requirements, design, development, integration, and testing. Deployment is a separate phase that follows iterative development.	これは、計画、開発、提供のアプローチにおいて、イテレーティブでインクリメンタルなものです。これらのモデルはコラボレーションを成功の基盤としています。 これらのモデルは、大きな成果物をより小さく管理しやすいものに分割し、さらに要件、設計、開発、インテグレーション、テストというアクティビティを繰り返しながら開発を循環させるため、循環型かつ反復型と見なされています。展開（デプロイメント）は、イテレーティブな開発に続く独立したフェーズです。
Agile development	**アジャイル開発**	This is a project management style focused on the early delivery of business value, continuous improvement, scope flexibility, team input, and delivering well-tested products that reflect customer needs.	ビジネス価値の早期提供、継続的改善、スコープの柔軟性、チームの意見、顧客ニーズを反映したテスト済みのプロダクトの提供などに重点を置いたプロジェクト管理スタイルです。
AIOps	**AIOps**	This is the application of machine learning and big data to IT operations to receive continuous insights that provide continuous fixes and improvements via automation.	これは、機械学習とビッグ・データを IT 運用に適用し、継続的なインサイトを受け取り、自動化によって継続的な解決と改善を提供するものです。
Authentication	**認証(Authentication)**	This is verification that a characteristic or attribute that appears, or is claimed to be true, is in fact true.	これは、ある特性や属性が真実であると思われる、あるいは主張されることが、実際に真実であることを検証することです。

Term	用語	Definition	定義
Automated testing	**自動化テスト**	This is a software testing technique to test and compare the actual outcome with the expected outcome. This can be achieved by writing test scripts or using any automation testing tool. Test automation is used to automate repetitive tasks and other testing tasks that are difficult to perform manually.	これは、ソフトウェア・テスト技法で、実際の成果を期待される成果と比較するためのものです。テスト・スクリプトを書いたり、任意の自動化テストツールを使用することで実現できます。テスト自動化は、手動で行うことが困難な反復タスクやその他のテスト・タスクを自動化するために使用されます。
Automation	**自動化**	This is the use of technology to perform a step or series of steps correctly and consistently with limited or no human intervention; also the standardization and streamlining of manual tasks such as defining the rules of part of a process to allow decisions to be made 'automatically'.	これは、テクノロジーを使用して、人間の介入が制限されているか、まったくない状態で、1 つまたは一連のステップを正しく一貫して実行することです。 また、プロセスの一部のルールを定義して「自動的に」意思決定できるようにするなど、手動タスクの標準化と合理化も行います。
Availability	**可用性**	Data and information systems are available when required. Hardware maintenance, software patching/upgrading, and network optimization ensures availability.	データおよび情報システムは必要なときに利用可能です。ハードウェアのメンテナンス、ソフトウェアのパッチ適用/アップグレード、ネットワークの最適化により可用性を確保します。
Blue/green deployments	**ブルーグリーン・デプロイメント**	This is two identical production environments that run simultaneously (one 'blue' and one 'green'), with only one of the environments being live and serving all production traffic, whereas the other is used for deployment of the new versions.	これは、2 つの同一の本番環境（1 つは「ブルー」、1 つは「グリーン」）を同時に稼働させ、1 つの環境だけが稼働中ですべての本番トラフィックに対応しており、もう 1 つは新しいバージョンの展開に使用されています。
Canary release	**カナリア・リリース**	This is a dark launch in which a few test users are invited to test the new functionality. The new feature is initially released to a small, targeted group of users, gradually increasing to all users.	これは、少数のテストユーザーを招待し、新機能をテストしてもらうダーク・ローンチです。新機能は、最初は少数のターゲットとなるユーザーグループにリリースされ、徐々に全ユーザーに拡大していきます。
Capacity planning	**キャパシティ計画立案**	This is the activity of creating a plan that manages resources to meet a demand for services.	これは、サービスの需要に応えるために、リソースを管理する計画を作成するアクティビティです。
Confidentiality	**機密性**	This ensures that data or an information system is accessed by only an authorized person.	これにより、データや情報システムへのアクセスは、認可された者のみとなります。

