AWS 起動時の耐障害性ベースライン

AWS 規範ガイダンス

AWS 規範ガイダンス: AWS 起動時の耐障害性ベースライン

Table of Contents

AWS 起動時の耐障害性ベースライン

Amazon Web Services (寄稿者)

2026 年 3 月 (ドキュメント履歴)

すべてのスタートアップファウンダーは緊張を知っています。チームは新機能を出荷するためにレースしていますが、顧客はすべてのリリースが機能することを期待しています。昨日の停止、顧客からの苦情の流入、ロードマップにはすでに約束された機能が含まれています。これはスタートアップの日々の現実です。イノベーションへのプレッシャーと安定したサービスを維持する必要性のバランスを取ります。

新機能を優先する本能は自然なものです。投資家は成長を望み、顧客はより多くの能力を求め、競合相手は立ち止まりません。しかし、経験上、レジリエンスの問題は、最も革新的なスタートアップであっても脱線する可能性があることを示しています。サービスの中断は、収益の損失を意味するだけでなく、顧客と構築するために一生懸命取り組んできた信頼を損ないます。実際、ロックソリッドの信頼性は、混雑した市場でサイレントな差別化要因になる可能性があります。

スタートアップにレジリエンスを構築しても、インフラストラクチャへの投資が遅くなったり、大規模な投資が行われたりするわけではありません。これは、問題を早期に防止する賢明な選択を行うことです。家を建てるようなものだと考えてください。すべてが構築されたら、構造上の問題を修正するよりも、最初に強固な基盤を構築する方がはるかに簡単です。回復力のあるプラクティスを初期のアーキテクチャと運用に織り込むことで、競争上の優位性を生み出すことができます。製品を提供するだけでなく、顧客ロイヤルティを構築する信頼性の高いエクスペリエンスを提供します。

AWS Startup Resiliency Baseline (AWS SRB) は、この課題に直面するチーム向けに特別に作成されました。スタートアップを特別なものにする速度を維持しながら、システムに信頼性を構築するための実用的な方法を提供します。重いプロセスやエンタープライズ規模の複雑さはなく、ビジネスとともに成長する実用的なパターンにすぎません。

このガイドでは、スタートアップ環境で AWS SRB を実装する方法について説明します。レジリエンスにとって本当に重要なこと、限られたリソースをどこに集中させるか、会社の成長に合わせてスケールするプラクティスを構築する方法を特定するのに役立ちます。

ガイダンスの範囲

スタートアップレジリエンスを構築する実践的なステップを検討するときは、このガイダンスが組織のより広範な成長軌跡のどこに当てはまるかを理解することが重要です。 AWS Startup Resiliency

Baseline は、耐障害性ライフサイクルフレームワークと一致しています。これは初期ロードマップとして機能し、最初のアプリケーションをデプロイする際の開発オーバーヘッドを最小限に抑えながら、基本的なレジリエンス対策を実装するのに役立つように設計されています AWS。これを、中断から効果的に回復できる信頼性の高いシステムを構築するためのスターターキットと考えてください。

AWS Well-Architected フレームワークとこの耐障害性ガイダンスは相互に補完し合い、安全で信頼性の高いインフラストラクチャを構築するのに役立ちます。このガイドでは、回復力のプラクティスについて詳しく説明します。一方、 AWS Well-Architected フレームワークは、より広範なアーキテクチャのベストプラクティスを提供します。の を使用して、これらのプラクティスに照らしてアーキテクチャを評価できますAWS Well-Architected Tool AWS アカウント。

責任共有モデル

クラウドで回復力のあるシステムを構築する上で重要な側面、つまり AWS とスタートアップとのパートナーシップを明確にすることが重要です。クラウドのレジリエンスは責任共有モデルに基づいて動作し、 AWS とチームはそれぞれ、システムのレジリエンスを確保するために異なる役割を果たします。

AWS は、サービスを強化する基盤となるクラウドインフラストラクチャの耐障害性に責任を負います。これを、アプリケーションを構築する基盤と考えてください。これには、データセンター、ネットワークコンポーネント、アーキテクチャ AWS のサービス で使用するコアの耐障害性の維持が含まれます。

