AWS ホワイトペーパー

AWS リソースのタグ付けのベストプラクティス

AWS リソースのタグ付けのベストプラクティス: AWS ホワイトペーパー

Table of Contents

AWS リソースのタグ付けのベストプラクティス

発行日: 2023 年 3 月 30 日 (ドキュメントの改訂)

Amazon Web Services (AWS) では、タグ形式で AWS のリソースにメタデータを割り当てることができます。各タグは、リソースや、そのリソースにあるデータに関する情報を保存するためのキーとオプション値で構成される簡単なラベルです。このホワイトペーパーでは、目的、チーム、環境、またはビジネスに関連するその他の基準でリソースを分類することができるタグ付けのユースケース、戦略、手法、ツールに焦点を当てます。一貫したタグ付け戦略を導入すれば、リソースのフィルタリングと検索、コストと使用状況の監視、AWS 環境の管理が容易になります。

このホワイトペーパーは、「AWS複数のアカウントを使用した環境の整理」ホワイトペーパーに記載されているプラクティスとガイダンスを基にしています。このホワイトペーパーを読む前にそちらのホワイトペーパーを一読することをお勧めします。AWS はクラウド基盤を総合的に確立することを推奨しています。追加情報については、「AWSクラウド基盤の構築」を参照してください。

Well-Architected の実現状況の確認

AWS Well-Architected フレームワークは、クラウド内でのシステム構築に伴う意思決定の長所と短所を理解するのに役立ちます。このフレームワークの 6 つの柱により、信頼性、安全性、効率、費用対効果、持続可能性の高いシステムを設計および運用するための、アーキテクチャのベストプラクティスを確認できます。AWS Management Consoleで無料で提供されている AWS Well-Architected Toolを使用すると、柱ごとに一連の質問に答えることで、これらのベストプラクティスに照らしてワークロードを評価できます。

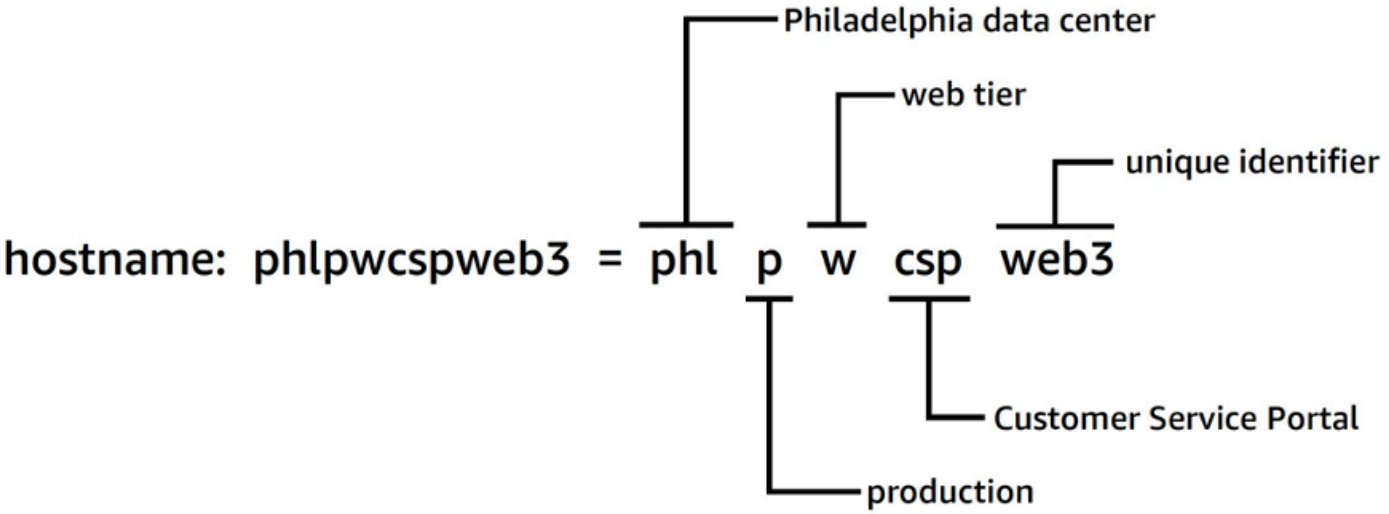
クラウドアーキテクチャに関する専門的なガイダンスやベストプラクティス (リファレンスアーキテクチャのデプロイ、図、ホワイトペーパー) については、AWS アーキテクチャセンターを参照してください。

序章

AWS を使用すると、Amazon EC2 インスタンス、Amazon EBS ボリューム、セキュリティグループ、AWS Lambda 関数などのリソースを作成して、ワークロードを簡単に AWS にデプロイできます。また、アプリケーションをホストし、データを保存し、時間の経過とともに AWS インフラストラクチャを拡張する AWS リソース群のスケーリングや拡張もできます。AWS 使用量が増え、複数のアプリケーションにまたがるリソースタイプが増えるにつれて、どのリソースがどのアプリケーションに割り当てられているかを追跡するメカニズムが必要になります。このメカニズムで、コスト

監視、インシデント管理、パッチ適用、バックアップ、アクセス制御などの運用アクティビティをサポートしてください。

オンプレミス環境では、この情報は、一般にナレッジ管理システム、文書管理システム、社内の Wiki ページに取り込まれます。構成管理データベース (CMDB) では、標準の変更管理プロセスで、関連する詳細なメタデータの保存や管理ができます。このアプローチでガバナンスは得られますが、開発と保守には追加の労力が必要です。リソースの名前付けには構造化されたアプローチが可能ですが、リソース名に格納できる情報は限られています。