Term	用語	Definition	定義
Container	**コンテナ**	This is a container that consists of an entire runtime environment: an application, plus all its dependencies, libraries and other binaries, and configuration files that needed to run it, and bundled into one package. By containerizing the application platform and its dependencies, differences in OS distributions and underlying infrastructure are abstracted away.	アプリケーションとそれに必要な全ての依存関係、ライブラリ、その他のバイナリ、構成ファイルなどからなるランタイム環境全体を、1 つのパッケージにバンドルして構成したコンテナのことです。アプリケーション・プラットフォームとその依存関係をコンテナ化することで、OS のディストリビューションや基盤となるインフラの違いを抽象化することができます。
Container cluster	**コンテナ・クラスタ**	This is a set of connected node machines for running containerized applications.	コンテナ化されたアプリケーションを実行するための、接続されたノードマシンの集合体です。
Continuous integration (CI)	**継続的インテグレーション(CI)**	This is a development practice where developers integrate code into a shared repository frequently, preferably several times a day.	開発者がコードを頻繁に、できれば 1 日に数回、共有リポジトリに統合する開発手法です。
Continuous delivery (CD)	**継続的デリバリ（CD）**	This is a set of practices designed to ensure that code is always able to be deployed rapidly and safely throughout its life cycle to production and is achieved by pushing the executables into a production-like environment and conducting automated testing to detect problems.	コードのライフサイクルを通じて、常に迅速かつ安全に本番環境に展開（デプロイメント）できるようにするためのプラクティスで、本番環境に近い環境に実行可能なファイルをプッシュし、自動テストを実施して問題を検出することによって実現されます。
Continuous monitoring	**継続的モニタリング**	This refers to the process and technology required to incorporate monitoring across each phase of the DevOps and IT operations life cycles.	DevOps と IT 運用ライフサイクルの各フェーズに渡ってモニタリングを組み込むために必要なプロセスとテクノロジーを指します。
Conway's Law	**コンウェイの法則**	Organizations that design systems are constrained to produce designs that are copies of the communication structures of these organizations. – Mel Conway	システムをデザインする組織は、その組織のコミュニケーション構造のコピーですデザインを生み出すという制約を受けている。- メル・コンウェイ
Critical to quality (CTQ)	**クリティカル・ツウ・クオリテイ(CTQ)**	This is an attribute of a part, assembly, subassembly, product, or process that is literally critical to quality or, more precisely, has a direct and significant impact on its actual or perceived quality.	部品、アセンブリ、サブアセンブリ、プロダクト、プロセスなどの属性で、文字通り品質にとって重要なもの、より正確には、その実際の品質または知覚される品質に直接および大きな影響を与えるものです。

Term	用語	Definition	定義
Critical success factor (CSF)	**重要成功要因（CSF）**	This is a success factor that describes a condition or characteristic that must be achieved for something to be considered successful.	何かが成功したとみなされるために達成されなければならない条件や特性を表す成功要因のことです。
Customer reliability engineer (CRE)	**顧客リライアビリティ・エンジニア（CRE）**	These are engineers who integrate with a customer's operations teams to share the reliability responsibilities for critical cloud applications.	顧客の運用チームと統合し、重要なクラウド・アプリケーションの信頼性責任を分担するエンジニアです。
Dark launch	**ダーク・ローンチ**	This means a new functionality is launched, but the link to the new functionality is not visible before the functionality is mature enough.	新機能がローンチされたが、その機能が十分に成熟する前は、新機能へのリンクが表示されないことを意味します。
Deployment management	**展開管理**	The purpose of deployment management is to move new or changed software or any other component to live environments.	展開管理の目的は、新規または変更されたソフトウェアやその他のコンポーネントを稼働中の環境に移動させることです。
Deployment pipeline	**デプロイメント・パイプライン**	This is a process model that gives you a starting point from which to provide transparency and control as work moves through the various sets of tests and deployments toward release.	このプロセスモデルは、リリースに向けてさまざまなテストや展開（デプロイメント）を行う際に、透明性を確保しコントロールするための出発点となるものです。
DevOps	**DevOps**	This is a movement represented by the convergence of existing IT best practices from ITIL®, Lean, and Agile into a development and operations approach that supports automation and continuous delivery. It also encourages a culture of collaboration and learning to help IT deliver business value better, faster, and cheaper than ever before.	これは、ITIL® 、リーン、アジャイルなどの既存の IT ベスト・プラクティスを、自動化と継続的デリバリーをサポートする開発・運用アプローチに収束させることで表現されるムーブメントです。また、コラボレーションと学習のカルチャを促進し、IT がこれまで以上に優れた、より速く、より安価に、ビジネス価値を提供できるよう支援します。
Durability	**耐久性 Durability**	This means the persistence of data.	これはデータの永続性を意味します。
Event	**イベント**	This is any change of state that has significance for the management of a configuration item (CI) or IT service.	これは、構成アイテム（CI）または IT サービスのマネジメントにとって重要な、あらゆる状態の変更を指す。