スタートアップの責任は、この基盤上に構築およびデプロイするシステムの耐障害性に重点を置いています。このガイドの推奨事項は、お客様の責任範囲に含まれます。これらのプラクティスを慎重に実装することで、信頼性の高い AWS インフラストラクチャを使用し、堅牢な復旧機能を組み込むアプリケーションを作成できます。

このモデルは、単独でレジリエンスを構築することがないことを意味します。顧客や事業運営に直接影響する側面に集中しながら、実証済みの信頼できる基盤を構築しています。このガイドでは、この責任共有モデルを最大限に活用し、信頼性の高い AWS インフラストラクチャを使用しながら、アプリケーションが中断から正常に回復するように実用的な対策を実装する方法について説明します。

ステージ 1: 目標を設定する

チームが主要な製品発売前の最後のスプリントにいるとします。新機能は画期的であり、投資家の興奮が高まっています。その後、定期的なデプロイ中に、コアサービスがダウンします。顧客の苦情が E メールに溢れていると、2 つの質問がわかりにくくなります。どのくらいの期間オフラインにできますか? どのようなデータを失う可能性がありますか?

すべてがうまくいくことを期待することは、良い戦略ではありません。レジリエンスが最も重要である場所とそうでない場所を決定する体系的な方法が必要です。ここで、ビジネス影響分析 (BIA) が重要になります。これは、レジリエンスにどこに投資するかについて、情報に基づいた意思決定を行うのに役立ちます。BIA は、システムのどの部分が本当にロックソリッドステートの信頼性を必要とし、どの部分がある程度の柔軟性を許容できるかを理解するのに役立ちます。

まず、コアユーザージャーニーをマッピングします。それぞれについて、次のことを自問してください。

- これが中断された場合の影響は何ですか?
- サービスをどのくらい早く復元する必要がありますか?
- 保護するために重要なデータは何ですか?

これは単なる技術的な演習ではなく、信頼性の問題がビジネスに与える影響を理解するのに役立ちます。収益の損失は始まりにすぎません。停止によって顧客の信頼が損なわれたり、規制要件に違反したり、競合相手に優位性を与えたりする方法を検討してください。

この分析から、目標復旧時間 (RTO) と目標復旧時点 (RPO) の 2 つの重要な数値をユーザージャーニーごとに導き出します。RTO は、そのジャーニーを復元する速度を定義します。RPO は、顧客が許容できるデータ損失の量を定義します。これらのビジネス主導のターゲットは、システムのあらゆる部分を過剰に設計することなく、選択したコンポーネントとその設計方法をガイドします。

このアプローチの長所は、限られたリソースを最も重要な場所に集中させるのに役立つことです。おそらく、コアトランザクション処理にはほぼ即時の復旧とデータ損失ゼロが必要ですが、レコメンデーションエンジンはより長いダウンタイムを許容できます。明確な目標を設定することで、迅速な機能開発を継続しながら、戦略的にレジリエンスを構築できるフレームワークを作成します。

これらの目標を明確に文書化します。これらはエンジニアリングチームだけのものではありません。企業の顧客に売り込んだり、投資家と技術的なデューデリジェンスを行ったりする場合、このドキュメントは、ビジネス継続性について批判的に考えたことを示しています。

これらのターゲットは、スタートアップの成長に合わせて進化します。最初の数千人のユーザーのレジリエンスのニーズは、最初のエンタープライズクライアントのニーズとは異なります。今日現実的に達成できる目標から始めますが、スケールに合わせてどのように強化するかを計画します。

このガイドでは、これらの目的を達成するレジリエンス対策を実装する方法について説明します。これらのターゲットの設定は、重要な最初のステップです。イノベーションと安定性の絶え間ない緊張を乗り越えるためのコンパスであり、顧客に信頼できる価値を提供するシステムを構築するのに役立ちます。

ステージ 2: 設計と実装

このセクションでは、レジリエンス目標の実現について説明します。ビジネスにとって最も重要なものをマッピングし、構築する時が来ました。イノベーションを遅らせずにレジリエンスを高めるにはどうすればよいですか?

