DevOps Institute

SRE Foundation℠
試験勉強ガイド

DevOps Institute's SKIL Framework

DevOps Institute is dedicated to advancing the human elements of DevOps success through our human-centered SKIL framework of Skills, Knowledge, Ideas and Learning.

We develop, accredit and orchestrate SKIL through certifications, research, learning opportunities, events and community connections.

Visit us at www.devopsinstitute.com to learn more.

Join Us!

Become a member and join the fastest growing global community of DevOps practitioners and professionals.

The DevOps Institute continuous learning community is your go-to hub for all things DevOps, so get ready to learn, engage, and inspire.

Visit www.devopsinstitute.com/membership/ to join today.

You belong.

DevOps Institute

DevOps Institute is dedicated to advancing the human elements of DevOps success. We fulfill our mission through our SKIL framework of Skills, Knowledge, Ideas and Learning.

Certification is one means of showcasing your skills. While we strongly support formal training as the best learning experience and method for certification preparation, DevOps Institute also recognizes that humans learn in different ways from different resources and experiences. As the defacto certification body for DevOps, DevOps Institute has now removed the barrier to certification by removing formal training prerequisites and opening our testing program to anyone who believes that they have the topical knowledge and experience to pass one or more of our certification exams.

This examination study guide will help test-takers prepare by defining the scope of the exam and includes the following:

- Course Description
- Examination Requirements
- DevOps Glossary of Terms
- Value Added Resources
- Sample Exam(s) with Answer Key

These assets provide a guideline for the topics, concepts, vocabulary and definitions that the exam candidate is expected to know and understand in order to pass the exam. The knowledge itself will need to be gained on its own or through training by one of our Global Education Partners.

Test-takers who successfully pass the exam will also receive a certificate and digital badge from DevOps Institute, acknowledging their achievement, that can be shared with their professional online networks.

If you have any questions, please contact our DevOps Institute Customer Service team at CustomerService@DevOpsInstitute.com.

Site Reliability Engineering (SRE) Foundation℠

講義時間 - 16時間

サービスの信頼性を向上させるために、自動化、作業方法、組織の再編成などを組み合わせたさまざまな実践方法を紹介します。
大規模なサービスの可用性を重視する方に適しています。

オーバービュー

SRE (Site Reliability Engineering) Foundation℠コースは、組織が重要なサービスを確実かつ経済的に拡張するための原則と実践を紹介します。サイト・リライアビリティの側面を導入するには、組織の再編成、エンジニアリングおよび自動化への新たな取り組み、そしてさまざまな新しい作業パラダイムの採用が必要です。

本コースでは、SREの進化とその将来的な方向性に焦点を当て、信頼性と安定性に関わる組織全体の人々を巻き込むための実践、手法、ツールを、実際のシナリオやケーススストーリーを用いて習得します。このコースを修了すると、参加者は、サービスレベル目標(SLO)の理解、設定、追跡など、会社に戻ってから活用できる具体的な成果を得ることができます。

このコースは、SREの主要な情報源を活用し、SRE分野のオピニオンリーダーと協力し、SREを採用している組織と協力して実際のベストプラクティスを抽出することで開発され、SREの採用を開始するために必要な主要な原則と実践を学ぶことができるように設計されています。

このコースは、SRE Foundation認定試験を成功させるための学習者を位置づけています。

コース目標

SRE Foundationコースの学習目標には、以下の実践的な理解が含まれています。

- SREの歴史とグーグルでの登場について
- SREとDevOpsやその他の一般的なフレームワークとの相互関係
- SREの基本理念
- サービスレベル目標(SLO)とそのユーザーフォーカス
- サービスレベルインジケータ(SLI)と現代のモニタリング事情
- エラー予算とそれに伴うエラー予算ポリシー
- 労働と組織の生産性への影響

- Toilをなくすためのいくつかの実践的なステップ
- サービスの健全性を示すものとしての観測性
- SREツール、自動化技術、セキュリティの重要性
- アンチフラジリティ、失敗と失敗のテストに対する私たちのアプローチ
- SRE導入がもたらす組織的なインパクト

対象者

SRE Foundationコースの対象者は、以下のようなプロフェッショナルです。

- 信頼性向上への取り組みを始めた方、または主導している方
- 現代のITリーダーシップや組織変革のアプローチに興味のある方
- ビジネスマネージャー
- ビジネス・ステークホルダー
- チェンジエージェント
- コンサルタント
- DevOpsプラクティショナー
- ITディレクター
- ITマネージャー
- ITチームリーダー
- プロダクトオーナー
- スクラムマスター
- ソフトウェア・エンジニア
- サイト・リライアビリティ・エンジニア
- システムインテグレーター
- ツール提供者

学習者用教材

- 16時間のインストラクターによるトレーニングとエクササイズのファシリテーション
- 学習者マニュアル(授業後の参考資料として最適)を含む。
    - コースのスライドウェア
    - 付加価値の高いリソース
    - 用語集
- コンセプトを応用した演習やディスカッションへの参加
- 事例紹介
- 追加の情報源やコミュニティへのアクセス

認定試験

40問の多肢選択式問題からなる60分の試験に合格(65%)すると、SRE(Site Reliability Engineering)Foundationの認定証が発行されます。
この認定資格は、DevOps Instituteによって管理・維持されています。

コースの概要

- コース紹介
    - コースゴール
    - コースアジェンダ
- モジュール1：SREの原則と実践
    - Site Reliability Engineeringとは？
    - SREとDevOps：その違いとは？
    - SREの原則と実践
- モジュール2：サービスレベル目標とエラー予算
    - サービスレベル目標(SLO)
    - エラー予算
    - エラー予算の方針
- モジュール3:Toilの削減
    - Toilとは？
    - Toilはなぜ悪いのか？
    - Toliをどうするか
- モジュール4：モニタリングとサービスレベル指標
    - サービスレベル指標(SLI)
    - モニタリング
    - 観測性
- モジュール 5: SRE ツールと自動化
    - 自動化の定義
    - オートメーションフォーカス
    - 自動化の種類の階層
    - セキュアオートメーション
    - 自動化ツール
- モジュール6：アンチフラジリティと失敗からの学習
    - なぜ失敗から学ぶのか
    - アンチフラジリティのメリット
    - 組織のバランスを変える
- モジュール 7: SRE の組織的影響
    - 企業がSREを導入する理由
    - SRE導入のパターン
    - オンコールの必需品
    - 非の打ちどころのないポストもーてむ
    - SREとスケール
- Module 8: SREと、その他のフレームワーク、トレンド
    - SREと、その他のフレームワーク
    - 未来の姿
- その他の情報源
- 試験の準備
    - 試験条件、問題の重み付け、用語のリスト
- サンプル試験レビュー

DevOps Institute

Site Reliability Engineering (SRE) Foundation SM

試験条件

Site Reliability Engineering (SRE) Foundation℠・認定

Site Reliability Engineering (SRE) Foundation認定は、DevOps Instituteが提供する独立した資格です。この資格および関連コースの目的は、SREの基本的な用語、原則、および実践に関する知識を付与し、テストし、検証することです。SRE Foundation認定は、SREの基本的なコンセプトを理解し、サイトリライアビリティエンジニアリングの原則とエンジニアリングプラクティスを適用して運用活動を改善するためにSREをどのように利用できるかを理解することを目的としています。

受験資格

試験のための正式な前提条件はありませんが、DevOps Instituteは、SRE Foundation認定につながる試験の受験者を準備するために、以下を強く推奨しています。

- 受験者は、DevOps Instituteの認定教育パートナーが提供する正式な認定トレーニングコースの一部として、少なくとも16時間（講義と実習）を受講することが推奨されています。

試験運営

SRE Foundation認定は、DevOps Instituteの厳格なプロトコルと基準に基づいて認定、管理、運営されています。

難易度

SRE Foundationの認定は、コンテンツと試験の両方の構築にBloom Taxonomy of Educational Objectives（教育目標のブルーム分類法）を使用しています。

- SRE Foundation試験には、SREの概念や語彙に関する学習者の***知識を***問うブルーム1問題が含まれています（以下のリストを参照）。
- また、この試験には、これらの概念を文脈の中で***理解しているかどう***かを問うブルーム2問題が含まれています。

試験の形式
SRE Foundation認定資格を取得するには、合格点を獲得する必要があります。

試験の種類	多肢選択問題 40問
試験時間	60分
前提条件	認定されたDevOps Institute Education PartnerのSite Reliability Engineering（SRE）Foundationコースを修了することが推奨されます。
監修	いいえ
オープンブック	はい
合格点	65%
デリバリー	ウェブベース
バッジ	SREファンデーション認定

試験のトピック領域と問題の重み付け

SRE Foundation試験では、以下のトピック領域の知識が求められます。

トピックエリア	説明	最大質問数
SRE - 1: SREの原則と実践	SREの定義、歴史、SREとDevOps、SREの原則と実践	4
SRE - 2: サービスレベル目標とエラーバジェット	サービスレベル目標（SLO）、エラーバジェット、エラーバジェットポリシー、組織におけるSLOの設定についての理解	6
SRE - 3: トイルの削減	トイルを理解し、なぜそれが悪いのか、トイルを減らすための人間的、組織的な機会を理解する。	5
SRE - 4: モニタリングとサービスレベル指標	サービスレベル指標（SLI）の理解とサービスレベル目標（SLO）との関連性、モニタリングの状況、観察可能性、測定可能なサービス目標の設定	7
SRE - 5: SREツールと自動化	自動化の定義、DevOpsとSREの自動化の焦点、SREの自動化の種類、ツールの状況の概要	6
SRE - 6: アンチフラジリティと失敗からの学習	失敗から学ぶことのメリット、アンチフラジリティの定義、組織のバランスを変える、カオスエンジニアリング	4

SRE - 7: SREの組織的影響	なぜ組織はSREを採用するのか、SRE採用のパターン、SREの組織的影響、持続可能なインシデントレスポンス、責任のない死後の処理、SREの拡大	4
SRE-8: SREとその他のフレームワーク、将来について	SREとDevOps、アジャイル、ITSM、SREの進化、ネットワーク信頼性工学や顧客信頼性工学などのSREのスピンオフについて	4

概念・用語一覧

以下のSRE Foundationのコンセプトと語彙をブルーム1（知識）および2（理解）レベルで理解、理解、応用することが期待されます。

アジャイル	モニタリング
アンチフラジリティ	平均サービス回復時間（MTRS）
アプリケーション・パフォーマンス・マネジメント（APM）	MTTD（Mean Time to Detect Defects）
オートスケーリング	MTTR（Mean Time to Repair/Recover）
自動ロールバック	ネットワーク・リライアビリティ・エンジニア（NRE）
入手方法	非機能テスト
アマゾン ウェブ サービス（AWS）	観測性
非の打ちどころのないポストモーテム	オンコール
爆発半径	病理
官僚的な文化	信頼性
事業継続性	レジリエンス
カナリアテスト	応答時間
カオスエンジニアリング	スケーラビリティ
クラウドコンピューティング	ソフトウェア・デリバリー・ライフサイクル (SDLC)
ChatOps	セキュアオートメーション
カスタマー・リライアビリティ・エンジニア（CRE）	セルフヒーリング
データベース・リライアビリティ・エンジニア（DBRE）	サービスレベルアグリーメント（SLA）
DevOps	サービスレベルインジケーター（SLI）
誤差バジェット	サービスレベル目標（SLO）
誤差バジェットの方針	シミアン・アーミー
外部自動化	サイト・リライアビリティ・エンジニア（SRE）
防災訓練	Software-Defined Networking (SDN)
機能テスト	安定性

ヘリテージ・リライアビリティ・エンジニア（HRE）	テレメトリー
不変的なインフラストラクチャ	3つの道
インシデントレスポンス	トイル
内部の自動化	トラフィック量
ITIL	ベロシティ
ITサービスマネジメント(ITSM)	ウェストラム（組織の種類）
Kubernetes	
レイテンシー	

DevOps Institute

DEVOPS 用語集

この用語集は、試験に出題されるかどうかにかかわらず、重要な用語が含まれているため、参考として提供しています

DevOps 用語集

関連コース名は略称です。正式名称は最終頁に記載しています。

用語	定義
12-Factor App デザイン	モダンでスケーラブル、かつメンテナンス性に優れた Software-as-a-Service アプリケーションを構築するための方法論。
2 要素（2 段階）認証	ユーザーが 2 つの認証要素を提供すること。通常、最初にパスワードを入力し、次にデバイスに送信されたコード、共有秘密キー、物理トークン、生体認証などの 2 つ目の認証レイヤーを入力する。
3 つの道	DevOps の主要な原則。フロー、フィードバック、継続的な実験と学習。
A/B テスト	異なる顧客に異なるバージョンの EUT（エンドユーザーテスト）を展開し、顧客からのフィードバックで最適なものを決定する
A3 問題解決	「A3 問題解決報告書」と呼ばれるリーンツールを使用した、構造的な問題解決アプローチ。
Amazon Web Services (AWS)	Amazon Web Services(*AWS*)*は*、安全なクラウドサービスプラットフォームであり、計算能力、データベースストレージ、コンテンツ配信などの機能を提供し、ビジネスの拡大と成長を支援する。
API テスト	EUT の API が期待通りに機能するかどうかを判断するテスト。
Application Programming Interface (API)	特定の OS 用のアプリケーションを作成するため、またはモジュールやアプリケーション間のインターフェースとして使用されるプロトコル群。
CALMS モデル	John Willis、Damon Edwards、Jez Humble が提唱した DevOps の柱や価値観である Culture、Automation、Lean、Measurement、Sharing を考慮しています。
ChatOps	グループチャットと DevOps ツールとの統合を組み合わせた、技術およびビジネスオペレーションを管理するためのアプローチ（GitHub による造語）。ツールの例は以下の通り。Atlassian HipChat/Stride、Microsoft Teams、Slack。
CI Regression Test	リグレッションテストのサブセットで、ソフトウェアコンポーネントが構築された直後に実行される。スモークテストと同じ。
Cloudbees	商業的にサポートされた独自の自動化フレームワークツールで、エンタープライズレベルのサポートとアドオン機能を提供する。Jenkins と連携し、Jenkins を強化する。
COTS	市販の既成概念にとらわれないソリューション。商用オフザシェルフ。
Dev	アプリケーションエンジニアやソフトウェアエンジニアなど、ソフトウェア開発活動に携わる個人。
Device Under Test (DUT)	テスト対象デバイス（ルーターやスイッチなど）。

DevOps 用語集

DevOps	ソフトウェア開発者と IT 運用担当者の間のコミュニケーション、コラボレーション、統合を重視し、ソフトウェアの提供やインフラの変更を自動化する文化的・専門的な動き。ソフトウェアの構築、テスト、およびリリースを、迅速かつ頻繁に、そしてより確実に行うことができる文化と環境を確立することを目的としている。 (出典：Wikipedia)
DevOps インフラストラクチャ	DevOps システムを構成するツールと設備のセット全体。CI、CT、CM、CD の各ツールを含む。
DevOps カイゼン	カイゼンとは、日本語で「より良い方向への変化」を意味する言葉であり、大なり小なり、すべての従業員が参加し、組織の壁を越えて継続的に改善を行うという考え方。デイモン・エドワーズの「DevOps Kaizen」では、小さな改善（Little J's）を積み重ねることで、長期的に生産性を向上させることができることを示している。
DevOps コーチ	チームがアジャイル開発や DevOps の手法を習得するのを支援し、生産的な仕事の進め方やコラボレーションを可能にする。
DevOps スコア	組織内での DevOps の採用状況と、それに伴うデリバリーベロシティへの影響を示す指標。
DevOps ツールチェーン	アイデアから価値の実現まで、DevOps の継続的な開発とデリバリーのサイクルをサポートするために必要なツール。
DevOps パイプライン	DevOps インフラストラクチャを構成する、相互に接続されたプロセスの全体像。
DevOps の 7 本柱	DevOps システムの基盤となるの 7 つの明確な「柱」。「協調的な文化」、「DevOps のための設計」、「継続的インテグレーション」、「継続的テスト」、「継続的なデリバリーとデプロイメント」、「継続的モニタリング」、「弾力的なインフラとツール」。
DevSecOps	必要な安全性を犠牲にすることなく、最高レベルのコンテキストを持つ人々にセキュリティに関する決定をスピードとスケールで安全に分配することを目的とした、「全員がセキュリティに責任を持つ」という考え方。
DMZ (De-Militarized Zone)	公共のインターネットと内部の保護されたリソースの間にあるネットワークゾーン。外部に公開する必要のあるアプリケーション、サーバー、サービス（API を含む）は、通常、DMZ に置かれ、複数の DMZ を並列に配置することも珍しくない。
EggPlant	エンタープライズ・アプリケーションの機能テストと回帰テストを自動化する。Test Plant 社のライセンスを取得している。
EUT	被試験機器。テストされるエンティティの総称。これらの用語は、しばしば xUT という形式に省略され、"x "はテスト中のエンティティのタイプを表す。