Term	用語	Definition	定義
Feature flags	**フィーチャー・フラグ**	This is a modern deployment technique that helps increase agility for cloud-native applications. They enable you to deploy new features into a production environment but restrict their availability. With the flick of a switch (toggle), you can activate a new feature for specific users without restarting the app or deploying new code.	これは、クラウドネイティブなアプリケーションのアジリティを高めるのに役立つ最新の展開（デプロイメント）技法です。本番環境に新しいフィーチャーを展開（デプロイメント）しますが、そのフィーチャーの利用を制限します。スイッチ（トグル）をスライドするだけで、アプリを再起動したり新しいコードを展開したりすることなく、特定のユーザーに対して新しいフィーチャーを有効にすることができます。
(The) First Way	**第一の道**	Flow/Systems thinking: The First Way emphasizes the performance of the entire system, as opposed to the performance of a specific silo of work or department.	フロー/システム思考。第一の道では、特定の作業や部門のパフォーマンスとは対照的に、システム全体のパフォーマンスを重視します。
Gamification	**ゲーミフィケーション**	This means adding game mechanics into nongame environments with a goal of engaging consumers, employees, and partners to inspire, collaborate, share, and interact.	これは、消費者、従業員、パートナーをエンゲージし、インスピレーションを与え、コラボレーションし、共有し、交流することを目的として、非ゲーム環境にゲームのメカニズムを追加することを意味します。
Gated commit	**ゲーテッド・コミット**	This is also called a pretested commit; it is an integration pattern in which a commit is not approved until a set of tests are run against the code being committed.	これはプリテスト・コミットとも呼ばれ、コミットされるコードに対して一連のテストが実行されるまで、コミットを承認しないという統合パターンです。
Horizontal scaling	**水平スケーリング**	Cloud horizontal scaling refers to provisioning additional servers to meet your needs, often splitting workloads between servers to limit the number of requests any individual server is getting. In a cloud-based environment, this would mean adding additional instances instead of moving to a larger instance size, often called scaling out or in.	クラウドの水平スケーリングとは、ニーズに合わせて追加のサーバーをプロビジョニングすることで、多くの場合、サーバー間でワークロードを分割して、個々のサーバーが受けるリクエストの数を制限します。クラウドベースの環境では、より大きなインスタンス・サイズに移行するのではなく、インスタンスを追加することを意味し、スケール・アウトやスケール・インと呼ばれることもあります。

Term	用語	Definition	定義
Idempotent	**べき等**	This is an operation, action, or request that can be applied multiple times without changing the result; i.e., the state of the system beyond the initial application. Making multiple identical requests has the same effect as making a single request.	これは、結果を変更することなく複数回適用できる操作、アクション、または要求です。つまり、システムの状態は最初の適用に及びません。同じリクエストを複数回行っても、1 回のリクエストと同じ効果があります。
Immutable infrastructure	**イミュータブル・インフラストラクチャ**	A cloud-native model in which no updates, security patches, or configuration changes happen 'in place' on production systems. If any change is needed, a new version of the architecture is built and deployed into production.	アップデート、セキュリティパッチ、コンフィグレーション/ 構成の変更などが、本番システム上で「その場」で行われないクラウド・ネイティブ・モデルです。変更が必要な場合は、新しいバージョンのアーキテクチャを構築し、本番環境に展開（デプロイメント）します。
Incident commander	**インシデント・コマンダー**	This is the incident manager who is accountable for a significant or major incident(s).	重大なインシデント（複数可）の説明責任者であるインシデント・マネージャです。
Infrastructure as a service (IaaS)	**IaaS（Infrastructure as a Service）**	This is a cloud deployment model in which the client manages their own application, meta data, runtime, middleware, and operating system, but the virtualization, servers, storage and networking are all managed in the cloud.	これは、アプリケーション、メタデータ、ランタイム、ミドルウェア、オペレーティングシステムはクライアント側で管理しますが、仮想化、サーバー、ストレージ、ネットワークはすべてクラウド側で管理するクラウド展開（デプロイメント）モデルです。
Infrastructure as code (IaC)	**IaC（Infrastructure as code）**	Also called programmable infrastructure, it uses a high-level descriptive coding language to automate the provisioning of IT infrastructure. This automation eliminates the need for developers to manually provision and manage servers, operating systems, database connections, storage, and other infrastructure elements every time they want to develop, test, or deploy a software application.	プログラマブル・インフラストラクチャとも呼ばれ、高水準の記述型コーディング言語を使用して、IT インフラストラクチャのプロビジョニングを自動化します。この自動化により、開発者はソフトウェア・アプリケーションの開発、テスト、展開のたびに、サーバー、オペレーティングシステム、データベース接続、ストレージ、およびその他のインフラストラクチャ要素を手動でプロビジョニングおよび管理する必要がなくなります。
Integrity	**完全性**	Integrity ensures that the data or information system can be trusted; ensures that it is edited by only authorized persons.	完全性は、データや情報システムが信頼できることを保証するものであり、認可された者のみが編集できることを保証します。