AWS マネージドサービスをレジリエンスショートカットと考える。インフラストラクチャを維持する貴重なエンジニアリング時間を消費するのではなく、冗長性を処理するサービスを使用してください。例えば、Amazon Simple Storage Service (Amazon S3) を考えてみましょう。耐久性 AWS リージョン のために、データの複数のコピーを に自動的に保存します。追加のコードや深夜のページャーの義務は必要ありません。

コアアプリケーションコンポーネントについてはどうですか? 賢明な選択は、チームの影響を倍増させる可能性があります。サービスのバックボーンであるデータベースを考えてみましょう。独自のレプリケーションシステムを構築する代わりに、フェイルオーバーを自動的に処理する Amazon Aurora の使用を検討してください。これらの機能にはコストがかかる場合がありますが、チームの焦点はインフラストラクチャの維持からビジネス上の問題の解決にシフトします。このコストは、機能配信を高速化し、停止中の収益損失を回避することで相殺できます。

スタートアップはカスタムソリューションを構築する必要がある場合があります。これが革新的なスタートアップの本質です。これを行うときは、シンプルでありながらスマートに保ちます。Elastic Load Balancing グループと Amazon EC2 Auto Scaling グループを使用して、アプリケーションを複数のアベイラビリティーゾーンに分散します。1 つのアベイラビリティーゾーンに障害が発生した場合でも、ベーストラフィックを処理するように Auto Scaling グループの最小容量を設定します。これにより、複雑なアーキテクチャパターンなしで、ローカライズされた障害に対する回復力が得られます。スタートアップが成長し、顧客がより高いレジリエンスを要求するにつれて、より高度なアプローチに進化できます。

本番稼働環境と開発環境は別々の にしておくことをお勧めします AWS アカウント。動きが速いときはそれらを混在させようとしていますが、この境界はセーフティネットです。これにより、優れた意味を持つ実験が本番稼働用サービスを停止するのを防ぐことができます。これを「高速で物事を壊す」開発文化の保険と考える - 開発中の物事を壊し、生産を安定させる。

アプリケーションがサードパーティーのサービスに依存している場合は、障害に備えてください。支払いプロセッサに問題がある場合、システムは問題を適切に処理できますか? シンプルなサーキットブレーカーとフォールバックオプションを構築します。エラーメッセージを表示する代わりに、これらのトランザクションをキューに入れることができます。顧客は、完全にはなくても、物事を機能させ続けたことに感謝します。

構築時に文書化しますが、実用性を維持します。重要な決定の背後にある理由の記録に集中し、簡単な復旧プレイブックを作成します。インシデントが発生した場合は、これらを準備することが重要です。

完全なレジリエンスのために構築しているわけではなく、適切なレジリエンスのために構築しているわけです。オーバーエンジニアリングレジリエンスに費やされる 1 時間ごとに、顧客が求める機能に費やされない 1 時間です。 AWS マネージドサービスを基盤として使用し、最も重要な場所にターゲットを絞ったレジリエンスを追加し、ビジネスの成長に合わせてレジリエンスをスケールアップするための明確なパスを作成します。

次の章では、エンジニアリングリソースを消費せずにこれらの設計選択肢を検証する方法について説明します。スタートアップ企業にとって、テストは合理的な向上であり、アプリケーションの耐障害性に対する賢明な投資でなければなりません。

ステージ 3: 評価とテスト

回復力のある基盤を構築しましたが、実際にはどのような仕組みになっているでしょうか。レジリエンステストは、製品市場に適合していることを証明するためにレースをしているときに、高級品のように聞こえる場合があります。ただし、この機能の開発を妨げずにこれを行うには、賢明な方法があります。この章では、スタートアップのペースに合ったリーンで実用的なテストについて説明します。