DevOps 用語集

Fail Early	DevOps の考え方。デベロップメントパイプライン・デリバリーパイプラインの中で、重要な問題をできるだけ早期に発見することを優先することを指す。
Fail Often	DevOps の考え方。重要な問題をできるだけ早く、頻繁に見つけることを優先することを指す。
GUI テスト	テストの目的は、グラフィカルユーザーインターフェイスが期待通りに動作するかどうかを判断すること。
Helm チャートレジストリ	Helm チャートは、関連する Kubernetes リソースを記述するもの。Artifactory と Codefresh は、Helm チャートのマスターレコードを管理するためのレジストリをサポートしている。
ID	デジタルシステムで認識される人、機器、またはその両方の組み合わせの固有名。アカウント」または「ユーザー」とも呼ばれる。
Identity as a Service (IDAAS)	クラウドまたはサブスクリプションベースで提供されるアイデンティティおよびアクセス管理サービス。
Implementation Under Test（IUT）	EUT（被試験機器）はソフトウェアの実装。例：組み込みプログラムをテストする。
Infrastructure as Code	コード（スクリプト）を使ってインフラを設定・管理すること。
Infrastructure - as - a - Service (IaaS)	設定可能なコンピューティングリソースの共有プールに、オンデマンドでアクセスできる。
INVEST	質の高いユーザーストーリーの特徴を表す、ビル・ウェイクが作った記憶法。
IoT（Internet of Things）	ウェブベースのワイヤレスサービスを介してインターネットに接続し、潜在的には相互に接続する物理的デバイスのネットワーク。
ISO 31000	リスクマネジメントに関する原則と一般的なガイドラインを定めた規格群。
IT サービスマネジメント（ITSM）	ビジネスのニーズを満たす高品質な IT サービスの導入と管理。(ITIL 定義)
iTest	Spirent Communications 社がライセンス供与している自動テストケース作成用ツール。
ITIL	顧客を含むすべてのステークホルダーに最適な価値を提供するために、組織が IT サービスを提供・維持するために適用できるベストプラクティスのフレームワークを提供する。

DevOps 用語集

Jenkins	Jenkins はフリーウェアのツールで、継続的インテグレーションのタスク自動化を中心に、マスターオートメーションのフレームワークツールとして最も人気がある。Jenkins のタスク自動化は、時間指定のプロセスが中心となる。多くのテストツールなどが、Jenkins との連携を簡略化するプラグインを提供している。
Kubernetes	Kubernetes は、アプリケーションのデプロイ、スケーリング、および管理を自動化するためのオープンソースのコンテナオーケストレーションシステム。Google が開発し、現在は Cloud Native Computing Foundation が管理している。
Lab - as - a - Service (LaaS)	クラウドコンピューティングサービスのカテゴリーで、ラボのインフラストラクチャを構築・維持する複雑さを伴わずに、顧客がアプリケーションをテストできるラボを提供する。
LoadRunner	アプリケーションをテストするためのツールで、負荷時のシステムの動作やパフォーマンスを測定する。HP からライセンスを受けている。
Mean Time Between Deploys	デプロイメントの頻度を測定する。
Mean Time Between Failures (MTBF)	CI や IT サービスが、合意した機能を中断することなく実行できる平均時間。信頼性の測定に用いられることが多い。CI またはサービスが動作を開始してから故障するまでの時間（アップタイム）を測定する。(ITIL 定義)
Mean Time to Detect Defects (MTTD)	故障したコンポーネントやデバイスを検出するのに必要な平均時間。
Mean Time to Discovery	脆弱性やソフトウェアのバグ・欠陥が発見されるまでの期間。
Mean Time to Patch	脆弱性が発見されてから、環境にパッチを適用するまでの時間。
Mean Time to Repair/Recover (MTTR)	故障したコンポーネントやデバイスの修理/回復に要する平均時間。MTTR には、サービスの回復・復旧に必要な時間は含まれない。
Mean Time to Restore Service (MTRS)	CI や IT サービスに障害が発生してから、完全に復旧して通常の機能を提供するまでの時間（ダウンタイム）を測定するために使用される。保守性の測定によく用いられる。(ITIL の定義)。
Minimum Viable Process	プロセスやマイクロプロセスがその「完了の定義」を満たすために必要最小限の量。
Minimum Viable Product	人々が喜んで使ってくれるような十分な価値を提供しつつ、リリース可能な製品の最も最小のバージョン。実用最小限の製品。
Ops	品質保証アナリスト、リリースマネージャー、システムおよびネットワーク管理者、情報セキュリティオフィサー、IT オペレーションスペシャリスト、サービスデスクアナリストなど、システムやサービスを展開・管理するために必要な日々の運用活動に携わる人。

DevOps 用語集

OS 仮想化	アプリケーション同士の干渉を防ぐために、サーバーを「コンテナ」や「仮想環境」と呼ばれる複数のパーティションに分割する方法。
OUT	EUT（被試験機器）は、ソフトウェア・オブジェクトまたはオブジェクトのクラス。
Pages	CI/CD パイプラインの一部として、サポートする Web ページを自動的に作成するためのもの。
Plan-Do-Check-Act	W.エドワーズ・デミングが提唱した、プロセス管理・改善のための 4 段階のサイクル。デミングサイクル、PDCA と呼ばれることもある。
Platform - as - a - Service (PaaS)	クラウドコンピューティングサービスのカテゴリーで、顧客がインフラストラクチャを構築・維持する複雑さなしに、アプリケーションを開発・実行・管理できるプラットフォームを提供する。
Policy as Code	セキュリティの原則や概念は、コード（ソフトウェア、構成管理、自動化など）の中で十分に表現することができ、従来のような大規模なポリシーフレームワークの必要性は大幅に軽減されるという考え方。標準やガイドラインは、コードやコンフィギュレーションに実装され、自動的に実施され、コンプライアンス、差異、違反の疑いといった点で自動的に報告されるべきである。
Provision Platforms	インフラをプロビジョニングするためのプラットフォームを提供するツール（例：Puppet、Chef、Salt）。
QTP	Quick Test Professional。ソフトウェアアプリケーションの機能テストおよび回帰テストの自動化ツール。HP からライセンスを受けている。
Ranorex	デスクトップ、ウェブベース、モバイルのアプリケーションをテストするための GUI テスト自動化フレームワーク。ライセンスは Ranorex が取得している。
REST	Representation State Transfer の略。ワールドワイドウェブのソフトウェアアーキテクチャスタイル。
Restful API	HTTP などのネットワーク上の REST（Representational State Transfer）または RESTful サービスは、リクエストしたシステムが、ステートレスな操作（GET、POST、PUT、DELETE など）を用いて、リソースのテキスト表現（XML、HTML、JSON）に迅速かつ確実にアクセスし、操作するためのスケーラブルな相互運用性を提供する。
RESTful インターフェーステスト	API がその設計基準と REST アーキテクチャの期待値を満たしているかどうかを判断するテスト。
Return on Investment (ROI)	達成された利益と、その利益を達成するためのコストとの差を、パーセンテージで表したもの。
Review Apps	リアルタイムでコードをコミットして起動できるようにする（開発者がアプリケーションをレビューできるように環境を回転させる）。

DevOps 用語集

Robot Framework	Google が開発・サポートしている TDD フレームワーク。
Scaled Agile Framework (SAFE)	リーン・アジャイルの原則と実践を企業規模で適用するための、実証済みで一般に利用可能なフレームワーク。
SCARF モデル	人が社会的に相互作用する方法について、神経科学から得られた重要な発見をまとめたもの。
Security as Code	セキュリティを自動化し、DevOps ツールやプラクティスに組み込むことで、ツールチェーンやワークフローに不可欠な要素とする。
Selenium	GUI や Web アプリケーションのソフトウェアテストを行う人気のオープンソースツール。
Service Level Agreement (SLA)	IT サービスプロバイダーとその顧客の間で交わされる書面による契約で、主要なサービス目標と両当事者の責任を定義する。SLA は、複数のサービスまたは顧客を対象とすることができる。(ITIL 定義)
Service Level Indicator (SLI)	SLI は、サービスに関する定量的なデータを伝えるために使用され、通常は SLO に対するサービスのパフォーマンスを測定する。
Service Level Objective (SLO)	SLO とは、製品やサービスがどのように機能すべきかを示す目標。SLO は、組織がサービスに何を期待しているかに基づいて設定される。
SilkTest	エンタープライズ・アプリケーションの機能テストと回帰テストを自動化しする。ボーランドのライセンスを受けている。
Simian Army	Simian Army は、Netflix が開発した障害を誘発するツール群。最も有名な例は Chaos Monkey で、カオスエンジニアリングのアプローチの一環として、本番環境のサービスをランダムに終了させる。
SOAP	コンピュータ間で情報を交換するための XML ベースのメッセージングプロトコル。
Software - as - a - Service (SaaS)	ソフトウェアがサブスクリプションベースでライセンスされるクラウドコンピューティングサービスのカテゴリー。
Spotify スクワッドモデル	大企業のチームがスタートアップのように行動し、軽快に活動できるようにするための組織モデル。
SRE (SRE)	ソフトウェアエンジニアリングの側面を取り入れ、それをインフラや運用の問題に応用した学問分野。主な目標は、スケーラブルで信頼性の高いソフトウェアシステムを構築すること。
Stormstack	時間ベースではなく、イベントトリガーをベースにした商用オーケストレーションツール。
StoStaKee	Stop、Start、Keep の頭文字をとったもので、過去の出来事に焦点を当てたインタラクティブなタイムボックス形式のエクササイズ。

DevOps 用語集

SUT	EUT（被試験機器）は、システム全体。例：銀行の窓口の機械がテストされる。
Test Fast	早くテストすることを意味する継続的テストの思想。
The Power of TED	「The Power of TED」は、カープマンのドラマトライアングル（犠牲者、迫害者、救済者の役割を持つ）に代わるものです。エンパワーメントダイナミック(TED)は、クリエーター、チャレンジャー、コーチの役割を持つ解毒剤の役割を提供し、人生の課題に対してより前向きなアプローチを可能にする。
Training From the Back of the Room	4C 教育設計マップ（Connection, Concept, Concrete Practice, Conclusion）を用いた、アジャイルの価値観と原則に沿った加速学習モデル。
Web 統合開発環境（Web IDE）	Web クライアント型の統合開発環境を持つツール。ローカルの開発ツールを使用することなく、開発者の生産性を高めることが可能。
Web アプリケーションファイアウォール (WAF)	アプリケーションに送信されるトラフィックを調査し、悪意のあるものをブロックできるツール。
Westrum（組織タイプ）	ロン・ウェストラムは、3 つのタイプの組織文化を含む組織文化の類型論を開発した。病理学的（権力志向）、官僚主義的（規則志向）、発生的（業績志向）。
Wiki	Confluence のようなリッチなコンテンツの Wiki を作成するツールを使うことで、ナレッジの共有が可能となる。
アーキテクチャー	コンピュータのハードウェア、ソフトウェア、またはその両方を組み合わせた基本的な設計。
アーティファクト	ドキュメント、テスト計画、イメージ、データファイル、実行可能なモジュールなど、ソフトウェア開発プロジェクトにおけるあらゆる要素。
アーティファクトリポジトリ	バイナリ、レポート、メタデータを保存する。ツール例：JFrog Artifactory、Sonatype Nexus。
アイデンティティ/アクセスマネジメント (IAM)	適切な人が適切にテクノロジーリソースにアクセスできるようにするための方針、手順、ツール。
アウトカム	意図した結果、または実際の結果。
アクセス・プロビジョニング	ユーザーアカウントの作成、ルールやロールによる電子メールの認証、システムや環境への新規ユーザーの参加に伴う物理的なリソースのプロビジョニングなどのタスクを調整するプロセス。
アクセスマネジメント	定義された基準（マッピングされた役割など）に基づいて、認証された ID に認可されたリソース（データ、サービス、環境など）へのアクセスを許可する一方で、未認可の ID によるリソースへのアクセスを防止する。

DevOps 用語集

アジャイル	複雑なプロジェクトにおいて、タスクを小さな「スプリント」と呼ばれる作業に分割し、頻繁に計画の再評価と修正を行うプロジェクト管理手法。
アジャイル（形容詞）	素早く簡単に動くことができ、協調性がある。素早く考え、理解することができ、問題を解決し、新しいアイデアを持つことができる。
アジャイルコーチ	チームがアジャイル開発や DevOps の手法を習得するのを支援し、生産的な仕事の進め方やコラボレーションを可能とする。
アジャイルサービスマネージャ	アジャイルサービスマネジメントの領域専門家であり、アジャイルサービスマネジメント・チームのコーチであり保護者。
アジャイルサービスマネジメント	ITSM プロセスがアジャイルの価値観を反映し、必要なときに必要な方法で顧客の成果を促進するサービスを効果的かつ効率的に提供するために、「過不足のない」コントロールと構造で設計されていることを確認するためのフレームワーク。
アジャイルサービスマネジメントチーム	少なくとも 3 人（顧客または実務者を含む）からなるチームで、1 つのマイクロプロセスあるいは完全なサービスマネジメントの実践に責任を持つ。
アジャイルサービスマネジメントのアーティファクト	プラクティスバックログ、スプリントバックログ、インクリメント
アジャイルサービスマネジメントのイベント	プラクティス／マイクロプロセスプランニング、スプリント、スプリントプランニング、プロセススタンドアップ、スプリントレビュー、スプリントレトロスペクティブ
アジャイルサービスマネジメントの役割	アジャイルプラクティスオーナー、アジャイルサービスマネジメントチーム、アジャイルサービスマネージャ
アジャイルソフトウェア開発	自己組織化された機能横断的なチームのコラボレーションを通じて、要件とソリューションを進化させるソフトウェア開発手法のグループ。通常、スクラムまたはスケールドアジャイルフレームワークのアプローチで実施される。
アジャイルソフトウェア開発宣言	ソフトウェア開発における反復的で人間中心のアプローチを導くための、価値と原則の正式な宣言。http://agilemanifesto.org
アジャイルな企業	予期せぬ課題や出来事、好機に迅速に対応できる、動きの速い、柔軟で強健な企業。
アジャイルの原則	アジャイル・マニフェストを支える 12 の原則。
アジャイルプラクティスオーナー	サービスマネジメントの全体的な品質に責任を持つ役割で、プラクティス・バックログの所有者。

DevOps 用語集

アジャイルプロセス	組織が最も迅速かつ効果的・効率的な方法でサービスがアウトカムに到達できるような「過不足のない」構造とコントロールを提供する。 理解しやすく、従いやすく、アーティファクトよりもコラボレーションとアウトカムを重視する。
アジャイルプロセスエンジニアリング	出荷可能なプロセスのインクリメントやマイクロプロセスを短期間で繰り返し設計する、反復的・漸進的なアプローチ。
アジャイルプロセスの改善	アジャイルプロセスエンジニアリングによって導入された ITSM のアジリティが、ITSM の継続的改善へのコミットメントの一環として、継続的に見直され調整されていることを確認する。
アジャイルポートフォリオマネジメント	プロジェクトや裁量労働への継続的な投資を形成し、管理するために、進行中のプロジェクトや提案された将来のイニシアチブを評価することを含む。CA の Agile Central や VersionOne がその例です。
アドバイスプロセス	意思決定を行う者は、その決定によって重要な影響を受けるすべての人と、その問題についての専門知識を持つ人に助言を求めなければならない。受け取った助言は考慮されなければならないが、必ずしも受け入れたり従ったりする必要はない。アドバイスプロセスの目的は、コンセンサスを形成することではなく、意思決定者が可能な限り最善の意思決定を行えるように情報を提供することである。アドバイスのプロセスに従わないと、信頼が損なわれ、ビジネスに不必要なリスクが生じる。
アドミニストレーションテスト	エンドユーザーテスト（EUT）が期待通りに管理タスクを処理できるかどうかを判断するためのテスト。
アナリティクス	テスト結果を、分析方法や基準に沿って整理された形で処理／提示する。
アプリケーション・パフォーマンス・マネジメント（APM）	ソフトウェアアプリケーションのパフォーマンスと可用性を監視・管理すること。期待されるサービスレベルを維持するために、複雑なアプリケーションのパフォーマンス問題を検出・診断することに努める。
アプリケーションテスト	アプリケーションがその要件や期待される動作に沿って動作しているかどうかを判断するテスト。
アプリケーションリリース	自動化（コードコミット時のリリース）を含む継続的デリバリーパイプラインの機能を管理。
アプリケーションリリースオートメーション（またはオーケストレーション）	自動化（コードコミット時のリリース）、環境モデリング（エンドツーエンドのパイプラインステージ、アプリケーションのバイナリ、パッケージ、その他のアーティファクトのターゲット環境へのデプロイ）、リリースコーディネーション（プロジェクト、カレンダー、スケジューリング管理、変更管理や IT サービスサポート管理との統合）を含む継続的デリバリパイプラインの機能を制御。ARA または ARO。
アンチパターン	よくある再利用だが、問題の解決にはならないもの。