Term	用語	Definition	定義
Interrupt	**割込み**	This is unplanned work (in the context of SRE); generally attributed to operational work that is required to keep systems running on a day-to-day basis.	これは、計画外の作業（SRE の文脈の中で）であり、一般的には、原因はシステムを日々稼働させるために必要な運用作業にあります。
Kaizen	**カイゼン/改善**	This is the Japanese word for continuous improvement by using small incremental changes; translates as “change for the better”.	日本語では、「より良い方向への変更」と訳される、小さな漸進的な変更による継続的な改善のことです。
Key performance indicator (KPI)	**重要業績評価指標（KPI）**	The critical (key) indicators of progress toward an intended result; usually the result of one or more metrics viewed over time – creating a trend.	意図した結果に対する進捗の重大な（重要な）指標。通常、時間の経過とともに 1 つ以上のメトリックを表示した結果であり、トレンドを生み出す。
Landing zone	**ランディング・ゾーン（着陸地帯）**	This is the underlying core configuration of any cloud adoption environment that provides a preconfigured environment to host workloads in private, hybrid, or public clouds.	これは、あらゆるクラウド採用の環境の基礎となるコア構成で、プライベート、ハイブリッド、またはパブリッククラウドでワークロードをホストするための構成済み環境を提供するものです。
Launches	**ローンチ**	This is any new code that introduces an externally visible change to an application.	これは、アプリケーションに外部から見える変更を導入する新しいコードです。
Linear scaling	**リニア・スケーリング**	A linearly scalable application is an application that can scale just by adding more machines and/or CPUs, without changing the application code.	リニア・スケーラブルなアプリケーションとは、アプリケーションのコードを変更することなく、マシンや CPU を追加するだけでスケールできるアプリケーションのことです。
Mean time to repair (MTTR)	**平均修理時間（MTTR）**	This is the average time to repair a system, including repair and testing time.	修理やテスト時間など、システムの修理にかかる平均的な時間です。
Metric	**メトリック**	This is a snapshot, at a given point in time, of the performance of an application or system.	これは、アプリケーションやシステムのパフォーマンスについて、ある時点のスナップショットです。
Microservices architecture (MSA)	**マイクロ・サービス・アーキテクチャ（MSA）**	This is a microservices architecture that involves smaller applications deployed independently as loosely coupled services that are tied together through application integration.	これは、小さなアプリケーションが疎結合のサービスとして独立して展開（デプロイメント）され、アプリケーションの統合によって結びつけられるマイクロ・サービス・アーキテクチャです。

Term	用語	Definition	定義
Monitoring and event management	**モニタリングとイベント管理**	The purpose of monitoring and event management is to systematically observe services and service components, as well as record and report selected changes of state identified as events.	モニタリングとイベント管理の目的は、サービスやサービスコンポーネントを体系的に観察し、イベントとして特定された状態の変更を記録し報告することです。
Monolithic application	**モノリシック・アプリケーション**	Monolithic applications are designed to handle multiple related tasks. They're typically complex applications that encompass several tightly coupled functions.	モノリシック・アプリケーションは、複数の関連するタスクを処理するために設計されています。一般的には、いくつかの機能が密結合された複雑なアプリケーションです。
Network engineering	**ネットワーク・エンジニアリング**	This is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of networks.	これは、ネットワークの開発・運用・保守に、体系的で規律正しく、定量化できるアプローチを適用することです。
Observability	**オブザバビリティ（可観測性）**	Observability goes beyond mere monitoring (even of very complicated infrastructures) and is instead about building visibility into every layer of your business. Increased visibility gives everyone invested in the business greater insight into issues and the user experience and creates more time for more strategic initiatives, instead of firefighting issues. It's also critical to the overall success of site reliability engineering (SRE) or DevOps organization models.	可観測性とは、単なるモニタリング（たとえ非常に複雑なインフラであっても）を超えて、ビジネスのあらゆるレイヤーに可視性を構築することです。可視性が向上すると、ビジネスに関わるすべての人が問題やユーザー・エクスペリエンスについてより深い洞察が得られ、課題を解決するための時間ではなく、より戦略的な取り組みに時間を割くことができるようになります。また、Site Reliability Engineering（SRE）や DevOps の組織モデルを全体的に成功させるためにも重要です。
On call	**オンコール**	This means being available during a set period of time and being ready to respond to production incidents during that time with the appropriate urgency.	これは、一定時間内に可用性を確保し、その間に発生した本番インシデントに適切な緊急性を持って対応できるようにすることを意味します。
OpenMetrics	**オープン・メトリクス**	This specifies the *de facto* standard for transmitting cloud-native metrics at scale, with support for both text representation and protocol buffers. It supports both pull- and push-based data collection.	これは、クラウド・ネイティブのメトリックを大規模に伝送するためのデファクト・スタンダードを規定したもので、テキスト表現とプロトコルバッファの両方をサポートしています。プル型とプッシュ型の両方のデータ収集に対応しています。