から始めてAWS Resilience Hub、それを初期アーキテクチャ評価ツールと見なします。アーキテクチャのレジリエンス基盤の有益なベースラインレビューを提供します。一般的な設定パターンと潜在的な単一障害点を確認することで、基本的なインフラストラクチャのセットアップが復旧目標と一致しているかどうかを評価するのに役立ちます。複数のアベイラビリティーゾーン設定の欠落やバックアップポリシーの不完全など、明らかなギャップにフラグを付けることができます。Resilience Hub は、慎重なアーキテクチャレビューと重要なパスのターゲットを絞ったテストを補完しますが、置き換えるものではありません。

文書化された復旧目標を検証するには、開発環境で AWS Backupで毎月の復元テストをスケジュールします。エンジニアリング時間が必要ですが、実際のインシデント中にバックアップが機能しないことを検出するよりも安い場合があります。ユニットテストの実行やコードレビューなど、通常の開発サイクルの一部にします。目標は完「完」ではなく、必要なときに回復できることを確信することです。

スタートアップが成長し、顧客が依存し始めるにつれて、テストゲームを段階的にレベルアップします。新機能をデプロイする場合は、パイプラインに基本的なレジリエンスチェックを含めます。を使用して簡単なカオス実験を試してくださいAWS Fault Injection Service。本番稼働前の環境で開始し、小規模で開始します。本番環境の実験を検討する前に、開発中の遅延した API レスポンスをアプリケーションがどのように処理するかをテストします。信頼度が高まったら、これらのテストを徐々に拡張しますが、常に本番稼働前に最初に検証してください。スタートアップにとって、本番環境でモノを壊すことは、意図的に壊すことなく十分にリスクがあります。

キーはバランスです。テストに費やされる 1 時間は、新機能の構築に費やされない 1 時間です。ただし、いくつかの戦略的テストにより、顧客の信頼を失うような停止を防ぐことができます。が提供する自動ツールを使用して、手間のかかる AWS 作業を行い、お客様にとって最も重要なテストに集中します。これにより、イノベーションを遅らせることなく、アプリケーションの耐障害性を信頼できます。

次の章では、スタートアップのスケールに応じてこの基盤を進化させる方法について説明します。

ステージ 4: 運用

回復力のあるアプリケーションを構築し、テストしました。現在、日常の現実はそれを実行し続けています。しかし、スタートアップでは、すべてのオペレーションをモニタリングすることはできず、試すべきではありません。重要なのは、あまりにも多くのメトリクスを提供したり、チームに負担をかけたりすることなく、重要なことに注意し続けることです。

顧客の視点から始めます。Amazon CloudWatch Synthetics Canary は自動化された顧客として機能します。重要なユーザージャーニーを継続的にテストします。特に繁忙期には、ログイン、テストアカウントを使用した購入のシミュレート、主要な機能へのアクセスを行います。これにより、カスタマーエクスペリエンスを理解し、実際のユーザーが問題を検出する前に問題を見つけることができます。Canary が失敗すると、顧客の観点から何かが間違っていることがすぐにわかります。

サポートインフラストラクチャを集中的にモニタリングして、この基盤を構築します。問題があることを示すシグナルは何ですか? Amazon CloudWatch は、これらのサインを追跡するダッシュボードの構築に役立ちます。技術的なメトリクスをモニタリングするだけでなく、ビジネスへの影響に結び付けます。たとえば、CPU 使用率が高いことは重要ですが、Canary で追跡しているカスタマーエクスペリエンスが低下する可能性があるためです。

実用的なアプローチとして、モニタリングをカスタマージャーニーにマッピングします。Software as a Service (SaaS) プラットフォームを実行している場合は、API の応答時間、認証の成功率、コア機能の可用性を重視している可能性があります。これらのメトリクスがドリフトしたときに通知するアラートを設定します。ただし、選択する必要があります。すべてのアラートはアクションを要求する必要があります。「おそらく何もない」ためにチームがアラートを無視し始めた場合、設定が多すぎるか、間違ったメトリクスを追跡しています。