DevOps 用語集

アンチフラジリティ	侵襲要因、衝撃、変動性、ノイズ、ミス、欠点、攻撃、故障などの結果を受けて、システムの能力を高めて成長させる特性。
アンドン	組立ラインの作業者が不具合を発見したときに生産を停止し、直ちに支援を要請することができ、さらにその権限も与えられるシステム。
イマーシブラーニング	コーチングと練習でチームを導き、新しい働き方を身につけさせる学習法。
イミュータブル	作成後に状態を変更することができないオブジェクト。反意語はミュータブルオブジェクトで、作成後に状態を変更することができる。
イミュータブルなインフラストラクチャ	インスタンス（サーバー、コンテナなど）をインスタンス化し、エラーが起こりやすく、時間のかかるパッチやアップグレード（つまり変異）を行う代わりに、別のインスタンスに置き換えて変更を導入したり、適切な動作を保証したりする。
イメージベーステスト	ビルドイメージには、あらかじめテストケースが割り当てられ、ビルドによるイメージの変更にマッチするように選択される。
インクリメンタルロールアウト	サービスに大きな変更を加えるのではなく、小さな変更を徐々に加えていくこと。ユーザーは新しいバージョンのサービスに徐々に移行し、最終的にはすべてのユーザーが移行する。ブルー／グリーンデプロイメントなど、色のついた環境で表現されることもあります。
インクリメント	スプリントの結果として完成した、出荷可能な製品。
インシデント	IT サービスの計画外の中断または IT サービスの品質の低下をいう。サービスを中断させる、または中断させる可能性のある事象を含む。(ITIL 定義)
インシデントマネジメント	ビジネスへの影響を最小限に抑え、合意されたレベルのサービス品質が維持されるように、可能な限り迅速に正常なサービス動作を回復するプロセス。(ITIL の定義)。 サービスインシデントの「誰が」「何を」「いつ」を把握し、サービスレベル目標の達成に向けてこのデータを活用すること。
インシデントレスポンス	セキュリティ侵害や攻撃（インシデントとも呼ばれる）の後遺症に対処し、管理するための組織的なアプローチのこと。被害を最小限に抑え、復旧にかかる時間とコストを削減する方法で対処することを目的とする。
インセンティブモデル	目的を達成するためにタスクを完了するように人々を動機付けるために設計されたシステム。このシステムは、動機付けのために正の結果または負の結果を採用することができる。
インフラストラクチャ	IT サービスを開発、テスト、提供、監視、制御、またはサポートするために必要なハードウェア、ソフトウェア、ネットワーク、設備などのすべての要素を指す。IT インフラという言葉には、情報技術のすべてが含まれるが、関連する人、プロセス、文書は含まれない。(ITIL 定義)
インフラストラクチャテスト	EUT（被試験機器）が動作するためのフレームワークを検証すること。例えば、特定のオペレーティングシステム・ユーティリティーがターゲット環境において期待通りに機能することを検証する。

DevOps 用語集

ウイルス（コンピュータウイルス）	ファイルに添付された悪意のある実行コードで、感染したファイルがシステムからシステムへと受け渡されることで広がるもので、無害（ただし迷惑）な場合もあれば、データの修正や削除を行う場合もある。
ウィルバーのクワドラント	人間の一般的なアプローチの 4 つのモードを認識するモデル。2 つの軸が使用され、1 つの軸では人々は個人性と集団性のどちらを好むかを示す。
ウォータースクラムフォール	ウォーターフォール開発とスクラム開発を組み合わせたハイブリッド型のアプリケーションライフサイクルマネジメント。一定の期間で完成させることができる。
ウォーターフォール（プロジェクトマネジメント）	ソフトウェアの設計・開発プロジェクトを管理するための直線的・逐次的なアプローチで、進捗状況が（滝のように）着実に（そして逐次的に）下に向かって流れていくように見られる。
エピック	複数のスプリントにまたがって作業する必要のある、関連するユーザーストーリーの集合体。
エラートラッキング	アプリケーションで発生する可能性のあるエラーを、関連するデータとともに簡単に発見し、表示するツール。
エラーバジェット	特定の期間内にどれだけサービスの信頼性が低下することが許されるかを決定する、明確で客観的な指標。
エラーバジェットポリシー	特定の期間に特定のサービスのエラーバジェットを使い果たしたときにチームが取るべき活動を列挙したもの。
エラスティックインフラストラクチャ	エラスティックは、クラウドコンピューティングで一般的に用いられる用語で、IT インフラの安定性、パフォーマンス、セキュリティ、ガバナンス、コンプライアンスなどのプロトコルに支障をきたすことなく、容量やサービスを迅速に拡大・縮小できる能力を指す。
エリクソンの発達段階論	エリク・エリクソン(1950, 1963)は、乳幼児期から成人期までの 8 つの段階からなる心理社会的発達の精神分析理論を提唱した。各段階において、人は心理社会的危機を経験し、それが人格形成にプラスまたはマイナスの結果をもたらす可能性があるとした。
オーケストレーション	複数のツールを連携させてツールチェーンを形成する、ビルディングオートメーションのアプローチ。
オープンソース	エンドユーザー組織やベンダーが独自の目的のために変更できるように、ソースコードとともに配布されるソフトウェア。
オペレーションマネジメント	IT サービスとそれを支える IT インフラを合意されたレベルで提供・サポートするために必要な日々の活動を行う機能。(ITIL)
オンコール	決められた時間内に誰かが待機し、その間に発生したプロダクション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しておくこと。

DevOps 用語集

カープマンのドラマの三角関係	人間関係の社会的なモデル。三角形は、対立する人々の間で起こりうる破壊的な相互作用の一種を図式化したものである。
カイゼン	継続的な改善の実践。
カオスエンジニアリング	プロダクション環境のソフトウェアシステムで実験を行い、システムが予期せぬ状況に耐えられることを確信するための規律。
カスタマーリライアビリティエンジニア (CRE)	SRE の原則と教訓を顧客に適用する。
カナリア・テスト	カナリア（カナリア・テストとも呼ばれる）とは、テストを志願していない少数のエンドユーザーにコードの変更をプッシュすることです。インクリメンタル・ロールアウトに似ていますが、ユーザーベースのごく一部を最初に新しいバージョンに更新することです。この一部のユーザー（カナリア）が、「炭鉱のカナリア」の役割を果たします。何か問題が発生した場合には、リリースをロールバックすることで、ごく一部のユーザーにのみ影響を与えることができます。
ガバナンスリスクマネジメント／コンプライアンス (GRC)	ポリシー、コンプライアンス要件、脆弱性データ、場合によっては資産目録、事業継続計画など、ガバナンス、コンプライアンス、リスク管理のデータを集中させることを目的としたソフトウェアプラットフォーム。セキュリティガバナンスに特化した文書・データリポジトリ。または、IT/セキュリティガバナンス、リスク管理、コンプライアンス活動を専門に行う人のチーム、多くの場合、技術系ではないビジネスアナリスト資源。
カルチャーアイスバーグ	文化の観察可能な要素（水面上）と観察不可能な要素（水面下）の違いを視覚化したメタファー。
カレントステートマップ	バリューストリームマップの一種で、現在のプロセスがどのように機能しているのか、どこに断絶があるのかを特定するのに役立つ。
カンバセーション・カフェ	カフェや会議室、教室など、この世界を理解するために人々が集まる場所で行われる、オープンな会話形式の催し。
カンバン	プロセスでの仕事の流れを管理可能なペースで引き出す仕事の方法。
カンバンボード	チームが仕事を整理し、可視化し、管理するためのツール。
キーワードベーステスト	テストケースは、テストに役立つプログラムを参照するために、あらかじめ定義された名前で作成される。
キャパシティ	あるスプリントのために利用可能なエンジニアリング時間の総量の見積もり。
キャパシティテスト	EUT（被試験機器）がユーザー数、セッション数、総帯域幅などの予想される負荷を処理できるかどうか を判断するテスト。

DevOps 用語集

キャプチャー／リプレイ	テストケースが EUT（被試験機器）とのライブインタラクションを、ツールで再生できる形式でキャプチャすることで作成される。例：Selenium
キューブラー・ロスの変化曲線	大きな変化に対する個人や組織の反応の段階を説明し、予測する。
ギルド	誰でも参加できる「利益共同体」であり、通常は組織全体を横断する。コミュニティ・オブ・プラクティスに似ている。
クラウドコンピューティング	プライベートデータセンターのローカルサーバーではなく、インターネット上にホストされたリモートサーバーを使用してアプリケーションをホストすること。
クラウドネイティブ	クラウドコンピューティングのために設計されたアプリケーション。NCA（Native Cloud Application）。
クラスタコスト最適化	モニタリングを用いてコンテナクラスタを分析し、リソース展開モデルを最適化する。ツール例：Kubecost、Replex、Cloudability など。
クラスタモニタリング	Kubernetes などのクラスターで動作するデプロイ環境の健全性を知ることができるツール。
クラスタリング	複数のコンピューター（ノードまたはメンバーと呼ばれる）が、高速ネットワークで接続されたクラスターとして動作し、1 つのシステムとして機能する。
グラスボックス	クリアボックステスト、ホワイトボックステストと同義。
クリアボックス	グラスボックステスト、ホワイトボックステストと同義。
グレイボックス	テストケースは、EUT（被試験機器）の内部設計構造に関する限られる。
ケイデンス	イベントの流れやリズム。
ゲートコミット	以下のようなすべての CD パイプラインステージ間で推進される変更の基準を定義し、コンセンサスを得る。Dev から CI ステージ、CI からパッケージング/デリバリーステージ、デリバリーからデプロイ/プロダクションステージ。
コードカバレッジ	テストで実行されたコードユニットをカウントすることで、ホワイトボックステストのカバレッジを測定すること。コード ユニットとは、コード ステートメント、コード ブランチ、コード モジュール内の制御パスやデータ パスなどを指す。
コードリポジトリ	開発者がコードをコミットして共同作業を行うためのリポジトリ。過去のバージョンを追跡し、同じコードの矛盾したバージョンを特定することもできる。「リポジトリ」または「レポ」とも呼ばれる。
コードレビュー	ソフトウェアエンジニアが、お互いにソースコードを検査しコーディングやコードのフォーマットの誤りを検出すること。

DevOps 用語集

コード品質	静的コード解析も参照。Sonar と Checkmarks は、コード品質の 7 つの主要な側面（コメント、アーキテクチャ、重複、ユニットテストカバレッジ、複雑さ、潜在的な欠陥、言語規則）を自動的にチェックするツールの例。
ゴールシーキングテスト	EUT（被試験機器）がピーク性能に達するまで段階的にストレスを掛け、EUT（被試験機器）の性能境界を決定するテスト。例：エラーなしで処理できる最大のスループットを決定する。
ゴールチェーン	ローマン・ピヒラーが考案した、製品開発プロセスのすべてのレベルで目標をリンクさせ、共有するための手法。
ゴールデンイメージ	仮想マシン（VM）、仮想デスクトップ、サーバー、ハードディスクドライブなどのテンプレート。(TechTarget)
ゴールデンサークル	サイモン・シネックが提唱したモデルで、「何を」「どのように」に焦点を当てる前に、ビジネスの「なぜ」を理解することを重視する。
ゴールマンによる「6 つのリーダーシップスタイル」	ダニエル・ゴールマン（2002）は、「6 つのリーダーシップ・スタイル」を作成し、研究の結果、リーダーは常にこれらのスタイルのいずれかを使用していることを発見した。
コッターのデュアル OS	ネットワークの起業家的能力と伝統的な階層構造の組織的効率性を組み合わせた二重の運営システムの必要性を提唱する。
コラボレーション	人々は共通の目標に向かって他の人と共同で作業をする。
コラボレーション文化	すべての人に適用される文化で、期待される一連の行動、言語、お互いに受け入れられる仕事のやり方が組み込まれています。 リーダーシップによって強化されます。
コルブの経験学習モデル	コルブが 1984 年に発表した学習スタイルモデル。経験学習理論は、4 段階の学習サイクルと 4 つの学習スタイルという 2 つのレベルで機能している。
コンウェイの法則	システムを設計する組織は、その組織のコミュニケーション構造のコピーであるデザインを作るという制約がある。
コンテナ	ソフトウェアを、開発、出荷、展開のために、実行に必要なすべてのもの（コード、ランタイム、システムツール、システムライブラリ、設定）を含む、軽量でスタンドアローンの実行可能なパッケージにする方法。
コンテナスキャニング	アプリケーションのコンテナイメージを構築する際に、ツールを使ってセキュリティスキャンを実行し、コードがデプロイされる環境に既知の脆弱性がないことを確認する。ツール例：Blackduck、Synopsis、Synk、Claire、klar など。
コンテナネットワークセキュリティ	コンテナクラスタ上で他のアプリケーションと共に実行できるアプリケーションが、他のアプリケーションの意図しない使用や意図しないネットワークトラフィックがないことを証明するために使用される。

DevOps 用語集

コンテナレジストリ	コンテナイメージのための安全でプライベートなレジストリ。一般的には、ビルドツールからイメージを簡単にアップロードおよびダウンロードできる。ツール例：Docker Hub、Artifactory、Nexus など。
ザ・スプリント	4 週間未満の期間で、1 つの仕事が完了すること。
サーバーレス	基盤となるインフラや依存関係を必要とせず、実行環境の構築を代行するサービスプロバイダー（一般的にはクラウド）によってコードの一部が実行されるコード実行のパラダイム。AWS の Lambda 関数や Azure Functions が例。
サービス	特定のコストやリスクを所有することなく、顧客が達成したい結果を促進することで、顧客に価値を提供する手段のこと。
サービスデスク	サービス提供者とユーザーの間の単一のコンタクトポイント。 Service Now のようなツールは、サービスのライフサイクルを管理するだけでなく、内部および外部のステークホルダーのエンゲージメントにも使用される。
サイクルタイム	作業開始から納品可能になるまでの時間を示す指標。
サニティテスト	ソフトウェアが機能しているかどうかを判断する、非常に基本的なテストのセット。
サプライヤー	IT サービスを提供するために必要な商品やサービスを供給する責任を負う外部（第三者）のサプライヤー、メーカー、ベンダー。
システムオブレコード	データ要素やデータ・エンティティの権威あるデータ・ソース。
システムテスト	完全なシステムが意図された構成で期待通りに動作するかどうかを判断する。
システム思考の法則	ピーター・センゲは著書「The Fifth Discipline」の中で、ビジネスシステムの理解を助け、複雑なビジネス問題に対処するための行動を特定するための 11 の法則を紹介している。
シフトレフト	テストを早期かつ頻繁に行うことで、ソフトウェア開発プロセスに品質を組み込むことを目的としたアプローチ。この概念は、セキュリティアーキテクチャ、ハードニングイメージ、アプリケーションセキュリティテストなどにも適用される。
シンセティック・モニタリング	アクティブ・モニタリング、またはセマンティック・モニタリングとも呼ばれる。アプリケーションの自動テストのサブセットを定期的にシステムに対して実行し、その結果はモニタリングサービスに反映され、障害が発生した場合にはアラートが出される。
スクラム	複雑なプロジェクトにおいて、チームが効果的にコラボレーションするためのシンプルなフレームワーク。スクラムは、チームがイノベーションを発揮して、他の方法では乗り越えられないような課題を解決するために、「ちょうどよい」構造を作り出すための小さなルールを提供する。(Scrum.org)

DevOps 用語集

スクラムチーム	スクラムフレームワークを使用して、反復的かつ漸進的に製品を提供する自己組織化されたクロスファンクショナルチーム。スクラムチームは、プロダクトオーナー、開発者、およびスクラムマスターで構成される。
スクラムの価値基準	スクラムのフレームワークを支える基本的な価値観と資質。確約（commitment）、勇気（courage）、集中（focus）、公開（openness）、尊敬（respect）。
スクラムの柱	スクラムのフレームワークを支える柱には以下のものがある。「透明性」「検査」「適応」。
スクラムマスター	スクラムのプロセスリーダーシップ（スクラムのプラクティスが理解され、遵守されていることを確認すること）を提供し、障害物を取り除くことでスクラムチームをサポートする個人。
スクワッド	クロスファンクショナルで、共同で配置され、自律的で自己管理されたチーム。
スケーラビリティ	サービスの特性の一つで、負荷が増加または拡大した場合に、それに対処して実行する能力。
スケジューリング	変更を本番にリリースするための計画を立てるプロセス。
スティック	負のインセンティブ：望ましくない行動を思いとどまらせたり、罰したりするためのもの。
ステークホルダー	組織、プロジェクト、または IT サービスに利害関係を持つ人。ステークホルダーには、顧客、ユーザー、サプライヤーなどが含まれる。(ITIL の定義)。
ステータスページ	顧客やユーザーにサービスの状況を簡単に伝えることができるサービスページ。
ストラテジックスプリント	4 週間以内のタイムボックス型のスプリントで、プラクティスプランニングで定義された戦略的要素を完成させ、チームがプロセスのアクティビティの設計に移れるようにする。
ストレージセキュリティ	データ・ストレージ・システムやエコシステム、およびこれらのシステムに存在するデータの安全性を確保するためのセキュリティの専門分野。
スナップショット	特定のビルドの合格/不合格の結果を報告する。
スニペット	保存されたコードスニペットを共有することで、特定のコード部分でのコラボレーションを可能にする。また、コードスニペットを他のコードベースで使用することもできる。BitBucket や GitLab がこれにあたる。
スパイウェア	ユーザーの知らないうちにコンピュータにインストールされ、ユーザーのコンピュータでの活動に関する情報を脅威エージェントに送信するソフトウェア。