Term	用語	Definition	定義
OpenTelemetry	**オープン・テレメトリー**	This is an observability framework for cloud-native software. It is a Cloud Native Computing Foundation (CNCF) sandbox project with the ultimate goal of providing a unified set of vendor-agnostic libraries/APIs for collecting and sending data to compatible back ends.	これは、クラウド・ネイティブ・ソフトウェアのための可観測性フレームワークです。Cloud Native Computing Foundation (CNCF) のサンドボックスプロジェクトで、データを収集して互換性のあるバックエンドに送るための、ベンダーに依存しないライブラリ/API の統一セットを提供することを最終目的としています。
Platform as a service (PaaS)	**Platform as a Service（PaaS）**	This is a cloud deployment model in which the client only manages the application and meta data with everything else managed by the vendor in the cloud.	これは、お客様がアプリケーションとメタデータのみを管理し、それ以外はすべてクラウド上のベンダーが管理するクラウド展開（デプロイメント）モデルです。
Platform team	**プラットフォームチーム**	This is the team responsible for building and running a compelling platform that accelerates and simplifies software delivery for stream-aligned teams.	このチームは、ストリーム・アラインド・チームのソフトウェア・デリバリーを加速および簡略化する、魅力的なプラットフォームの構築と運営に責任を負っています。
Policy as code	**Policy as code**	This is the idea of writing code in a high-level language to manage and automate policies.	これは、ポリシーの管理と自動化のために、高級言語でコードを書くというものです。
Predictive development model	**プレディクティブ・デベロップメント・モデル（予知可能な条件下での開発モデル）**	This model is also known as traditional, sequential, linear, or – most famously – as waterfall approaches that are less flexible, because they take an entire deliverable through large phases of requirements, design, development, integration, testing, and deployment. Progress flows steadily downward through the linear phases.	このモデルは、伝統的、順次的、線形的、あるいは最も有名なウォーターフォール・アプローチとしても知られており、要件、設計、開発、統合、テスト、展開（デプロイメント）という大きな段階を経て成果物全体を完成させるため、柔軟性に欠けるものです。進捗は、直線的なフェーズを着実に下へ下へと流れていきます。

Term	用語	Definition	定義
Process	**プロセス**	This is a set of interrelated or interacting activities that transform inputs into outputs. Processes define the sequence of activities and their dependencies. They describe what is done to accomplish an objective, and well-defined processes can improve productivity within and across organizations. They are usually detailed in procedures that outline who is involved in the process and work instructions, which explain how they are carried out.	これは、インプットをアウトプットに変換する、相互に関連する、または相互作用するアクティビティの集合です。プロセスは、アクティビティの順序とその依存関係を定義します。プロセスは、目的を達成するために何が行われるかを記述するものであり、明確に定義されたプロセスは、組織内外の生産性を向上させることができます。通常、実行方法を説明したプロセスと作業指示に関与させる人物の概要を示す手順で詳細に述べられます。
Product engineering	**プロダクト・エンジニアリング**	This is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of IT products.	IT プロダクトの開発・運用・保守に、体系的・規律的・定量的なアプローチを適用することです。
Product team	**プロダクトチーム**	See the definition for stream-aligned team.	ストリーム・アラインド・チームの定義を参照してください。
Production readiness review	**プロダクト・レディネス・レビュー（PRR：Production readiness review）**	This is an engineering discipline and activity that assesses a product or service at various stages to determine if it is ready for its intended use.	プロダクトやサービスを様々なステージでアセスメントし、意図した用途に使えるかどうかを判断する工学的な学問分野であり、アクティビティでもあります。
Provisioning	**プロビジョニング**	These are activities performed by an organization to provide services, such as the management of the provider's resources that are configured to deliver a service.	サービスを提供するために組織が行う活動で、サービスを提供するために構成されたプロバイダのリソースの管理など。
Psychological safety	**心理的安全性**	This refers to making it safe to talk about problems, because solving problems requires prevention, which requires honesty, and honesty requires the absence of fear.	問題解決には予防が必要で、そのためには正直さが必要で、正直さには恐怖心がないことが必要ですため、問題について安全に話ができるようにすることを意味します。
Public key infrastructure (PKI)	**PKI（公開鍵暗号基盤）**	This is a set of roles, policies, hardware, software, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates and manage public-key encryption.	デジタル証明書の作成、管理、配布、使用、保管、失効、公開鍵暗号の管理に必要な役割、ポリシー、ハードウェア、ソフトウェア、手順のセットです。