チームが既に使用しているツールを使用して、これらのアラートをルーティングします。エンジニアが特定のメッセージングアプリケーションに住んでいる場合は、そこでアラートを送信します。目標は、新しいプロセスを作成せずに迅速に認識することです。アラートが発生すると、チームはアラートの意味と対処方法を正確に把握する必要があります。

運用ドキュメントを無駄なく実用的なものにします。コードを含むランブックをバージョン管理で保存しますが、これらは小説ではないことに注意してください。何かが壊れた場合、チームには明確で実用的なステップが必要です。各アラートは対応するランブックにリンクされ、各ランブックは 3 つの質問に答える必要があります。

- 何が壊れたのか。
- それが重要なのはなぜか。

- 解決策は?

シンプルなインシデント管理プロセスを実装します。複雑なフレームワークは必要なく、インシデントを構成するものと、事態がエスカレートしたときに誰を呼び出すかを明確に定義します。インシデントログは、アプリケーションの耐障害性の向上に役立つため、保持します。

重要なのは、警戒とオーバーヘッドの間の甘い場所を見つけることです。 AWS ツールを使用して、できることを自動化し、顧客に影響を与えるメトリクスのモニタリングに集中し、成長に合わせて進化するのに十分な軽量なプロセスを維持します。

次の章では、スタートアップを特別なものにするスピードとイノベーションを犠牲にすることなく、レジリエンスの考え方を育む方法について説明します。結局のところ、レジリエンスはテクノロジーに関するものと同じくらい人々に関するものです。

ステージ 5: 対応と学習

スタートアップを実行すると、複雑な事後プロセスによってチームが遅くなる可能性があります。この章では、インシデントを行政機関の演習に変えることなく、インシデントから学ぶ方法について説明します。

インシデント学習を既存のリズムに統合します。チームにすでに定期的な会議がある場合は、10 分間で最近のインシデントについて話し合います。次のような実用的な質問に焦点を当てます。

- ランブックは役に立ちましたか?
- アラートは適切なタイミングで行われましたか?
- AWS マネージドサービスはこれを防ぐことができましたか?

非難ではなく、アクションに集中してください。スタートアップでは、完璧なシステムを構築していません。何か問題が発生するたびに改善されるシステムを構築しています。

チケットシステムを使用してインシデントを追跡できます。特殊なツールは必要ありません。インシデントのタイムライン、顧客への影響、実行された復旧手順、教訓を含むシンプルなテンプレートを作成します。アクティブに使用すると、このカメラは機関メモリになります。オンボーディング中に過去のインシデントを確認し、新しいエンジニアを高速化します。類似システムを設計するときは、アーキテクチャレビューで参照してください。それらをゲームデーにプルして、実際のイベントに基づいて現実的な障害シナリオを作成します。テンプレートは何が起こったのかをキャプチャし、定期的に使用すると組織学習に変換されます。

スタートアップが増えるにつれて、パターンが現れます。特定のコンポーネントが失敗する頻度が高い場合や、特定のタイプの変更が原因で問題が発生する場合があります。これらのパターンを使用して、レジリエンスへの投資をガイドします。データベースのフェイルオーバーが原因で問題が発生した場合は、複数のアベイラビリティーゾーンの設定を改善することを検討してください。サードパーティーのサービス中断が一般的なテーマである場合は、サーキットブレーカーの改善を検討してください。

目標は、考えられるすべての障害を防ぐことではありません。これは不可能であり、速度が下がりすぎます。目標は、急速に成長している間に、迅速に学習し、迅速に適応し、アプリケーションの信頼性を十分に維持することです。各インシデントを機会として使用して、システムの耐障害性を高め、チームの知識を深め、お客様のサービスに対する信頼度を高めます。スタートアップの場合、スピードと学習ビートの完成度。イノベーションを遅らせることなく、インシデントから学ぶのに役立つ軽