DevOps 用語集

スプリント	2〜4 週間の期間で、製品の作業の一部分が完了すること。
スプリント（スクラム）	製品の機能のインクリメントが実装されるまでの、タイムボックス化された作業の繰り返しのこと。
スプリントゴール	スプリントの目的・目標であり、解決しようとしているビジネス上の問題として表現されることが多い。
スプリントバックログ	スプリント目標を実現するために完了しなければならない作業を表すバックログのサブセット。
スプリントプランニング	スプリント目標、スプリント期間中に完了する製品バックログのインクリメント、およびその完了方法を定義する 4〜8 時間のタイムボックス型イベント。
スプリントレトロスペクティブ	1.5〜3 時間のタイムボックスイベントで、チームは最後のスプリントをレビューし、次のスプリントのための改善点を特定し、優先順位付けを行う。
スプリントレビュー	4 時間以内のタイムボックスイベントで、チームとステークホルダーがスプリントの成果を確認し、プロダクトバックログを更新する。
スモークテスト	ソフトウェアコンポーネントが構築された直後に実行される、基本的な機能テストのセット。継続的インテグレーションの会期テストと同義。
セキュアオートメーション	安全な自動化は、配信パイプラインで使用されるツールを保護することで、ヒューマンエラー（および故意の妨害行為）の可能性を排除する。
セキュリティ（情報セキュリティ）	コンピュータシステムのデータの機密性、完全性、可用性を悪意のある者から守るための行為。
セキュリティテスト	試験の目的は、被測定機がそのセキュリティ要件を満たしているかどうかを判断することである。例としては、被測定機がログイン認証情報を適切に処理するかどうかを判断する試験がある。
セルフヒーリング	サービスや基盤となる環境が自動的に問題を検出して解決する能力。これにより、人間が手動で介入する必要がなくなる。
ソースコードツール	重要な資産（アプリケーションとインフラ）のソースコードを、単一の真正のソースとして管理するためのリポジトリ。
ソフトウェアバージョン管理システム	ソフトウェアの変更を管理するために使用されるリポジトリツール。例としては Azure DevOps、BitBucket、Git、GitHub、GitLab、VSTS。
ソフトウェア開発ライフサイクル(SDLC)	高品質なソフトウェアを設計、開発、テストするためのプロセス。
ソフトウェア構成分析	ソースコード中のライブラリや関数に既知の脆弱性がないかどうかをチェックするツール。

DevOps 用語集

ソフトウェア定義ネットワーク(SDN)	ソフトウェアアプリケーションを用いてネットワークをインテリジェントに集中制御、または「プログラム」することを可能にするネットワークアーキテクチャのアプローチ。
ターゲットオペレーティングモデル	組織のオペレーティング・モデルの望ましい状態を記述したもの。
タイムトゥバリュー	機能またはサービスからビジネスが価値を実現するまでにかかる期間。
タイムトゥマーケット	アイデアが発想されてから、顧客に提供されるまでの期間。
タイムトラッキング	個々の課題、その他の仕事やプロジェクトの種類に応じて、時間を追跡するツール。
タイムボックス	スクラムイベントの最大継続時間。
タグベーステスト	テストとコードモジュールには、あらかじめタグが割り当てられ、事前に割り当てられたタグにマッチしたビルドが選択される。
ダッシュボード	まとめられたテスト結果をグラフィカルに表示する。
タッチタイム	リーン生産方式で、製品が実際に作業され価値が付加されている時間。
チームダイナミクス	チームがどのように連携しているかを測る指標。チーム文化、コミュニケーションスタイル、意思決定能力、メンバー間の信頼関係、チームの変化への意欲などが含まれる。
チェックイン	ソフトウェアの変更をシステムのバージョン管理システムに提出すること。
チェックボックストラップ	監査中心の視点が、全体的なセキュリティ目標を考慮することなく、コンプライアンス要件の「チェックボックス」にのみ焦点を当てている状況。
チェンジリーダー開発モデル	ジム・カンテラが提唱する、チェンジリーダーの能力を 5 段階に分けたモデル。
チャプター	同じようなスキルを持ち、同じ部族の中で同じ一般的な能力領域で仕事をしている人たちの小さな集まり。チャプターは定期的に集まり、課題や専門分野について話し合い、共有、スキル開発、再利用、問題解決を促進する。
チャプターリード	Spotify モデルにおけるスクワッドラインマネージャー。従来のピープルマネジメント業務を担当し、日々の業務に携わり、個人と部門の能力を成長させる役割を担う。
ツール	EUT（被試験機器）やインフラをオーケストレーション、自動化、シミュレート、モニターするツール。
ツールチェーン	単一ベンダーのソリューションではなく、タスクに特化した補完的なツールの統合セットを使用して、エンド・ツー・エンドのプロセスを自動化することを意味する。

DevOps 用語集

ティール組織	組織の運営において、それまでの世界観をすべて含めた意識レベルを提唱する新しい組織パラダイム。
デイリースクラム	スプリント期間中、チームが翌日の作業を再計画するための 15 分以内の日次のタイムボックスイベント。
データベースリライアビリティエンジニア (DBRE)	プロダクション環境の、すべてのユーザー向けサービスを支えるデータベースシステムを円滑に運営する責任者。
データロスプロテクション(DLP)	サービス環境や組織内からファイルやコンテンツが削除されるのを防ぐためのツール。Data Loss Protection。
テスター	システムやサービスのテストに責任を持つ人。
テストアーキテクト	EUT（被試験機器） のエンド・ツー・エンドの全体的なテスト戦略を定義する責任を有する者。
テストアーティファクトリポジトリ	テストに使用したファイルのデータベース。
テストキャンペーン	テストキャンペーンには、1 つまたは複数のテストセッションが含まれる。
テストケース	データや構成情報を含む一連のテストステップ。テストケースは、EUT（被試験機器）の少なくとも 1 つの属性をテストする特定の目的を持つ。
テストスイート	特定の時間に 1 つのビルドで一緒に実行されるテストケースのセット。
テストスクリプト	テストケースの自動化。1 つのテストスクリプトに、データに応じて 1 つまたは複数のテストケースを実装することができます。
テストセッション	特定の時間に 1 つのビルドで一緒に実行される、1 つまたは複数のテストスイートのセット。
テストタイプ	テストの目的が何であるかを示すもの。
テストツール	ビルドを通す前にコードの品質を検証するツール。
テストトレンド	判断の履歴。
テストバージョン	特定のビルドのテストに使用されたファイルのバージョン。
テストハーネス	テストの自動化を可能にするツールのこと。テストの実行に必要なシステム・テスト・ドライバやその他のサポート・ツールを指します。テスト対象のソフトウェアと相互作用する小さなプログラムであるスタブやドライバを提供する。
テストフレームワーク	自動テストが設計・実装されるプロセス、手順、抽象的な概念、環境のセット。
テストメソドロジー	テストで使用される一般的な手法を特定する用語の総称。例として、ホワイトボックス、ブラックボックス

テストロール	このクラスの用語は、テストに関連する人々の一般的な役割と責任を特定するもの。
テスト階層	テストをグループにまとめることを表す用語の総称。
テスト環境	テスト環境とは、テストが行われるオペレーティングシステム（Linus、Windows バージョンなど）、ソフトウェアの構成（パラメータオプションなど）、動的条件（CPU やメモリの使用率など）、物理的環境（電源、冷却など）を指す。
テスト期間	テストを実行するのにかかる時間。例：1 回のテストにかかる時間
テスト駆動開発(TDD)	開発者がコードを作成する前にテストを作成するソフトウェア開発プロセス。 1. テストを書く 2. そのテストと関連する他のテストを実行して、失敗を確認する 3. コードを書く 4. テストを実行する 5. 必要に応じてコードを修正する 6. 繰り返す ユニットレベルのテストやアプリケーションテストは、テスト対象となるコードの前に作成される
テスト結果（トレンドベース）	相関係数のマトリクスにより、テスト結果（評点）に応じてテストケースとコードモジュールを関連付ける。
テスト結果リポジトリ	テスト結果のデータベース
テスト作成手法	テストケースを作成するための方法論を指すテスト用語の一種である。
テスト選択手法	EUT（被試験機器）の、あるバージョンで実行するテストを選択するために使用される方法を指す。
テスト対象のアプリケーション (AUT)	EUT（被試験機器）はソフトウェア・アプリケーション。例：業務アプリケーションのテスト。
テスト容易性設計	EUT（被試験機器）が、テストを可能にする機能を備えて設計される。
デプロイメント	指定された環境に、指定されたバージョンのソフトウェアをインストールすること（例：新しいビルドを本番環境に昇格させること）。
デミングサイクル	W.エドワーズ・デミングが提唱したプロセス管理のための 4 段階のサイクル。PDCA（Plan-Do-Check-Act）とも呼ばれる。
デリバリーケイデンス	デリバリーの頻度。例：1 日あたりのデリバリー回数、1 週間あたりのデリバリー回数など。
デリバリーパッケージ	デプロイメントのためにパッケージ化されたリリースアイテム（ファイル、イメージなど）のセット。

DevOps 用語集

テレメトリー	遠隔地やアクセスできない場所で測定値やその他のデータを収集し、そのデータを受信機器に自動的に送信して監視すること。
トイル	生産サービスの運営に関連する仕事の一種で、手動、反復、自動化可能、戦術的、永続的な価値を持たない傾向がある。
トーマス・キルマンコンフリクトモード	特定のコンフリクト状況下での人の行動選択を測定する。
トライブ	長期的なミッションを持つスクワッドの集まりで、関連するビジネス能力に基づいて活動する。
トライブリード	すべてのスクワッドの技術分野における、広く深い技術的専門知識を持ったシニアテクニカルリーダー。Spotify の定義では、共通の機能セット、製品、サービスに共同で取り組む分隊のグループがトライブである。
トラフィック量	サービス（Web サイトや API など）への訪問者が送受信するデータ量。
トランク	ソフトウェア製品の主要なソースコード統合リポジトリ。
トレース	与えられたリクエストを処理する各機能やマイクロサービスを追跡し、デプロイされたアプリケーションのパフォーマンスと健全性を把握すること。
トロイの木馬	オンラインゲームをプレイするなど、目的の操作を装って悪意ある操作を行うマルウェアのこと。トロイの木馬がウイルスと異なるのは、トロイの木馬が画像ファイルや音声ファイルなどの非実行ファイルに結合するのに対し、ウイルスは動作に実行ファイルを必要とする点。
ナレッジマネジメント	情報に基づいた意思決定を可能にするために、適切な情報が適切な場所や人に適切なタイミングで提供されることを保証するプロセス。
ニューロサイエンス	脳と神経系の研究。
にんじん（Carrots）	望ましい行動を奨励し、報いるためのポジティブなインセンティブ。
ネットワークリライアビリティエンジニア(NRE)	信頼性エンジニアリングの手法を用いて、ネットワークの信頼性を測定し、自動化する。
バージョンコントロールツール	「単一真正のソース」を確保し、すべてのプロダクションアーティファクトの変更管理と追跡を可能とする。
バースティング	パブリッククラウドのリソースが必要に応じて追加され、プライベートクラウドの総計算能力を一時的に増加させる。
ハードニング	不要なソフトウェアの削除や無効化、OS の既知の良いバージョンへのアップデート、ネットワークレベルのアクセスを必要なものだけに制限、アラートを取得するためのログの設定、適切なアクセス管理の設定、適切なセキュリティツールの導入などにより、サーバーやインフラ環境を保護する。

DevOps 用語集

バグ	ソフトウェアのエラーや欠陥で、予期せぬ事態やシステムを劣化させる状態になること。
バックドア	システムへのアクセスに使用される通常の認証をバイパスするもの。バックドアの目的は、組織がシステムへの攻撃に使用された初期の脆弱性を修復したとしても、サイバー犯罪者に将来的なシステムへのアクセスを許可すること。
バックログ	システムに対する要求で、通常は「ユーザーストーリー」の形式を取りプロダクトバックログ項目の優先順位付けされたリストとして表現される。プロダクトバックログは、プロダクトオーナーによって優先順位付けされ、機能的、非機能的、および技術的なチームで生成された要件を含む必要がる。
パッケージレジストリ	ソフトウェアパッケージ、アーティファクト、およびそれらに対応するメタデータのためのリポジトリ。組織が独自に作成したファイルや、サードパーティのバイナリを保存することができる。Artifactory や Nexus などが代表的。
パッチ	バグや弱点を修正するために行われるソフトウェアのアップデート。
バッチサイズ	1 回のコードリリースに含まれる機能の量を指す。
パッチマネジメント	パッチを特定して実施するプロセス。
パフォーマンステスト	EUT（被試験機器）がそのシステムの性能基準を満たしているかどうかを判断すること、あるいはシステムの性能能力がどの程度であるかを判断すること。
バリューストリーム	顧客の要望から製品やサービスが提供されるまでのすべての活動。
バリューストリームマッピング	時間や品質などの無駄を定量化することに重点を置いて、機能サイロを越えた情報、材料、作業の流れを描いたリーンツール。
バリューストリームマップ	最初のリクエストから顧客の価値創造に至るまでの活動のエンドツーエンドの流れを視覚的に表現する。
バリューストリームマネジメント	DevOps のライフサイクルを通じた価値提供の流れを可視化する機能。ツール例：Gitlab CI（Cloud Bees）、Jenkins 拡張機能である DevOptics。
ハンドオフ	特定のタスクの責任を、ある個人またはチームから別の個人に移譲するための手順。
ビジネスケース	予想される商業的利益に基づき、提案されたプロジェクトまたは事業の正当性を証明するもの。
ビジネストランスフォーメーション	ビジネスのあり方を変えること。これを現実のものとするには、組織のミッションを実現するために全員がよりよく連携するために、文化、プロセス、テクノロジーを変えることが必要。
ビジネスバリュー	主要なビジネス KPI へのアプローチがもたらすメリット。

ファジング	アプリケーションに無効な、あるいは予期しない、あるいはランダムなデータを入力する自動ソフトウェアテストの手法。
フィーチャートグル	ソフトウェアのスイッチを使用して機能を隠したり有効にしたりすること。これにより、継続的なインテグレーションが可能となり、特定のステークホルダーと一緒に機能をテストできる。
フェデレーティッド ID	広範囲のアプリケーション、システム、サービスへのアクセスに使用される集中形式の ID。Identity-as-a-Service（IDaas）と呼ばれることもある。特に SAML または OAuth 認証メカニズムを介して、複数のサイトで再利用可能な ID。
フォールスネガティブ	偽陰性。EUT（被試験機器）が実際にはテストの目的に対して合格であるにも関わらず、誤って「不合格」と報告されること。
フォールスポジティブ	偽陽性。EUT（被試験機器）が実際にはテストの目的に対して不合格であるにも関わらず、誤って「合格」と報告されること。
フューチャーステートマップ	バリューストリームマップの一種で、目標とする最終状態がどのようなものであるか、必要な変更にどのように取り組むかを策定し、伝えることができる。
プラグイン	オーケストレーションツールと他のツールとの間であらかじめプログラムされた統合。例えば、多くのツールが Jenkins と統合するためのプラグインを提供している。
プラクティス	サービス提供の特定の側面（変更、インシデント、サービスレベルなど）を管理するための完全なエンドツーエンドの機能。
プラクティス/マイクロプロセスプランニング	プラクティスやマイクロプロセスの目標、目的、インプット、アウトカム、アクティビティ、ステークホルダー、ツールなどを定義するハイレベルなイベント。このミーティングにはタイムボックスはない。
プラクティスバックログ	現在の要件と将来の要件を含め、業務上設計または改善する必要があるすべてのものを優先的にリストアップしたもの。
ブラックボックス	テストケースに EUT（被試験機器） の外部から観察可能な動作の知識のみを使用する。
プリフライト	トランク・ブランチに統合される前の EUT（被試験機器）に実施される活動やプロセスの名称を指す用語の総称。
ブルーグリーンテスト（またはデプロイメント）	ブルーとグリーンという 2 つの環境を使って、ソフトウェアをテストの最終段階から本番に移行させる。ソフトウェアがグリーンの環境で動作するようになったら、ルーターを切り替えて、すべての着信要求がグリーンの環境に行くようにする（ブルーの環境はアイドル状態）。
フレームワーク	ツールをプラグインするためのバックボーン。自動化されたタスクを起動したり、自動化されたタスクの結果を収集したりする。