Term	用語	Definition	定義
Pull production system	**プル生産システム**	Production that is based on a customer order is “pulling” the product through the process.	顧客からの受注に基づく生産は、工程を通じてプロダクトを「プルする」ことになります。
Push production system	**プッシュ生産システム**	Production that takes place before a customer order is “pushing” the product through the process.	顧客からのオーダーを受ける前に行われる生産は、工程を通じてプロダクトを「プッシュする」ことになります。
Release engineering	**リリース・エンジニアリング**	This is a discipline within software engineering (SWE) that’s focused on defining and managing all the steps required to release software, including the building and delivering of software.	ソフトウェア・エンジニアリング（SWE）の中の一分野であり、ソフトウェアの構築やデリバリーなど、ソフトウェアのリリースに必要なすべてのステップを定義し管理することに重点を置いています。
Release management	**リリース管理**	The purpose of release management is to make new and changed services and features available for use by the customer.	リリース管理の目的は、新しいサービスや変更されたフィーチャーを、顧客が使用できるようにすることです。
Release train	**リリース・トレイン**	An agile release train (ART) is a preplanned release cycle that has been stringently planned into phases of which individual projects and/or business-as-usual (BAU) release teams must align to in order to deliver their release.	アジャイル・リリース・トレイン（ART）とは、事前に計画されたリリースサイクルのことで、個々のプロジェクトや通常業務（BAU）のリリースチームが、そのリリースを実現するために合わせなければならないフェーズに厳格に計画されているものです。
Reliability	**リライアビリティ（信頼性）**	This is the probability that the system will meet certain performance standards and yield a correct output for a specific time.	これは、システムが特定のパフォーマンス基準を満たし、特定の時間に正しいアウトプットを生成する可能性のことです。
Risk management	**リスク管理**	This is the practice of ensuring that an organization understands and effectively handles risks.	これは、組織がリスクを把握し、効果的に対処するためのプラクティスです。
Root cause analysis	**根本原因分析**	This is a process of human deduction paired with reporting tools to uncover causes of problems.	これは、人による推理とレポート報告書との組み合わせで、問題の原因を明らかにするプロセスです。
Scrum	**スクラム（Scrum）**	This is a lightweight framework that helps people, teams, and organizations generate value through adaptive solutions for complex problems.	これは、人、チーム、組織が複雑な問題に対してアダプション（適応）的な解決策を講じることでバリューを生み出すことを支援する軽量なフレームワークです。

Term	用語	Definition	定義
(The) Second Way	第二の道	Amplify feedback loops: The Second Way is about creating the right-to-left feedback loops. The goal of almost any process improvement initiative is to shorten and amplify feedback loops so necessary corrections can be continually made.	フィードバック・ループを増幅させる。第二の道は、右から左へのフィードバック・ループを作成することです。ほぼすべてのプロセス改善イニシアチブの目標は、フィードバック・ループを短縮および増幅し、必要な修正を継続的に行えるようにすることです。
Service level agreement (SLA)	サービスレベルアグリーメント（SLA）	This is an explicit or implicit contract with your users that includes the consequences of meeting (or missing) the service level objectives (SLOs) they contain.	これは、ユーザーとの明示的または暗黙的な契約であり、そこに含まれるサービスレベル目標（SLO）を満たした場合（または満たせなかった場合）の結果を含んでいます。
Service level indicator (SLI)	サービスレベル指標（SLI）	This is a carefully defined quantitative measure of some aspect of the level of service that is provided.	これは、提供されるサービス・プロバイダのレベルのある側面について、慎重に定義された定量的な測定です。
Service level objective (SLO)	サービスレベル目標（SLO）	This is a target value or range of values for a service level that is measured by a service level indicator (SLI).	サービスレベル指標（SLI）で測定されるサービスレベルの目標値または数値の範囲です。
Single source of truth (SSOT)	シングル・ソース・オブ・トゥルース（SSOT）	This is the practice of aggregating the data from many systems within an organization to a single location.	これは、組織内の多くのシステムからデータを一カ所に集約するプラクティスです。
Software as a service (SaaS)	SaaS（Software as a service）	This is a cloud deployment model in which the vendor manages everything.	ベンダーがすべてを管理するクラウド展開（デプロイメント）モデルです。
Software engineering	ソフトウェア・エンジニアリング	This is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software.	これは、ソフトウェアの開発・運用・保守に、体系的で規律正しく、定量化可能なアプローチを適用することです。
Service configuration management	サービス構成管理	The purpose of service configuration management is to ensure that accurate and reliable information about the current configuration of services, and the components that support them, is available when and where it is needed.	サービス構成管理の目的は、サービスの現在の構成とそれをサポートするコンポーネントに関する正確で信頼性の高い情報を、必要なときに必要な場所で利用できるようにすることです。