量なプロセスを作成します。レジリエンスに関するベストプラクティスは、チームが実際に使用するものです。

スタートアップの次のステップ

ハートレースを起こした深夜のデプロイを覚えていますか? または、主要な顧客がレジリエンス対策について質問したその瞬間ですか? スタートアップのレジリエンスジャーニーは、これらの瞬間をなくすことではなく、自信を持って向き合うことです。

このガイドでは、現実的な復旧目標の設定、 AWS マネージドサービスの使用、リーンテストの実装、重要な事項のモニタリング、インシデントからの学習など、レジリエンスへの実用的な道筋について説明します。このアプローチは、ユーザーとともに成長するため、スタートアップにとって強力です。スタートアップが今日の問題から回復するのを支援するのと同じフレームワークが、将来のエンタープライズ顧客にサービスを提供する基盤を構築します。

レジリエンスを製品ロードマップのように考えてみましょう。すべてを一度に構築するわけではありません。コアサービスを保護することから始めます。重要なカスタマージャーニーのモニタリングを追加します。明らかな問題を検出する基本的なテストを実装します。学習したことを文書化します。各ステップは管理可能で実用的です。最も重要なのは、配送機能を妨げないことです。

スタートアップが成長するにつれて、レジリエンスのニーズは進化します。エンタープライズのお客様は、より厳しい質問をすることがあります。ピークトラフィックは新しい上限に達する可能性があります。国際展開には新しい課題があります。基盤に回復力を構築したので、適応する準備が整います。

スタートアップの目標は、完全な稼働時間やインシデントゼロではありません。目標は、スタートアップを特別なものにする速度を維持しながら、顧客の信頼を得るのに十分な信頼性のシステムを構築することです。このアプローチに従うことで、完璧なシステムを作成するよりも価値のあるものを作成できます。スタートアップの目標に追いつきながら、各チャレンジから学習することで、毎日向上するシステムを作成できます。

レジリエンスジャーニーは終了しません。出荷するすべての機能、獲得するすべての顧客、処理するすべてのインシデントに応じて進化します。このガイドは、自信を持って実践的なステップを進めるのに役立つフレームワークを構築するのに役立ちます。最終的に、スタートアップの成功は、回復力とスピードのどちらを選択するかではありません。これは、両方を配信するスマートな方法を見つけることです。

リソース

AWS Well-Architected フレームワーク

- 信頼性の柱
  - REL05-BP01 該当するハードな依存関係をソフトな依存関係に変換するため、グレースフルデグラデーションを実装する
  - REL06-BP01 ワークロードのすべてのコンポーネントをモニタリングする
  - REL06-BP03 通知を送信する (リアルタイム処理とアラーム)
  - REL08-BP03 デプロイの一部として回復力テストを統合する
- 回復力に関する責任共有モデル

AWS 規範ガイダンス

- でのバックアップとリカバリのアプローチ AWS
- レジリエンスライフサイクルフレームワーク: レジリエンス向上のための継続的なアプローチ

その他の AWS リソース

- AWS 障害分離境界 (AWS ホワイトペーパー)
- クラウドアプリケーションの RPO および RTO ターゲットの確立 (AWS ブログ記事)
- でのデプロイオプションの概要 AWS (AWS ホワイトペーパー)

寄稿者

このガイドの寄稿者は次のとおりです。

オーサリング

- Dylan McCarroll、Associate Solutions Architect、 AWS
- Igor Fil、スタートアップソリューションアーキテクト、 AWS
- Parth Shah、シニアソリューションアーキテクト、 AWS
- Vrajesh Prajapati、ソリューションアーキテクト、 AWS

レビューアー

- Clark Richey、プリンシパル技術者、 AWS
- Bruno Emer、プリンシパル技術者、 AWS

テクニカルライター

- " AbouHarb、シニアテクニカルライター、 AWS

ドキュメント履歴

以下の表は、本ガイドの重要な変更点について説明したものです。今後の更新に関する通知を受け取る場合は、RSS フィードをサブスクライブできます。

変更	説明	日付
大幅な更新	AWS のサービス 全体で推奨事項と の使用を更新しました。	2026 年 3 月 6 日
初版発行	—	2024 年 1 月 24 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。