DevOps 用語集

フロー	人、製品、情報がプロセスの中でどのように移動するかを意味する。フローは、「3 つの道」の「第 1 の道」。
プログラミングベーステスト	テストケースプログラミング言語でコードを書いて作成するテスト。例：JavaScript、Python、TCL、Ruby
プロセス	特定の目的を達成するために設計された、構造化された一連の活動のこと。プロセスは、入力を受け取り、それらを定義された出力に変える。特定のインプットを取り、顧客にとって価値のある特定のアウトプットを生み出す、関連する作業活動。
プロセスオーナー	プロセスの全体的な品質に責任を持つ役割。プロセスマネージャの役割を担う人と同じ人に割り当てられることもあるが、大規模な組織ではこの 2 つの役割は別々になることもある。(ITIL の定義)
プロセススタンドアップ	スプリント目標に向けた進捗状況を確認し、阻害要因を可能な限り迅速に特定するための 15 分間のタイムボックスイベントです。
プロセスの変更	ソフトウェア開発プラクティス、ITIL プロセス、変更管理、承認など、標準的な IT プロセスの変更に焦点を当てる。
プロダクトオーナー	製品の価値を最大化し、製品バックログを管理する責任者。バックログに優先順位をつけ、手入れをし、所有する。スクワッドに目的を与える。
プロダクトバックログ	システムに対する機能的および非機能的な要求事項を優先的にまとめたもので、通常はユーザーストーリーとして表現される。
ベイトソンのステークホルダーマップ	進行中のイニシアティブに対するステークホルダーのエンゲージメントをマッピングするためのツール。
ペネトレーションテスト	セキュリティ上の弱点を探し、システムの機能やデータにアクセスできる可能性のある、コンピュータシステムに対する許可された模擬的な攻撃のこと。
ヘリテージリライアビリティエンジニア (HRE)	レガシーアプリケーションや環境に SRE の原則と実践を適用する。
ベロシティ	あらかじめ定義された間隔で行われた仕事の量の測定。個人やチームが所定の時間内に完了できる仕事の量。
ボイスオブカスタマー (VOC)	顧客の要求やフィードバックを把握・分析し、顧客が何を求めているかを理解するプロセス。
ポリシー	組織が業務の一環として行ってよいこと、または行ってはならないことの観点から境界を定義する正式な文書。

ホワイトボックステスト（またはクリアテスト、グラステスト、トランスペアレントボックステスト、構造的テスト）	アプリケーションの機能ではなく、内部の設計構造や仕組みに関する広範な知識を使用する。
ホワイトリスティング	コンピュータシステム上に存在し、アクティブであることが許可されている承認済みソフトウェアアプリケーションのインデックスを指定する手法である。
マージ	ソフトウェアの変更点をまとめてソフトウェアのバージョン管理システムに統合すること。
マイクロサービス	API を介して相互に作用する小さなモジュールで構成され、システム全体に影響を与えることなく更新可能なソフトウェア・アーキテクチャ。
マイクロプロセス	独立して定義、設計、実施、管理することができる明確な活動で、一般的には主要なサービスマネジメントの実践に関連付けられている。マイクロプロセスは、他のサービスマネジメント手法と統合されることもある。
マイクロプロセスアーキテクチャー	エンド・ツー・エンドのサービスマネジメントを成功させるために必要なすべての活動を統合的に実行する、統合されたマイクロプロセスの集合体。
マインドセット	人の普段の態度や精神状態。
マルウェア	通常は第三者の利益のために、ユーザーの許可なくコンピュータシステムにアクセスするために設計されたプログラム。
マルチクラウド	開発環境やテスト環境へのオンデマンドのマルチテナントアクセスを提供する。
ムダ（リーン生産方式）	プロセス、製品、サービスに付加価値を与えないあらゆる活動。
メトリックス	プロセス、IT サービス、アクティビティを管理するために測定、報告されるもの。
メンタルモデル	現実の世界で何かがどのように機能するかについて、誰かの思考プロセスを説明したもの。
モックオブジェクト	実際のメソッド/オブジェクトの動作を制御してシミュレートするメソッド/オブジェクト。モックオブジェクトはユニットテストで使用され、多くの場合、テスト対象のメソッドはその中で他の外部サービスやメソッドを呼び出す。これらを依存関係と呼ぶ。
モデル	システム、プロセス、IT サービス、CI などを表現したもので、将来の動作を理解または予測するために使用される。プロセスの場合、モデルは特定のタイプのトランザクションを処理するための事前定義されたステップを表す。

モデルベーステスト	テスト対象物のモデルからテストケースを自動的に導き出す。ツール例：Tricentus
モニタリング	コンピュータ・サービスのシステム・リソースやパフォーマンスを監視するためのハードウェアまたはソフトウェア・コンポーネントの使用。
モニタリングツール	IT 組織が特定のリリースの特定の問題を特定し、エンドユーザーへの影響を理解するためのツール。
モノシリック	ソフトウェアシステムは、機能的に区別できる部分（例えば、データの入出力、データ処理、エラー処理、ユーザーインターフェースなど）が、アーキテクチャ的に独立したコンポーネントではなく、すべて織り込まれているモノリシックなアーキテクチャを持つ場合、「モノリシック」と呼ばれる。
ユーザー	IT サービスの消費者。または、認証時に主張されるアイデンティティ（別名：ユーザー名）。
ユーザー/エンティティ行動分析 (UEBA)	正常なユーザーの行動と「異常な」ユーザーの行動を分析し、後者を防止することを目的とした機械学習の手法。
ユーザーストーリー	ユーザーの視点から要件を説明するための簡潔な文章。ユーザーストーリーは、利害関係者とアジャイルサービスマネジメントチームの間のコミュニケーション、計画、交渉活動を促進するために使用される。
ユーザビリティテスト	人間が EUT（被試験機器）を使用する際に満足のいく体験ができるかどうかを判断する。
ユニットテスト	コードのロジックを検証する。
ライセンススキャニング	依存関係にあるライセンスがアプリケーションと互換性があるかどうかをチェックし、それらを承認またはブラックリストに登録する。ツール例：Blackduck や Synopsis など
ラルー（Laloux）（カルチャーモデル）	フレデリック・ラルーは、組織文化を理解するためのモデルを作成した。
ランサムウェア	ユーザーのデバイスまたはネットワークのストレージデバイス上のファイルを暗号化します。暗号化されたファイルへのアクセスを回復するには、ユーザーはサイバー犯罪者に「身代金」を支払わなければならない。通常、ビットコインなどの追跡が困難な電子決済手段を使用する。
ランタイムアプリケーション自己保護 (RASP)	本番環境の脅威が脆弱性を悪用する前に、積極的に監視してブロックするツール。Runtime Application Self Protection。
ランブック	サービスを円滑に運用するために必要な手順をまとめたもの。以前は手動で行われていたが、現在は Ansible のようなツールで自動化されていることが多い。
リーン	ムダを省き、工程の流れを改善することで、顧客価値全体を向上させることに主眼を置いた生産思想。

DevOps 用語集

リーン（形容詞）	余分な、経済的な。豊かさ、潤いがないこと。
リーン IT	IT 製品やサービスの開発・管理に、リーン生産方式の主要な考え方を適用する。
リーンエンタープライズ	リーン生産方式を支える重要なアイデアを戦略的に企業全体に適用する組織。
リーンキャンバス	1 ページのビジネスプランテンプレート。
リーンスタートアップ	失敗のリスクを減らすために、最も効率的な方法でビジネスや製品を開発するためのシステム。
リーンプロダクト開発	リーンの原則を活用して製品開発の課題を解決する。
リーン生産方式	主にトヨタ生産方式から派生したリーン生産方式の思想。
リスク	危害や損失をもたらしたり、組織の目的達成能力に影響を与えたりする可能性のある出来事。リスクの管理は、リスクの特定、リスクの分析、リスクの管理という 3 つの活動からなる。将来の損失の確率的な頻度と確率的な大きさのこと。危害や損失を引き起こしたり、組織の目的を実行または達成する能力に影響を与えたりする可能性のある事象に関連する。
リスクイベント	危害や損失をもたらしたり、組織の目的達成能力に影響を与えたりする可能性のある出来事。リスクの管理は、「リスクの特定」「リスクの分析」「リスクの管理」の 3 つの活動から成り立つ。
リスクマネジメントプロセス	「リスク」がコンテキストに沿って評価され、処理されるプロセスのこと。ISO31000 より：1)コンテキストの確立、2)リスクの評価、3)リスクの処理（再調停、低減、受容）。
リトルの法則	ジョン・リトルが提唱した定理で、静止したシステムにおける顧客の長期平均数 L は、長期平均の有効到着率 λ に顧客がシステム内で過ごす平均時間 W を掛けたものに等しいというもの。
リリース	構築され、テストされ、本番環境に展開されるソフトウェア。
リリースオーケストレーション	典型的なのはデプロイメント・パイプラインで、本番で問題になるような変更を検出するために使用される。他のツールをオーケストレーションすることで、パフォーマンス、セキュリティ、ユーザビリティなどの問題を特定する。Jenkins や Gitlab CI などのツールは、リリースを「オーケストレーション」することができる。
リリースガバナンス	何が変化しているのかを理解したいというビジネスのニーズを満たすために、監査可能で追跡可能な方法でリリースを管理するためのコントロールと自動化（セキュリティ、コンプライアンス、またはその他）に関するもの。
リリースマネジメント	リリースを管理し、Continuous Delivery と Deployment Pipeline を支えるプロセス。

DevOps 用語集

リリース候補	デプロイ用に準備されたリリースパッケージは、「リリース」に合格している場合もあれば、合格していない場合もある。
リリース受け入れ基準	リリースパッケージの測定可能な属性で、リリース候補が顧客への展開に適しているかどうかを判断する。
リワーク	不良を修正するために必要な時間と労力（無駄）。
レイテンシー	メッセージの通信に発生する遅延のことで、サーバーなどが最初のリクエストを受け取ってから、クライアントなどがレスポンスを受け取るまでの「通信上の」時間である。
レジリエンス	変化やインシデントに寛容な環境や組織を構築すること。
レスポンスタイム	レスポンスタイムとは、ユーザーがリクエストをしてからレスポンスを受け取るまでにかかる時間の合計。
レメディエーション	DevOps プロセス中に発見された問題を解決するためのアクション。例：継続的テスト中に、テストケースで失敗判定した EUT（被試験機器）の変更に対するロールバック。
ロールバック	統合が完了したソフトウェアの変更は、統合から削除される。
ロールベースアクセス制御 (RBAC)	システムへのアクセスを許可されたユーザーに制限するためのアプローチ。
ロギング	プロセスコール、イベント、ユーザーデータ、レスポンス、エラー、ステータスコードなど、システムパフォーマンスに関連するすべてのログを取得、集約、保存すること。Logstash や Nagios が代表的な例。
ログ	テストアクティビティや EUT（被試験機器）コンソールログなどの詳細をシリアル化したレポート。
ログマネジメント	情報システム内で作成される大量のログデータの生成、送信、分析、保管、アーカイブ、最終的な廃棄を管理、促進するために使用されるプロセスとポリシーの集合体。
ロングライフテスト	システム全体が長期間にわたって期待通りの性能を発揮するかどうかを判断するテスト。
ワークアラウンド	インシデントや問題の影響を軽減または排除するための一時的な方法。既知のエラーデータベースに既知のエラーとして記録されることがある。(ITIL の定義)。
ワーム（コンピュータ）	ワームは、さまざまなファイルに付着し、コンピュータ間の経路を探すことによって、システム上で自己複製を行う。通常、ネットワークの速度を低下させ、単独で実行することができる（ウイルスは実行するためにホスト・プログラムが必要）。

DevOps 用語集

ワールドカフェ	知識共有のための構造化された会話プロセスで、グループの人々が複数のテーブルで 1 つのトピックについて議論し、個人は定期的にテーブルを交換し、「テーブルホスト」によって新しいテーブルでの前回の議論を紹介してもらう。
安定性	サービスが変化を受け入れる感受性と、システム変更によって引き起こされる可能性のある負の影響。サービスには、長期間にわたって機能するという信頼性はあるが、変更が容易ではないため、安定性はない。
依存関係スキャニング	アプリケーションの開発やテストを行う際、依存関係にあるセキュリティ脆弱性を自動的に発見するために使用する。よく知られているツールとして、Synopisis、Gemnasium、Retire.js、bundler-audit などがある。
依存関係ファイアウォール	多くのプロジェクトでは、未知の、あるいは検証されていないプロバイダから提供されている可能性のあるパッケージに依存しており、潜在的なセキュリティの脆弱性をもたらしている。依存関係をスキャンするツールはあるが、それはダウンロードされた後のこと使用される。これらのツールは、そのような脆弱性が最初からダウンロードされないようにする。
依存関係プロキシ	多くの組織にとって、頻繁に使用されるアップストリームイメージ／パッケージのためのローカルプロキシがあることは望ましい。CI/CD の場合、プロキシはリクエストを受け取り、レジストリからアップストリームイメージを返す役割を担い、プルスルーキャッシュとして機能する。
運用 (Ops)	品質保証アナリスト、リリースマネージャー、システムおよびネットワーク管理者、情報セキュリティオフィサー、IT オペレーションスペシャリスト、サービスデスクアナリストなど、システムやサービスを展開・管理するために必要な日々の運用活動に携わる人。
価値の流れ	エンドツーエンドのバリューストリームを示すマップの一形態。このビューは通常、企業内では利用できない。
価値効率	最小限の時間と資源で価値を生み出せること。
可観測性	サービス全体に関するできるだけ多くのデータを外部に出し、そのサービスの現在の状態を推測することに焦点を当てる。
可変速な IT	従来のプロセスとデジタルプロセスが組織内で共存し、それぞれのスピードで動くようなアプローチ。
可用性	システムが機能している状態で、（ユーザが）使用可能な時間の割合。
課題管理	ソフトウェア開発のライフサイクルを通して、バグや問題を捉え、追跡し、解決するためのプロセス。
回帰テスト	以前は機能していたものを壊していないかどうかを判断する。

DevOps 用語集

改善のカタ	継続的な学習と改善の文化を作るための構造的な方法。（日本のビジネスでは、「型」とは、物事を「正しい」方法で行うという考え方。組織の文化は、一貫したロールモデリング、ティーチング、コーチングを通じて、その「型」として特徴づけることができる）
開発テスト	開発者のテスト環境がプロダクションのテスト環境を適切に表現していることを確認するテスト。
開発者 (Dev)	EUT（被試験機器）の変更を開発する責任を有する個人。あるいはアプリケーションエンジニアやソフトウェアエンジニアなど、ソフトウェア開発活動に携わる個人。
外部自動化	トイルの削減を目的としたサービス以外のスクリプトや自動化。
完了の定義	生産ラインへのリリースが可能となるために、インクリメントが満たさなければならない期待値についての共通理解。(Scrum.org)
官僚主義的文化	標準的なチャネルや手順を使用する傾向があるが、危機的状況ではそれでは不十分な場合がある(Westrum)。
監査マネジメント	製品やサービスの監査性を確保するための自動化されたツールを使用すること。これには、ビルド、テスト、デプロイ活動の監査ログの保存、構成やユーザーの監査、本番操作のログファイルなどが含まれる。
基本セキュリティハイジーン	すべての環境に例外なく適用されなければならない、最低限のセキュリティ対策の共通セット。具体的には、基本的なネットワークセキュリティ（ファイアウォールと監視）、ハードニング、脆弱性とパッチの管理、ログの記録と監視、基本的なポリシーと実施（「コードとしてのポリシー」というアプローチで実施される場合もある）、アイデンティティとアクセスの管理などが含まれる。
既知のエラー	文書化された根本原因と回避策がある問題。(ITIL 定義)
機械学習	データから学習するアルゴリズムを用いたデータ分析。
機能テスト	サービスの機能動作が期待通りであるかどうかを判断するためのテスト。
機密管理	アプリケーション、サービス、特権アカウント、および IT エコシステムのその他の機密部分で使用するパスワード、鍵、API、トークンなどのデジタル認証資格（秘密）を管理するためのツールと方法を指します。
機密情報の検出	パスワード、認証トークン、秘密鍵などの機密情報が、意図せずにリポジトリのコンテンツの一部として流出することを防ぐことを目的とする。
規格適合性テスト	EUT（被試験機器）が規格に準拠しているかどうかを判断するテスト。
技術経済的なパラダイムシフト	カルロタ・ペレスが考案した、イノベーションに基づく経済・社会発展の一般的な理論の中核をなすもの。