Term	用語	Definition	定義
Shift-left testing	**シフトレフト・テスト**	The principle of 'shift left' supports the testing team to collaborate with all the stakeholders early in the software development phase. Hence, they can clearly understand the requirements and design the test cases to help the software 'fail fast' and enable the team to fix all the failures at the earliest time possible.	「シフトレフト」の原則は、テストチームがソフトウェア開発フェーズの早い段階で、すべてのステークホルダーとコラボレーションすることをサポートします。そのため、要件を明確に理解し、テストケースを設計することで、ソフトウェアの「フェイル・ファスト」を支援し、チームができるだけ早い段階ですべての障害を修正できるようにすることができます。
Site reliability engineering (SRE)	**サイト・リライアビリティ・エンジニアリング（SRE）**	SRE, pioneered by Google, is radically different from IT operations of the past and is a specific approach to IT operations for large-scale, cloud-native software systems. The SRE model sets up a healthy and productive interaction between the development and SRE teams by using service-level objectives (SLOs) and error budgets to balance the speed of new features with whatever work is needed to make the software reliable.	SRE は、Google が開拓したもので、これまでの IT 運用とは根本的に異なり、大規模なクラウド・ネイティブのソフトウェア・システムに対する IT 運用の特有のアプローチです。SRE モデルでは、サービスレベル目標（SLO）とエラーバジェットを用いて、新しいフィーチャーの開発スピードとソフトウェアの信頼性を高めるために必要なあらゆる作業のバランスを取ることで、開発チームと SRE チームの間に健全で生産的なやり取りを設定します。
Smoke testing	**スモーク・テスト**	This is surface-level testing to certify that the build provided by the development team to the QA team is ready for further testing. This testing is normally used in integration testing, system testing, and acceptance level testing.	開発チームから QA チームに提供されたビルドが、さらなるテストを行える状態ですことを証明するための表面レベルのテストです。このテストは通常、統合テスト、システム・テスト、受入レベル・テストで使用されます。
Source code management (SCM)	**ソースコード管理（SCM）**	This tracks changes to a source code repository. SCM also maintains a history of changes. This is used to resolve conflicts when merging updates from multiple developers.	これは、ソースコード・リポジトリへの変更を追跡するものです。また、SCM は変更履歴も保守します。これは、複数の開発者からの更新をマージする際に、コンフリクトを解決するために使用されます。

Term	用語	Definition	定義
Stream-aligned team	**ストリーム・アラインド・チーム**	These are sometimes called full-stack, cross-functional, or product teams. Stream-aligned teams are aligned to the main flow of business change and have the cross-functional skills and ability to deliver significant increments of value without waiting on other teams. These teams might be oriented around a single product, a user journey, a set of features, etc.	これらは、フルスタック・チーム、クロスファンクション・チーム、プロダクト成 果物チームと呼ばれることもあります。ストリーム・アラインド・チームは、ビジネス変更の主要なフローに沿ったもので、他のチームを待たせることなく、大きなバリューのインクリメントを提供できるクロス・ファンクショナルなスキルと能力を備えています。このようなチームは、1 つのプロダクト、ユーザー・ジャーニー、一連のフィーチャーなどを中心に活動することがあります。
Swarming	**スウォーミング**	This involves many different stakeholders working together, initially, until it becomes clear which of them is best placed to continue and which can move on to other tasks.	最初はさまざまなステークホルダーが協力し合い、その中から継続した方が良い人、他の仕事に移った方が良い人が明確になるまで、作業を続けます。
System resilience	**システム・レジリエンス**	The ability of a system (including the people dimension) to withstand a major disruption within acceptable degradation parameters and to recover within an acceptable time.	システム（人の側面を含む）が許容される劣化パラメータの範囲内で大きなディスタラプティブに耐え、許容時間内に回復する能力です。
Technical debt	**技術的負債**	This is the accumulation of complicated workarounds and rework that occurs when easy solutions are consistently implemented, instead of the best solutions.	これは、ベストソリューションではなく、安易なソリューションを一貫して実装することで発生する、複雑なワークアラウンドや手戻りの積み重ねです。
Test-driven development (TDD)	**テスト駆動開発(TDD)**	This is a software development process that relies on software requirements being converted to test cases before the software is fully developed, and then tracks all the software development by repeatedly testing the software against all test cases.	これは、ソフトウェアが完全に開発される前に、ソフトウェアの要件がテストケースに変換されることに依存し、すべてのテストケースに対してソフトウェアのテストを繰り返すことによって、すべてのソフトウェア開発を追跡するソフトウェア開発プロセスです。

Term	用語	Definition	定義
Theory of constraints	**制約理論（ToC）**	“The Theory of Constraints is a methodology for identifying the most important limiting factor (i.e., constraint) that stands in the way of achieving a goal and then systematically improving that constraint until it is no longer the limiting factor. In manufacturing, the constraint is often referred to as a bottleneck.”	制約理論とは、目標達成の妨げとなる最も重要な制限要因（＝制約）を特定し、その制約が制限要因でなくなるまで体系的に改善する方法論です。製造業では、制約条件は「ボトルネック」と呼ばれることが多いです。
(The) Third Way	**第三の道**	A culture of continual experimentation and learning: The Third Way is about creating a culture that fosters two things: continual experimentation, taking risks, and learning from failure; and understanding that repetition and practice is the prerequisite to mastery.	実験と学習を継続するカルチャ：第三の道は、継続的な実験、リスクを取ること、失敗から学習すること、そして、反復と実践が習得の前提ですことを理解するという 2 つのことを育むカルチャを創造することです。
Toil	**トイル（Toil）**	This is the kind of work tied to running a production service that tends to be manual, repetitive, automatable, tactical, devoid of enduring value, and that scales linearly as a service grows.	これは、プロダクションサービスの運営に関連する作業で、手作業で、反復的で、自動化可能で、戦術的な永続的な価値を持たず、サービスの成長に伴ってリニアに規模が拡大する傾向があります。
Utilization	**ユーティリゼイション（使用率）**	This is a function of how a given service works and how it is provisioned.	これは、あるサービスがどのように機能し、どのように供給されるかの機能です。
Value stream	**バリュー・ストリーム**	This is a series of steps an organization undertakes to create and deliver products and services to consumers. A value stream is a combination of the organization’s value chain activities.	これは、組織がプロダクトやサービスを生み出し、消費者に提供するために行う一連のステップです。バリュー・ストリームは、組織のバリュー・チェーン活動の組み合わせです。

Term	用語	Definition	定義
Version control	**バージョン・コントロール**	Version control allows you to manage changes to files over time and store these modifications in a database. Also called source control, you can use version control to version code, binary files, and digital assets. This includes version control software, version control systems, or version control tools. Version control is a component of software configuration management. It's sometimes referred to as version control system (VCS) programming.	バージョン・コントロールは、時間の経過に伴うファイルの変更点を管理し、これらの変更点をデータベースに保管することができます。ソース・コントロールとも呼ばれ、バージョン・コントロールを使って、コード、バイナリ・ファイル、およびデジタル資産をバージョン管理することができます。これには、バージョン・コントロール・ソフトウェア、バージョン・コントロール・システム、またはバージョン・コントロール・ツールが含まれます。バージョン・コントロールは、ソフトウェア構成管理の構成要素です。バージョン・コントロール・システム（VCS）プログラミングと呼ばれることもあります。
Vertical scaling	**垂直スケーリング**	Cloud vertical scaling refers to adding more CPU, memory, or I/O resources to an existing server or replacing one server with a more powerful server, often called scaling up or down.	クラウドの垂直スケーリングとは、既存のサーバーに CPU やメモリー、I/O などのリソースを追加したり、あるサーバーをより高性能なサーバーに置き換えることを指し、しばしばスケール・アップやスケール・ダウンと呼ばれます。
Voice of the customer (VoC)	**顧客の声（VOC）**	The in-depth process of capturing a customer's expectations, preferences, and aversions with the objective to create products or services that meet the customer's needs and preferences.	顧客のニーズや好みに合ったプロダクトやサービスを作るために、顧客の期待や好み、嫌悪感などを深く把握するプロセス。
Waterfall development	**ウォーターフォール開発**	See the predictive development model.	プレディクティブ・デベロップメント・モデル（予知可能な条件下での開発）を参照。
Work in progress (WIP)	**ワーク・イン・プログレス(WIP)**	This is the number of task items that a team is currently working on. It frames the capacity of the team's workflow at any moment.	チームが現在作業しているタスクの数です。チームのワークフローの現時点でのキャパシティを枠で表したものです。

PeopleCert

DevOps.

www.peoplecert.org