協調と競争	文化的な価値観の変化として、社内での競争や分裂を避け、協調性・協力性を高めることが重要である。
脅威インテリジェンス	脅威の性質または脅威が実行することが知られている行為に関連する情報。また、ある脅威の行動に関連する「侵害の指標」や、ある脅威の行動を修正する方法を記述した「行動指針」を含むこともある。
脅威エージェント	人間または自動化された行為者で、システムに危害を加える、またはシステムを危険にさらすことを意図して、そのシステムに対して行動する者。「脅威行為者」とも呼ばれることがある。
脅威モデリング	潜在的な脅威をランク付けし、モデル化することで、リスクが関連するアプリケーションの価値との関連で理解し、軽減できるようにする手法。
脅威検知	攻撃を検知し、報告し、対応する能力を支援する能力を指す。侵入検知システム、DoS（サービス妨害）システムにより、ある程度の脅威の検知と防止が可能。
継続的インテグレーション（CI）	開発者は少なくとも毎日、自分のコードをトランクやマスターにマージし、コードをコミットするたびにテスト（ユニットテスト、インテグレーションテスト、受け入れテスト）を実施することが求められる開発手法。
継続的インテグレーション・ツール	定期的にコードのマージ、ビルド、テストを行うことで、即時のフィードバックループを実現するツール。例として以下のようなものがある。Atlassian Bamboo、Jenkins、Microsoft VSTS/Azure DevOps、TeamCity
継続的サービス改善（CSI）	ITIL コア・パブリケーションの一つで、サービス・ライフサイクルのステージの一つ。Continual Service Improvement。
継続的テスト（CT）	これは、DevOps 環境における EUT（被試験機器）のテストと検証に関連する用語の総称。
継続的デプロイメント	自動化されたテストに合格したすべての変更を、自動的にプロダクション環境にデプロイできるようにするための一連の手法。
継続的デリバリー（CD）	ソフトウェアのライフサイクルを通じて、常にリリース可能な状態にしておくことに重点を置いた手法。
継続的デリバリーパイプライン	製品の変更に対して段階的に実行される一連のプロセス。パイプラインの最初に変更が投入される。変更とは、アプリケーションの新しいバージョンのコード、データ、イメージの新バージョンである。各ステージでは、前のステージで生じた成果物を処理し、最後のステージでは、プロダクション環境へのデプロイメントが行われる。
継続的デリバリーパイプライン・アーキテクト	継続的デリバリーパイプラインの実装とベストプラクティスを導く責任者。

DevOps 用語集

継続的デリバリーパイプライン・ステージ	継続的デリバリーパイプラインにおける各プロセス。これらは標準的なものではない。例えば、設計：実装の変更を決定する、作成：設計の変更が統合されていないバージョンを実装する、統合：統合する。
継続的フロー	プロセスの最初のステップから最後のステップまで、ステップ間のバッファを最小限（またはゼロ）にして、人や製品をスムーズに移動させること。
継続的モニタリング(CM)	試験結果情報の記録、通知、警告、表示、分析に関連する用語の総称。
継続的改善	デミングの「Plan-Do-Check-Act」をベースにした、製品・プロセス・サービスの改善を継続的に行うためのモデル。
欠陥密度	ユニットで発見された欠陥の数 例：欠陥数/KLOC、欠陥数/変更数。
堅牢（DevOps）	継続的なデリバリーパイプラインのできるだけ早い段階でセキュリティ対策を盛り込み、DevOps の実践だけでは得られないサイバーセキュリティ、リリースのスピード、品質を向上させる手法。
堅牢な開発(DevOps)	継続的なデリバリーパイプラインのできるだけ早い段階でセキュリティ対策を盛り込み、DevOps のプラクティスだけで得られる以上のサイバーセキュリティ、リリースのスピード、品質を実現する手法。
故障率	単位時間当たりの失敗判定数
互換性テスト	ピア・ツー・ピアのアプリケーションやプロトコルなど、EUT（被試験機器）が他の EUT（被試験機器）相互運用できるかどうかを判断するテスト。
攻撃パス	攻撃者の目的を達成するために、脅威が悪用する可能性のある弱点の連鎖のこと。例えば、攻撃経路は、ユーザーの認証情報の漏洩から始まり、脆弱なシステムで特権を昇格させ、保護されたデータベースにアクセスし、その情報を攻撃者自身のサーバーにコピーするために使用される。
構成管理	製品の性能、機能、物理的属性と、要求、設計、運用情報との一貫性を、プロダクトライフサイクルを通して確立し、維持するためのシステムエンジニアリングプロセス。Configuration Management（CM）。
構造改革	権限の階層、目標、構造的特徴、管理手順、管理システムの変更。
根本原因分析	問題やインシデントの根本的な原因を特定するために取る行動。Root Cause Analysis (RCA)。
仕掛り作業(WIP)	着手したが完了していない作業。
指標	製品やインフラの健全性を監視するために使用される測定に関連する用語の総称。
事業継続	災害や予期せぬ事故によって重要なサービスが停止した場合でも、業務や中核的なビジネス機能に深刻な影響が及ばないようにする組織の能力。

DevOps 用語集

次世代育成能力	長期的な成果を第一に考える文化的な考え方で、組織がその成果を達成するための投資や協力を推進する。
自動 DevOps	Auto DevOps は、ソフトウェア開発ライフサイクルを自動的に構成することで、DevOps のベストプラクティスをプロジェクトにもたらす。アプリケーションの検出、ビルド、テスト、デプロイ、監視を自動的に行う。
自動スケーリング	コストをコントロールしながら、トラフィックや容量の変化に応じて、インフラを自動的かつ弾力的に拡大・縮小できること。
自動ロールバック	デプロイ中に障害が検出された場合、オペレーター（または自動化されたプロセス）が障害を確認し、障害のあるリリースを以前の既知の動作状態にロールバックする。
自由と責任	DevOps のような自己管理の自由には、勤勉であること、アドバイスのプロセスに従うこと、成功と失敗の両方にオーナーシップを持つことなどの責任が伴うという文化的な中核的価値観。
邪悪な質問	邪悪な質問は、私たちの行動や選択を形作る仮定を明らかにするために使われる。ある問題、課題、コンテキストについて私たちが抱いている、埋め込まれた、そしてしばしば矛盾した仮定を明確にする質問である。
弱点	攻撃者がアプリケーションやシステム、またはそこに含まれるデータを危険にさらすために利用できるソフトウェアのエラー。脆弱性とも呼ばれる。
主要指標	プロセス、IT サービス、アクティビティを管理するために測定、報告されるもの。
主要成功要因 (CSF)	IT サービス、プロセス、計画、プロジェクト、その他の活動が成功するために起こるべきこと。Critical Success Factor。
受け入れテスト駆動開発	開発を始める前に、チーム全体で受け入れ基準を例を挙げて共同で議論し、具体的な受け入れテストに落とし込んでいく手法。ATDD（Acceptance Test Driven Development）。
修復計画	失敗した変更やリリースの後に取るべき行動を決定する計画。(ITIL 定義)
従業員ネットプロモータースコア（eNPS）	組織が従業員のロイヤリティを測定するための方法。ネットプロモータースコアは、元々は顧客サービスのツールで、後に顧客ではなく従業員を対象に使われるようになった。
処理時間	製造または開発の手順によって、1 つまたは複数の入力が完成品に変換される期間のこと。(ビジネス用語辞典)

助言プロセス	決定を下す者は、その決定によって重要な影響を受けるすべての人と、その問題について専門知識を持つ人に助言を求めなければならない。受け取った助言は考慮されなければならないが、必ずしも受け入れたり従ったりする必要はない。助言プロセスの目的は、コンセンサスを形成することではなく、意思決定者が可能な限り最善の意思決定を行えるように情報を提供することであり、助言プロセスに従わないと、信頼が損なわれ、ビジネスに不必要なリスクが生じる。
信頼感の高い文化	信頼感の高い文化を持つ組織は、良好な情報の流れ、部門を超えた協力関係、責任の共有、失敗からの学習、新しいアイデアを奨励します。
信頼性	サービス、コンポーネント、または CI が、合意された機能を中断することなく実行できる時間を示す尺度。通常、MTBF または MTBSI として測定される。(ITIL 定義)
信頼性テスト	このテストの目的は、長期間にわたってストレスや負荷がかかる条件下で、完全なシステムが期待通りに動作するかどうかを判断する。
心理的安全性	チームが対人的なリスクを取るのに安全であるという信念を共有すること。
振る舞い駆動開発 (BDD)	テストケースは、EUT（被試験機器）の外部から観察可能な入力と出力をシミュレートすることで作成される。Behavior Driven Development (BDD)。ツール例：Cucumber。
神経可塑性	特に学習や経験、あるいは傷害に対応して、シナプス結合を形成し、再編成する脳の能力を指す。
人の変化	従業員の態度、行動、スキル、またはパフォーマンスを変えることに焦点を当てる。
垂直スケーリング	例えば、より高速なコンピュータを使用してより多くのタスクをより速く実行するなど、処理速度を向上させるためにコンピューティングリソースの規模を拡大する。
水平スケーリング	処理量を増やすために計算機資源の規模を大きくする。例：コンピュータの台数を増やし、より多くのタスクを並行して実行する。
制約	制限や制約、束縛するもの。「ボトルネック」も参照。
制約条件理論	目標達成を阻む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計画的に改善していく方法論。
静的アプリケーション・セキュリティ・テスト (SAST)	ソースコードにバグや弱点がないかをチェックするテストの一種。
静的コード解析	メモリリーク、使用されていない変数、使用されていないポインターなど、ソースコードの論理的なエラーや抜けを検出すること。
脆弱性	攻撃者が悪用できる設計、システム、またはアプリケーションの弱点。

脆弱性インテリジェンス	既知の脆弱性に関する情報（影響を受けるソフトウェアのバージョン、脆弱性の相対的な重要度：例えば、ユーザー・ロールの特権の昇格につながるか、サービス拒否を引き起こすか）、脆弱性の悪用可能性（悪用の容易さ/難しさ）、場合によっては現在の悪用率（活発に悪用されているか、理論上だけか等）を記載。また、この情報には、記述された脆弱性を修正したことが確認されているソフトウェアのバージョンに関するガイダンスも含まれることが多い。
脆弱性マネジメント	脆弱性を特定し、改善するプロセス。
設計原則	DevOps のデリバリー運用モデルを設計、組織、管理するための原則。
組織モデル	DevOps では、Spotify の Squad という IT 組織をモデルにしたアプローチをとる。
組織的変化	組織内の人間の行動を、新しい構造やプロセス、要求に適合させるための努力。
組織文化	組織のメンバーを束ねる、共有された価値観、前提、信念、および規範のシステム。
多要素人情	認証に 2 つ以上の要素を使用すること。2 要素認証と同義で使われることが多い。
多要素認証	認証に少なくとも 2 つの要素を使用すること。2 つの要素は同じクラスのものでもよい。
妥当性	継続的テストでは、最も重要なテストとテスト結果に焦点を当てることを重視する。
待ち時間	仕事を待つための無駄な時間（例：開発・テストのインフラ待ち、リソース待ち、経営陣の承認待ち）。
脱予算経営	コマンド＆コントロールを超えて、よりエンパワーされた適応的な状態を目指すマネジメントモデル。
単一障害点(SPOF)	SPOF（Single Point of Failure）。システムの一部で、故障するとシステム全体が機能しなくなるもの。
統合開発環境 (IDE) 'lint' チェック	リンティングは、コードに潜在的なエラー（フォーマットの不一致、コーディング標準や規約への非準拠、論理的エラーなど）がないかを分析するプログラムを実行すること。
統合開発環境(IDE)	開発者がソフトウェアを書いたりテストしたりするための基本的なツールを統合したソフトウェアスイート。通常、IDE にはコードエディタ、コンパイラまたはインタープリタ、デバッガが含まれており、開発者は単一の GUI（グラフィカル・ユーザー・インターフェース）でアクセスする。IDE は独立したアプリケーションである場合もあれば、1 つ以上の既存の互換性のあるアプリケーションの一部として含まれている場合もある。(TechTarget)

動的アプリケーションセキュリティテスト (DAST)	構築されたコードに対して実行され、公開されたインターフェースに対するテスト一種。Dynamic Application Security Testing。
動的解析	オフラインでコードの検証を繰り返すのではなく、アプリケーションが稼働している間に不具合を検出することを目的として、リアルタイムにデータを実行してテストすること。
特権アクセスマネジメント (PAM)	企業が重要な資産への安全な特権的アクセスを提供し、特権的なアカウントとアクセスを保護、管理、監視することで、コンプライアンス要件を満たすことを支援する技術(Gartner 社)。Privileged Access Management。
内部自動化	負担軽減を目的としたサービスの一環として提供されるスクリプトやオートメーション。
認可	リソースへのアクセス権を持つユーザーに役割を付与するプロセス。
認証	主張された身元を確認するプロセス。認証は、知っていること（パスワードや PIN など）、持っているもの（トークンやワンタイムコード）、本人であること（バイオメトリクス）、またはコンテキスト情報などに基づいて行われる。
認知バイアス	認知バイアスとは、人間の脳が個人的な経験や好みのフィルターを通して情報を認識する傾向があるために起こる、客観的な思考の制限であり、判断において規範や合理性から逸脱する体系的なパターン。
爆発半径	サービスインシデントのインパクト分析に使用。特定の IT サービスに障害が発生した場合、影響を受けるユーザー、顧客、他の依存するサービスなど。
発生的文化 (DevOps)	発生的文化の組織では、ミッションとの同一化を通じて連携が行われる。個人は、自分がすべきこと、そしてその結果への影響に「納得」する。発生的文化の組織は、必要な情報を必要な人に積極的に届ける傾向がある。(Westrum)
判定	テスト結果は不合格、合格、判定不能のいずれかに分類される。
避難訓練	ライブサービスの運用に焦点を当てた計画的な障害テストプロセス。サービスの障害テストのほか、コミュニケーション、ドキュメンテーション、その他のヒューマンファクタのテストを含む。
非機能テスト	ソフトウェアサービスの性能、使い勝手、信頼性などの非機能面をチェックすることを目的としたサービステストの一種。
非機能要求	特定の動作や機能（可用性、信頼性、保守性、サポート性など）ではなく、システムの動作を判断するための基準を規定した要求事項、システムの品質。
非難なしのポストモーテム	サービスインシデントの原因となった行為を行ったエンジニアが、罰や報復を恐れずに自分の行った行為について詳細な説明を行うプロセス。

標準的な変更	事前に承認された、手順や作業指示に従った低リスクの変更。(ITIL の定義)
病理学的文化	情報を個人的なリソースとして捉え、政治的な権力闘争に利用する傾向がある（Westrum）。
品実マネジメント	テストケースの計画、テストの実行、欠陥の追跡（多くはバックログへ）、重要度と優先度の分析を扱うツール。CA の Agile Central など
頻度	アプリケーションがリリースされる頻度。
付加価値時間	価値を生み出す活動（開発、テストなど）に費やされた時間。
分散バージョン管理システム(DVCS)	ソフトウェアのリビジョンが格納される。分散型リビジョン管理システム（DRCS）とも呼ばれる。
文化 (組織文化)	組織に固有な心理的・社会的環境に貢献する価値観や行動様式。
変化による疲労	個人やチームが組織の変化に対して無関心であったり、消極的な諦めの気持ちを持っていること。
変革型リーダーシップ	リーダーがフォロワーの価値観や目的意識に訴えることで、フォロワーがより高いパフォーマンスを発揮するように鼓舞し、動機づけを行い、大規模な組織変革を促進するリーダーシップモデル（State of DevOps Report, 2017）。
変更	IT サービスに影響を与える可能性のあるものを追加、変更、削除すること。(ITIL®定義)
変更に伴うテスト選定方法	テストの属性と、ビルドで変更されるコードの属性を照合する基準に従って選択されます。
変更リードタイム	変更要求から変更の提供までの時間を示す指標。
変更管理	ライフサイクルを通してすべての変更をコントロールするプロセス。(ITIL®定義)
変更管理 (組織的)	個人、チーム、組織を現在の状態から望ましい未来の状態へと移行させるためのアプローチ。求められるビジネス成果を達成するために、変化の人間側を管理するプロセス、ツール、テクニックを含む。
変更失敗率	失敗した変更やロールバックされた変更の割合を示す指標。
変更要求 (RFC)	変更を行うための正式な提案。RFC という用語は、変更記録や変更そのものを意味するものとして誤用されることが多い。(ITIL 定義)
法規制対応テスト	試験の目的は、EUT（被試験機器）が特定の規制要件に適合しているかどうかを判断することである。例：EUT（被試験機器）が消費者クレジットカード処理に関する政府の規制を満たしていることを検証する。
問題	1 つまたは複数のインシデントの根本的な原因のこと。(ITIL 定義)

DevOps 用語集

役割	人やチームに与えられる責任、活動、権限のセット。役割はプロセスによって定義される。一人の人間やチームが複数の役割を持つこともある。ユーザーがシステムまたはアプリケーション内でアクションを実行できるように、ユーザーまたはユーザーグループに割り当てられた一連の権限のこと。
優先順位	インパクトと緊急性に基づいた、インシデント、問題、または変更の相対的な重要性のこと。(ITIL の定義)
要求事項マネジメント	要件定義、トレーサビリティ、階層化、依存関係を扱うツール。また、コード要求や要求に対するテストケースを扱うことも多い。
累積フローダイアグラム	アジャイルソフトウェア開発やリーン製品開発で使用されるツール。ある状態での作業量を描いた面積グラフで、到着、待ち行列の時間、待ち行列の量、出発を示す。
論理爆弾 (Slag Code)	ロジックボム。プログラムされた条件が満たされたときに、システムに害を及ぼすために使用される悪意のあるコードの文字列。
冪等性(べきとうせい)	CM ツール（Puppet、Chef、Ansible、Salt など）は、サーバーの望ましい状態をコードや宣言として定義し、定義された状態を時系列で一貫して実現するために必要なステップを自動化することで、「冪等性」を実現していることを訴求している。

SREファウンデーションコース付加価値の高いリソース

このドキュメントは、DevOps InstituteのSite Reliability Engineering (SRE)コースに関連する記事やビデオへのリンクを提供しています。この情報は、SRE関連の概念や用語の理解を深めるために提供されているものであり、試験的なものではありません。もちろん、ウェブ上には他にもビデオ、ブログ、ケーススタディなどが豊富にあります。
追加のご提案もお待ちしております。

講座で紹介された動画

モジュール	タイトルと内容	リンク
1: SREの原則と実践	「DevOpsとSREは何が違うのか？ GoogleのSeth Vargo氏とLiz Fong-Jones氏 (05:10)	https://youtu.be/uTEL8Ff1Zvk
2: サービスレベル目標とエラー予算	GoogleのSeth Vargo氏とLiz Fong-Jones氏による「リスクとエラーの予算」(06:17)	https://youtu.be/y2ILKr8kCJU
3：トイルの削減	GCPのMax Luebbe氏による「Pragmatic Automation」(04:45)	https://www.youtube.com/watch?v=oDcjAcFTFC0&t=0m56s
4: モニタリングとサービスレベル指標	SLI & Reliability Deep-Dive」。マイクロソフトのデビッド・N・ブランク-エデルマン氏と (08:35)	https://www.youtube.com/watch?v=1iMo3SkdQqQ
5: SREツールと自動化	自動化のアイロニー」。マイクロソフトのタナー・ルンド氏との「A Comedy in Three Parts」（18:32）	https://www.youtube.com/watch?v=U3ubcoNzx9k
6: アンチフラジリティと失敗からの学習	Indeed.comのPreetha Appan氏による「Sloth, a Tool for Inducing Network Failures」（04:45）	https://www.usenix.org/conference/srecon17americas/program/presentation/appan
7: SREの組織的影響	UberのRick Boone氏による「A History of SRE at Uber」(06:24)	https://www.youtube.com/watch?v=qJnS-EfIIIE
8: SREとその他のフレームワーク、将来について	DevOps InstituteのJayne Groll氏による「A Look at ITIL4 & SRE」(11:25)	https://dev.tube/video/vFyPXIsUEhE

SREファウンデーションコース付加価値の高いリソース

SREレポート

レポート名	作家・出版社	リンク
2019年 SREレポート	キャッチポイント	http://pages.catchpoint.com/SRE-Report-2019.html
SREとは？	カート・アンダーセン、クレイグ・セベニク、オライリー・メディア	https://www.oreilly.com/library/view/what-is-sre/9781492054429/

SREの記事

記事タイトル・著者名	関連モジュール	リンク
ソフトウェアプロジェクトのメンテナンスコストに影響を与える要因は何か」 by Sayed Mehdi Hejazi Dehaghani and Nafiseh Hajrahimi	1: SREの原則と実践	https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3610582/
サービスレベル目標（SLO）の測定と評価」Serhat Can著	1: SREの原則と実践	https://medium.com/@serhatcan/measuring-and-evaluating-service-level-objectives-slos-84b0dc740a0a
Bloomberg Bets Big on SREs」マイケル・レンベッツィー著	1: SREの原則と実践	https://www.techatbloomberg.com/blog/bloomberg-bets-big-on-sres/
ブルームバーグのサイト・リライアビリティ・エンジニアリング」スティグ・ソレンセン氏	1: SREの原則と実践	https://player.fm/series/devops-chat/site-reliability-engineering-sre-bloomberg-w-stig-Sorenson
サイト・リライアビリティ・エンジニアであることの意味」Molly Struve著	1: SREの原則と実践	https://dev.to/molly_struve/what-it-means-to-be-a-site-reliability-engineer-32ki
誤差予算-実践編」ヤロスラフ・モロチコ著	2: SLO's & Error Budgets	https://www.slideshare.net/yaroslavmolochko/implementing-error-budgets-125400822
リヨン・ウォン氏による「最高のチームでも捕まる5つのSRE導入の罠を回避する方法	2: SLO's & Error Budgets	https://thenewstack.io/how-to-avoid-the-5-sre-implementation-traps-that-catch-even-the-best-teams/

SREファウンデーションコース付加価値の高いリソース

'Site Reliability Engineering:DevOps 2.0」 Saba Anees著	2: SLO's & Error Budgets	https://www.appdynamics.com/blog/engineering/site-reliability-engineering-devops-2-0/
サイト・リライアビリティ・エンジニアリングを始めよう」ジェニファー・ペトフ著	2: サービスレベル目標とエラーバジェット	https://www.devops.talksplus.com/wp-content/themes/dotc/2019_Melbourne/presentations/Getting%20Started%20with%20Site%20Reliability%20Engineering%20(Jennifer%20Petoff%20DOTC%20Deck).pdf
Invent More, Toil Less」 ベッツィー・ベイヤー、ブレンダン・グリーソン、デイブ・オコナー、ヴィヴェック・ラウ著	3：トイルの削減	https://storage.googleapis.com/pub-tools-public-publication-data/pdf/45765.pdf
SREのレッスン」。継続的に最適化して苦労を減らす」デイモン・エドワーズ氏	3：トイルの削減	https://www.rundeck.com/blog/sre-lessons-continuously-optimize-to-reduce-toil
Toil」。誰もが感じたことのある問題にようやく名前がついた」デイモン・エドワーズ著	3：トイルの削減	https://www.rundeck.com/blog/toil-finally-a-name-for-a-problem
SREのレッスン」。継続的に最適化して苦労を減らす」デイモン・エドワーズ氏	3：トイルの削減	https://www.rundeck.com/blog/sre-lessons-continuously-optimize-to-reduce-toil
SREのアンチパターン。"Known workaround.Bug closed." by Damon Edwards	3：トイルの削減	https://www.rundeck.com/blog/sre-anti-pattern-known-workaround-bug-closed
サイト・リライアビリティ・エンジニアリング（SRE）」。A Simple Overview」マック・スローカム著	3：トイルの削減	https://www.oreilly.com/ideas/site-reliability-engineering-sre-a-simple-overview
クレイグ・セベニックとカート・アンダーセンによる「SREとは何か？	3：トイルの削減	https://www.oreilly.com/library/view/what-is-sre/9781492054429/
'Is It Worth the Time?' by Xkcd	3：トイルの削減	https://imgs.xkcd.com/comics/is_it_worth_the_time.png
SREがDevOpsか、DevOpsがSREか」 Julia Lamenza氏	3：トイルの削減	https://speakerdeck.com/devopslx/2020-dot-01-meetup-talk-1-sre-is-devops-or-devops-is-sre-julia-lamenza

SREファウンデーションコース付加価値の高いリソース

An Engineer's Guide To SLA, SLO, and SLI」 Ram Lyengar著	4: モニタリングとサービスレベル指標	https://plumbr.io/blog/monitoring/an-engineers-guide-to-sla-slo-and-sli
サービスレベル指標の実際」 スティーブン・ソーン氏	4: モニタリングとサービスレベル指標	https://medium.com/@jerub/service-level-indicators-in-practice-6a1125e24bee
アンディ・サイクスの「Stop Using Nagios (So It Can Peacefully Die)」。	4: モニタリングとサービスレベル指標	http://www.slideshare.net/superdupersheep/stop-using-nagios-so-it-can-die-peacefully
Why Does (My) Monitoring Suck?" by Todd Palion	4: モニタリングとサービスレベル指標	https://www.usenix.org/conference/srecon19asia/presentation/palino- モニタリング
Observability - A 3-Year Retrospective」 by Charity Majors	4: モニタリングとサービスレベル指標	https://thenewstack.io/observability-a-3-year-retrospective/
ピーター・ウォーターハウス氏による「モニタリングとオブザバビリティ-何が違うのか、なぜそれが重要なのか？	4: モニタリングとサービスレベル指標	https://thenewstack.io/monitoring-and-observability-whats-the-difference-and-why-does-it-matter/
モニタリングでアラートノイズを減らす3つの方法」 （Christina DiSomma氏	4: モニタリングとサービスレベル指標	https://www.metricly.com/3-ways-reduce-alert-noise/
Charity Majorsによる「 Observability and Understanding Operational Ramifications of a System」。	4: モニタリングとサービスレベル指標	https://www.infoq.com/articles/charity-majors-observability-failure/
SLI（Service Level Indicator）ワークショップの実施」BY GDS	4: モニタリングとサービスレベル指標	https://gds-way.cloudapps.digital/standards/slis.html
Googleにおける自動化の進化」 （ニール・マーフィー氏	5: SREツールと自動化	https://landing.google.com/sre/sre-book/chapters/automation-at-google/
様々な人による「労働年金省でのSRE」。	5: SREツールと自動化	https://dwpdigital.blog.gov.uk/category/site-reliability-engineering-sre/
サービスレベル目標（SLO）の測定と評価」Serhat Can著	5: SREツールと自動化	https://www.atlassian.com/blog/opsgenie/measuring-and-evaluating-service-level-objectives
'Best NoSQL Databases 2019'	5: SREツールと自動化	https://www.improgrammer.net/most-popular-nosql-database/

SREファウンデーションコース付加価値の高いリソース

DevOps文化をサポートするオンコールツール」（Dan Holloran氏	5: SREツールと自動化	https://victorops.com/blog/devops-on-call-tools-to-support-culture
Githubによる「A curated list of awesome Site Reliability and Production Engineering resources」。	5: SREツールと自動化	https://github.com/dastergon/awesome-sre
アンシブルによる「セキュリティ＆コンプライアンス	5: SREツールと自動化	https://www.ansible.com/use-cases/security-and-compliance
OWASPによる「Secure Coding Best Practices」。	5: SREツールと自動化	https://www.owasp.org/images/0/08/OWASP_SCP_Quick_Reference_Guide_v2.pdf
Cindy Sridharan氏による「Testing in Production, the safe way」。	5: SREツールと自動化	https://medium.com/@copyconstruct/Testing-in-production-the-safe-way-18ca102d0ef1
'Amazon Andon Cord:その正体と対処法」ヴェレンティン・バヤール著	5: SREツールと自動化	https://blueboard.io/resources/amazon-andon-cord/
GitLabによる「DevOps Tools Landscape」。	5: SREツールと自動化	https://about.gitlab.com/devops-tools/
The Best SRE Tools According to NewRelic by Kevin Casey	5: SREツールと自動化	https://blog.newrelic.com/technology/best-sre-tools/
効率、効果、文化を測定してDevOpsの変革を最適化する」by IT Revolution	6: アンチフラジリティと失敗からの学習	http://devopsenterprise.io/media/DOES_forum_metrics_102015.pdf
マイク・ブリテンの「Tracking Every Release	6: アンチフラジリティと失敗からの学習	https://codeascraft.com/2010/12/08/track-every-release/
A recovery point objective (RPO)」by Margaret Rouse	6: アンチフラジリティと失敗からの学習	https://whatis.techtarget.com/definition/recovery-point-objective-RPO
学習する組織」アンドリュー・シェイファー著	6: アンチフラジリティと失敗からの学習	https://www.slideshare.net/littleidea/the-learning-organization-modev
'The Three Ways:The Principles Underpinning DevOps」（ジーン・キム著	6: アンチフラジリティと失敗からの学習	https://itrevolution.com/the-three-ways-principles-underpinning-devops/
A Typology of Organizational Cultures」R Westrum著	6: アンチフラジリティと失敗からの学習	http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1765804/pdf/v013p0ii22.pdf

SREファウンデーションコース付加価値の高いリソース

「クラウドソリューションを成功させたいですか？失敗を受け入れることから始めよう！」 by Arman Kamran	6: アンチフラジリティと失敗からの学習	https://medium.com/@armankamran/do-you-want-your-cloud-solutions-to-succeed-start-with-embracing-failures-8f5f40b57a64
The Cost of IT Downtime」マイケル・コープランド著	7: SREの組織的影響	https://www.the20.com/blog/the-cost-of-it-downtime/
SREチームはどのように組織され、どのようにスタートするか」マット・ブラウン氏	7: SREの組織的影響	https://cloud.google.com/blog/products/devops-sre/how-sre-teams-are-organized-and-how-to-get-started
SREへの入り方」アリス・ゴールドファス著	8: SREとその他のフレームワーク、将来について	https://blog.alicegoldfuss.com/how-to-get-into-sre/
Kubernetes Up & Running」Brendan Burns、Joe Beda、Kelsey Hightower著	7: SREの組織的影響	https://clouddamcdnprodep.azureedge.net/gdc/gdckTlBtc/original
非難されない死後処理と公正な文化」 by ジョン・オールスポー	7: SREの組織的影響	https://codeascraft.com/2012/05/22/blameless-postmortems/
ノーム・ケルスによる「首相指令	7: SREの組織的影響	https://retrospectivewiki.org/index.php?title=The_Prime_Directive
'Creating Antifragile Systems: 企業のためのサイト・リライアビリティ・エンジニアリング」（コンティノ著	7: SREの組織的影響	https://www.contino.io/files/Enterprise-Site-Reliability-Engineering-Contino.pdf
SRE組織のスケールアップ」。1つのチームから多くのチームへの旅」Gustavo Franco氏	7: SREの組織的影響	https://www.usenix.org/sites/default/files/conference/protected-files/sre19amer_slides_franco.pdf
ジョン・ウィリスの「The Convergence of DevOps」。	8: SREとその他のフレームワーク、将来について	http://itrevolution.com/the-convergence-of-devops/
サイト・リライアビリティ・エンジニア（SRE）の役割と責任」（Dan Holloran氏	8: SREとその他のフレームワーク、将来について	https://victorops.com/blog/site-reliability-engineer-sre-roles-and-responsibilitis
ITIL4とSREがDevOpsとどのように連携するか」ジェイン・グロール著	8: SREとその他のフレームワーク、将来について	https://techbeacon.com/enterprise-it/how-itil4-sre-align-devops
信頼性工学の未来」マイケル・キホー氏	8: SREとその他のフレームワーク、将来について	https://michael-kehoe.io/tags/future-of-sre/

データベース信頼性入門」マッケンジー・クラーク著	8: SREとその他のフレームワーク、将来について	https://softwareengineeringdaily.com/2018/10/16/an-introduction-to-database-reliability/

ウェブサイト

タイトル	リンク
USENIX	https://www.usenix.org/
ハニカム	https://www.honeycomb.io/
Player FM - DevOps Chat	https://player.fm/series/devops-chat
SREウィークリー	https://sreweekly.com/
Netflix	https://github.com/Netflix
ダウンディテクター	https://downdetector.co.uk/

SREブログ

ブログ	リンク
AppDynamicsブログ	https://www.appdynamics.com/blog
アトラシアンブログ	https://www.atlassian.com/blog
プロメテウス・ブログ	https://prometheus.io/blog/
ルンデック・ブログ	https://www.rundeck.com/blog
SREです。カンバン以来の大嘘	https://theagileadmin.com/2018/10/02/sre-the-biggest-lie-since-kanban/
ブルームバーグの技術	https://www.techatbloomberg.com/blog
VictorOpsブログ	https://victorops.com/blog

SREファウンデーションコース付加価値の高いリソース

その他の興味あるビデオ

関連モジュール	タイトル	リンク
2: サービスレベル目標とエラーバジェット	Yoann Fouquet氏による「データ集約型サービスのSLO」(23:47)	https://www.youtube.com/watch?v=ZdguHXglT8M
2: サービスレベル目標とエラーバジェット	Heinrich Hartmann氏による「Latency SLO Done Right」(27:12)	https://www.youtube.com/watch?v=ycsc2kCaJxM&feature=youtu.be
4: モニタリングとサービスレベル指標	Molly Struveとの「Building a Scalable Monitoring System」（26.48）	https://www.youtube.com/watch?v=vI1ecpFohZQ&feature=youtu.be

SREブック

タイトル	著者	リンク
サイトリライアビリティエンジニアリング	ベッツィー・ベイヤー、クリス・ジョーンズ、ジェニファー・ペトフ、ナイアール・リチャード・マーフィー	https://landing.google.com/sre/sre-book/toc/index.html
サイト・リライアビリティ・ワークブック	Betsy Beyer、Niall Richard Murphy、David K. Rensin、Kent Kawahara、Stephen Thorne	https://landing.google.com/sre/workbook/toc/
ソフトウェア工学の事実と誤謬	ロバート・L・グラス	https://www.amazon.com/Facts-Fallacies-Software-Engineering-Robert/dp/0321117425
カオスエンジニアリング	アリ・バシリ、ノラ・ジョーンズ、アーロン・ブローウィアック、ロリン・ホフスタイン、ケイシー・ローゼンタール	https://www.oreilly.com/library/view/chaos-engineering/9781491988459/
サイトリライアビリティエンジニアの育成	ジェニファー・ペトフ、JC・バン・ウィンケル、プレストン・ヨシオカ with ジェシー・ヤン、ジーザス・クリメント・コラード、マイク・テイラー	https://landing.google.com/sre/resources/practicesandprocesses/training-site-reliability-engineers/

SREファウンデーションコース付加価値の高いリソース

講座で紹介されたケーススストーリー

会社概要	モジュール	リンク
アクセンチュア	3：トイルの削減	https://techbeacon.com/devops/how-accenture-retrofitted-site-reliability-engineering
ブルームバーグ	1: SREの原則と実践	• https://player.fm/series/devops-chat/site-reliability-engineering-sre-bloomberg-w-stig-sorenson • https://www.techatbloomberg.com/blog/bloomberg-bets-big-on-sres/ • https://www.ca.com/us/modern-software-factory/content/outsmarting-outages-bloomberg-banks-on-sre-for-reliability.html
エバーノート	2: サービスレベル目標とエラーバジェット	https://landing.google.com/sre/workbook/chapters/slo-engineering-case-studies/
ホームデポ	2: サービスレベル目標とエラーバジェット	https://landing.google.com/sre/workbook/chapters/slo-engineering-case-studies/
Netflix	6：アンチフラジリティと失敗からの学習	https://github.com/Netflix/SimianArmy
Sage Group	7: SREの組織的影響	https://www.meetup.com/DevOpsNorthEast/events/262263231/
スタンダード・チャータード	5: SREツールと自動化	https://www.youtube.com/watch?v=d5lMvK0YHTg
トリバゴ	4: モニタリングとSLI	https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=11&cad=rja&uact=8&ved=2ahUKEwj4m6HJ9qXjAhU_QEEAHX6-CgQQFjAKegQIABAC&url=http%3A%2F%2Fpages。catchpoint.com%2Frs%2F005-RHC-551%2Fimages%2FCatchpoint-Case-Study-Trivago.pdf&usg=AOvVaw3UUcZ1bqtes0_99EYcBZSm
VictorOps (Splunk)	8: SREとその他のフレームワーク、将来について	https://victorops.com/blog/site-reliability-engineer-sre-roles-and-responsibilities

DevOps Institute

SRE (Site Reliability Engineering) Foundation v1.1 Sample Exam with Answer Key

サンプル試験

1.SREの定義として最も適切なものはどれですか？

A. ソフトウェアエンジニアが運用を任されると起きること
B. ソフトウェアエンジニアリングの側面を取り入れ、インフラや運用の問題に適用する学問分野
C. SLOやエラーバジェットの管理など、プロダクション後のオペレーション業務を行う役割
D. ソフトウェアエンジニアリングの側面を取り入れ、プロダクション後のサービスの管理に適用する原則のこと

2.Googleによると、**DevOps**の柱とならないものはどれですか？

A. 組織のサイロ化を減らす
B. 失敗を当たり前のように受け入れる
C. 徐々に変化させる
D. 右から左へのフローを増やす

3.SREにおけるサービスの「管理対象」は何ですか？

A. サービスレベル目標（SLO）
B. サービスレベル指標（SLI）
C. エラーバジェット
D. サービスレベルアグリーメント（SLA）

4.失敗のコストを削減できるものはどれですか？

A. MTTR（Mean Time To Repair）の改善
B. 小さな変化を積み重ねる
C. カナリアデプロイメント
D. 上記の全て

5.最も広く追跡されているサービスレベル目標（**SLO**）はどれですか？

A. 故障率の変化
B. セキュリティ
C. 可用性
D. キャパシティ

6.Catchpointの調査によると、「レイテンシー」の定義は何ですか？

A. ダウンタイムに伴う遅延
B. ユーザーがリクエストをしてからレスポンスを受け取るまでにかかる総時間
C. サービスレベル目標（SLO）とエラーバジェットの相違
D. メッセージを伝達する際に発生する遅延

7.あるチームの月次サービスレベル目標（**SLO**）の可用性は**99.9%**です。彼らのエラーバジェットにはどのくらいの時間が割り当てられていますか？

A. 45分
B. 43分
C. 25分
D. 32分

8.SLOのVALETモデルの "T"が表しているのは何ですか？

A. トラフィック
B. チケット
C. テスト
D. 修復にかかる時間

9.何百万人もの顧客に利用されている旅行会社のウェブサイトで、合意したエラーバジェットを超える**1**時間の障害が発生しました。ビジネスへの影響として、どのようなものが考えられますか？

A. 関連アプリケーションの不具合
B. 従業員の注意力不足
C. サービスレベル指標（SLI）の未達
D. ソーシャルメディアからの反発

10.エラーバジェットは毎月ゼロになるまで使用すべきですか？

A. はい
B. いいえ

11.あるヘルスケア企業の**SRE**チームは、自動化できる可能性のある運用タスクを検討し始めました。自動化可能なタスクの例としてはどのようなものがありますか？

A. 定常業務
B. 非難なしのポストモーテム
C. 本番導入を承認するための物理的なミーティング
D. 企画セッション

12.問題を解決するための実験は、トイルの一種ですか？

A. はい
B. いいえ

13.20年の歴史を持つあるソフトウェア会社は、多くの新興企業に囲まれて競争力を維持するのに苦労しています。新機能を市場に投入するのに時間がかかりすぎ、リリースされた機能にはエラーが多いです。この会社がより速く、より高い品質を実現するには、どのような方法がありますか？

A. サービスレベル目標（SLO）
B. トイルの軽減
C. エラーバジェットの管理
D. 自動的にスケーリングするインフラストラクチャ

14.SREの時間の少なくとも**50%**を費やすべきエンジニアリング作業はどれですか？

A. サービスの改善
B. エラーバジェットポリシーの遵守
C. サービスサポートの向上
D. コスト削減

15.Googleが**SRE**のエンジニアリング作業に**50%**ルールを導入した理由の一つは何ですか？

A. SREが開発者から学ぶ時間を設けること
B. 1つのチームや個人が、トイルの自動化を助けるのに十分なエンジニアリングスキルを持つことを保証するため
C. 1つのチームや個人が、「運用」にならないようにするため
D. 自動化の評価と導入に十分な時間を確保するため

16.サービスレベル指標（**SLI**）を説明したものはどれですか？

A. サービスの信頼性の目標レベル
B. サービスレベルの目標が達成されているかどうかを示すデータ
C. サービスレベルを定めた正式な契約
D. 顧客が定義したサービスレベルの要求事項

17.多くの数字がサービスレベル目標（**SLI**）として役立ちますがが、**Google**は**SLI**をどのように扱うことを推奨していますか？

A. 数式
B. 自動化されたアルゴリズム
C. 比率
D. 観測可能なデータ

18.複数のリモートエンドポイントからデータを収集・集計する自動化されたプロセスを何と呼びますか？

A. アラート
B. 可観測性
C. テレメトリー
D. アプリケーションパフォーマンスマネジメント

19.アプリケーションパフォーマンスマネジメントの目的は何ですか？

A. ネットワーク、サーバー、アプリケーションソフトウェアなどのアプリケーションに関するリモートデータの監視と集計
B. ソフトウェアアプリケーションのパフォーマンスと可用性の監視と管理
C. システムのリソースやパフォーマンスを監視するためのハードウェアまたはソフトウェアコンポーネントの使用
D. 異常検知

20.モニタリングアプローチに含まれないものはどれですか？

A. 何が正しく何が間違っているかについてのルールの確立
B. 時間軸での集計と適切なスケールでのグラフ化
C. SLOと関連するSLIのダッシュボードで表示
D. インシデント対応を追跡するインシデントチケットシステム

21.ある運用チームが新しい監視ソリューションを導入するにあたり、アラートを発生させる**CPU**の閾値について合意しようとしています。これは何の例ですか？

A. グラフ作成
B. ダッシュボード
C. インシデントの特定
D. 異常検知

22.テレメトリーの定義はどれですか？

A. ソフトウェアアプリケーションのパフォーマンスと可用性の監視と管理
B. コンピュータシステムのシステムリソースやパフォーマンスを監視するためのハードウェアまたはソフトウェアコンポーネント
C. エンジニアがシステムの定量的データを伝える手段
D. 高度に自動化された通信プロセスで、離れた場所やアクセスできない場所で測定値やその他のデータを収集し、モニタリングのために受信機器に送信すること

23.SREが主導するサービスの自動化で正しいものはどれですか？

A. テストは失敗することがわかっていることに集中する
B. 影響を考慮することなく、より多くの機能をプロダクションに送り出すことを推奨する
C. 一貫性を保つためには、Infrastructure as code (あるいはConfiguration as code)として環境をプロビジョニングする必要がある
D. 自動化されたデプロイメントは、SREによる手動のデプロイメントに置き換えられる

24.SREがプロダクション環境で行うべきテストの種類はどれですか？

A. 機能テスト
B. 非機能テスト
C. ユーザー要求テスト
D. 機能テストと非機能テスト

25.Infrastructure as Codeの主なメリットは何ですか？

A. インフラストラクチャのコードがアプリケーションのコードと同じように扱われるため、開発者はプロダクション環境でコードをテストできる
B. 開発、テスト、ステージング、プロダクションに同じコードを使用しているため、すべての環境が一貫している
C. コンテナ化を実現する
D. SREがより多くの機能をより早くプロダクションに投入しても、信頼性を高められる

26.サービスを外部から観測可能とするために、インストルメントが重要な理由はどれですか？

A. 正しいデータとサービスレベル指標（SLI）が返されていること、ログファイルが生成・保存されていることを確認するため
B. 結果の検証にあたってプリプロダクションテストとプロダクションテストが同一であることを確認するため
C. DevOpsパイプラインが、より優れたプロダクションテストによる信頼性の向上に焦点を当てていることを確認するため
D. 開発者やビジネス関係者がサービスにアクセスできるようにするため

27.将来の成長の可能性を概観することのメリットでないものはどれですか？

A. トイルの削減
B. 手直しの削減
C. セキュリティと監査のイベントの一元化
D. サービスの総所有コストの削減

28.Infrastructure as Codeを用いたプロビジョニングのメリットとして、当てはまらないものはどれですか？

A. プロダクション環境での変更点のテストや監査が容易になる
B. プリプロダクションとポストプロダクションのすべての環境の一貫性が確保される
C. 開発者はインフラストラクチャのコードを書くために環境を理解する必要がある
D. テスト環境でのエラーの再現が容易になる

29.失敗から学ぶ文化の導入に悩んでいる組織は何をすべきですか？

A. リリース前により多くテストする
B. レトロスペクティブを強化する
C. ダウンタイムのコストを調査する
D. 専門的なトレーニングへ投資する

30. Westrumによると失敗を調査のきっかけとするのはどのタイプの組織ですか？

A. 病理学的
B. 発生的
C. 官僚主義的
D. 学習的

31.定期的に避難訓練を行うことの主なメリットは何ですか？

A. 組織がカオスエンジニアリングを導入できるようにする
B. 事業継続計画のための資金が適切であることを確認する
C. 不測の事態や障害が発生した際にも事業を継続できるようにする
D. 重大なインシデントになる前にプロダクション環境のインシデントを特定する

32.カオスエンジニアリングの定義はどれですか？

A. 予期せぬ障害が発生した際のリカバリーを訓練するために、サーバーやインフラストラクチャをランダムに停止させること
B. 大規模なアウテージが発生してもビジネスをリカバリーできるように、頻繁に避難訓練を行うこと
C. プロダクション環境のソフトウェアシステムで実験を行い、システムが混乱や予期せぬ状況に耐えられるという確信を得ること
D. ソフトウェアエンジニアリングの側面を取り入れ、それをプロダクション後のサービス管理に適用した学問分野

33.SRE導入のアプローチにあるフル**SRE**説明しているものはどれですか？

A. サービス/デリバリーチームがSREのオーナーシップを持つ
B. SREと開発チームが責任とオンコールを共有する
C. 運用方法についてSREチームが全面的に責任を持つ
D. オペレーションチームとエンジニアリングチームを混成とする

34.サービスレベル目標（**SLO**）を確実に達成するためにはオンコールプログラムが不可欠ですが、だれがオンコールプログラムに参加すべきですか？

A. 全員
B. SREチームとインシデント・レスポンス・チーム
C. サービスデスクとSREチーム
D. オペレーションチームとエンジニアリングチームの代表者

35.Googleは**SRE**のオンコール時間を何%と提唱していますか？

A. 25%
B. 50%
C. 30%
D. 10%

36.非難なしのポストモーテムの重要なアウトプットの一つは何ですか？

A. 将来の類似インシデント発生を軽減するための、フォローアップアクションのリスト
B. 類似インシデント最小化のために、SREがトイルを軽減する全てのタスクのリスト
C. 将来の類似インシデントを回避するために必要なリソースについての議論
D. SLOとインシデントによってどのような影響を受けたかに関するリスト

37.ITIL/ITSMのどのような点が**SRE**を特にサポートできますか？

A. 変更諮問委員会（CAB）
B. サービスバリューシステム
C. サービスデスク
D. プロセスモデル

38.データベースリライアビリティエンジニアの主な仕事は何ですか？

A. 組織を官僚主義的な情報の流れから発生的な情報の流れに移行させるために支援する
B. 避難訓練や頻繁なデータの整合性テストによりデータの信頼性を確保する
C. データベースへカオスエンジニアリングの適用しフェイルオーバーとリストアの対応を確認する
D. データが適切にバックアップされ回復可能であることを保証する

39.レガシーアプリケーションも**SRE**の原則と実践から恩恵を受けることができます。レガシー処理に焦点を当てた場合、どのようなサービスレベル目標を設定できますか？

A. レガシーアプリケーションの数
B. COBOLコードのライン
C. メインフレームのプロセッサ速度
D. バッチ処理時間

40.SREとアジャイルのプラクティスを連携させることの、主なメリットではないものはどれですか？

A. 完了の定義が明確になり、よりエンド・ツー・エンドの視点を持つようになる
B. 顧客がソフトウェアを使うことで、より多くの価値を得ることができる
C. トイルのバックログが作業を可視化し、エンジニアリングや自動化のストーリーの優先順位付けに役立つ
D. SREがアジャイルあるいはスクラムチームに組み込まれる

解答

質問	正解	トピックエリア
1	**B**	**1: SRE**の原則と実践
2	**D**	**1: SRE**の原則と実践
3	**A**	**1: SRE**の原則と実践
4	**D**	**1: SRE**の原則と実践
5	**C**	**2:** サービスレベル目標とエラーバジェット
6	**D**	**2:** サービスレベル目標とエラーバジェット
7	**B**	**2:** サービスレベル目標とエラーバジェット
8	**B**	**2:** サービスレベル目標とエラーバジェット
9	**D**	**2:** サービスレベル目標とエラーバジェット
10	**A**	**2:** サービスレベル目標とエラーバジェット
11	**C**	**3:** トイルの削減
12	**B**	**3:** トイルの削減
13	**B**	**3:** トイルの削減
14	**A**	**3:** トイルの削減
15	**C**	**3:** トイルの削減
16	**B**	**4:** モニタリングとサービスレベル指標
17	**C**	**4:** モニタリングとサービスレベル指標
18	**C**	**4:** モニタリングとサービスレベル指標
19	**B**	**4:** モニタリングとサービスレベル指標
20	**D**	**4:** モニタリングとサービスレベル指標
21	**D**	**4:** モニタリングとサービスレベル指標
22	**D**	**4:** モニタリングとサービスレベル指標

23	**C**	**5: SRE**ツールと自動化
24	**D**	**5: SRE**ツールと自動化
25	**B**	**5: SRE**ツールと自動化
26	**A**	**5: SRE**ツールと自動化
27	**C**	**5: SRE**ツールと自動化
28	**C**	**5: SRE**ツールと自動化
29	**C**	**6:** アンチフラジリティと失敗からの学習
30	**B**	**6:** アンチフラジリティと失敗からの学習
31	**C**	**6:** アンチフラジリティと失敗からの学習
32	**C**	**6:** アンチフラジリティと失敗からの学習
33	**B**	**7: SRE**の組織的影響
34	**D**	**7: SRE**の組織的影響
35	**A**	**7: SRE**の組織的影響
36	**A**	**7: SRE**の組織的影響
37	**D**	**8: SRE**とその他のフレームワーク、将来について
38	**C**	**8: SRE**とその他のフレームワーク、将来について
39	**D**	**8: SRE**とその他のフレームワーク、将来について
40	**D**	**8: SRE**とその他のフレームワーク、将来について

Your Path to

DevOps Success

DevOps Institute is dedicated to advancing the human elements of DevOps success. Our goal is to help advance careers and support emerging practices using a role-based approach to certification which focuses on the most modern competencies and hireable skills required by today's organizations adopting DevOps.

Take the next steps in your learning and certification journey to DevOps success.

Click on a certification or visit www.devopsinstitute.com/certifications to learn more.

Become a Member

Join the fastest growing global communityof DevOps practitioners and professionals and gain access to invaluable learning content, the latest news, events, emerging practices, developyour network and advance your career.

You belong.

www.devopsinstitute.com/membership