リソース命名に対する構造化されたアプローチ

例えば、EC2 インスタンスには似たような機能の Name という定義済みのタグがある場合、ワークロードを AWS に移動するときに名前を付けることができます。

2010 年に、AWS はユーザーリソースにメタデータをアタッチするための柔軟でスケーラブルなメカニズムを備えたリソースタグを発表しました。このホワイトペーパーでは、ユーザーの AWS 環境全体で強固なタグ付け戦略を策定して実装するプロセスを紹介します。このガイダンスでは、意思決定や運用活動をサポートするタグ付けの一貫性と適用範囲を確保します。

タグとは

タグは、リソースに適用されるキーと値のペアであり、目的は、それらのリソースに関するメタデータを格納することです。タグは、キーとオプションの値で構成されるラベルです。現在、すべてのサービスとリソースタイプがタグをサポートしているわけではありません (「Services that support the Resource Groups Tagging API」を参照)。他のサービスでは、独自の API でタグをサポートしている場合があります。タグは暗号化されていないため、個人を特定できる情報 (PII) など、機密データの保存には使用しないでください。

AWS CLI、API、または AWS Management Console を使用してユーザーが作成して AWS リソースに適用するタグを、ユーザー定義タグと呼びます。AWS CloudFormation、Elastic Beanstalk や Auto Scaling などのいくつかの AWS サービスでは、作成し、管理するリソースに自動的にタグが割り当てられます。これらのキーをAWS 生成タグと呼び、通常は `aws` のプレフィックスが付けられます。このプレフィックスはユーザー定義のタグキーには使用できません。

AWS リソースに追加できるユーザー定義タグの数には、使用要件と制限があります。詳細については、『AWSジェネラルリファレンスガイド』の「タグ名の制限と要件」を参照してください。 AWS 生成タグは、これらのユーザー定義のタグ制限の対象にはなりません。

表 1 — ユーザー定義のタグキーと値の例

インスタンス ID	タグキー	タグ値
`i-01234567abcdef89a`	`CostCenter`	`98765`
	`Stack`	`Test`
i-12345678abcdef90b	`CostCenter`	`98765`
	`Stack`	`Production`

表 2 — AWS 生成タグの例

AWS 生成タグキー	根拠
`aws:ec2spot:fleet-request-id`	インスタンスを起動した Amazon EC2 スポットインスタンスリクエストを識別します。

AWS 生成タグキー	根拠
aws:cloudformation:stack-name	リソースを作成した AWS CloudFormation スタックを識別します。
lambda-console:blueprint	AWS Lambda 関数のテンプレートとして使用されるブループリントを識別します。
elasticbeanstalk:environment-name	リソースを作成したアプリケーションを識別します。
aws:servicecatalog: provisionedProductArn	プロビジョニングされた製品のAmazon リソースネーム (ARN)
aws:servicecatalog:productArn	プロビジョニング済み製品の起動元の製品の ARN。

AWS 生成タグは名前空間を形成します。例えば、AWS CloudFormation テンプレートでは、まとめてデプロイするリソースのセットを stack で定義します。ここで、stack-name はリソースを識別するために割り当てるわかりやすい名前です。aws:cloudformation:stack-name などのキーを調べると、パラメータのスコープに使用される名前空間には、aws組織、cloudformationサービス、、stack-nameパラメータの 3 つの要素が使用されていることがわかります。

ユーザー定義タグも名前空間を使用できるため、組織識別子をプレフィックスとして使用することをお勧めします。そうすれば、タグが管理対象スキーマのものなのか、環境内で使用しているサービスやツールによって定義されたものなのかを、簡単に区別できます。

「Establishing Your Cloud Foundation on AWS」ホワイトペーパーでは、実装すべき一連のタグを推奨しています。1 つのタグで使用できるパターンやリストは、多くの場合、企業によって異なります。表 3 の例を見てください。

表 3 — 同じタグキー、異なる値の検証ルール

組織	タグキー	タグ値の検証	タグ値の例
会社 A	CostCenter	5432, 5422, 5499	5432
会社 B	CostCenter	ABC*	ABC123

これら 2 つのスキーマが別々の組織に属していれば、タグが競合しても問題はありません。ただし、これら 2 つの環境を統合すると、名前空間が競合する可能性があり、検証がより複雑になります。このようなシナリオはないと思われますが、企業は買収や合併されることがあります。また、マネージドサービスプロバイダー、ゲームパブリッシャー、ベンチャーキャピタル企業と提携しているクライアントなど、異なる組織のアカウントが共有された AWS 組織の一部である状況も考えられます。ビジネス名をプレフィックスとして使用して一意の名前空間を定義すれば、表 4 に示すような問題を回避できます。

表 4 - タグキーにおける名前空間の使用

組織	タグキー	タグ値の検証	タグ値の例
会社 A	company-a :CostCenter	5432, 5422, 5499	5432
会社 B	company-b :CostCenter	ABC*	ABC123

事業の買収や売却が定期的に行われる大規模で複雑な組織では、このような状況がより頻繁に発生します。新規買収のプロセスと慣行がグループ全体で共有されれば、このような状況の問題は解決します。名前空間を明確にしておくと、古いタグの使用状況を報告して関連チームに連絡でき、新しいスキーマを採用してもらえるので便利です。名前空間は、範囲を示したり、組織の所有者と連携したユースケースや責任範囲を表すためにも使用できます。

表 5 - タグキー内のスコープまたはユースケーススコープの例

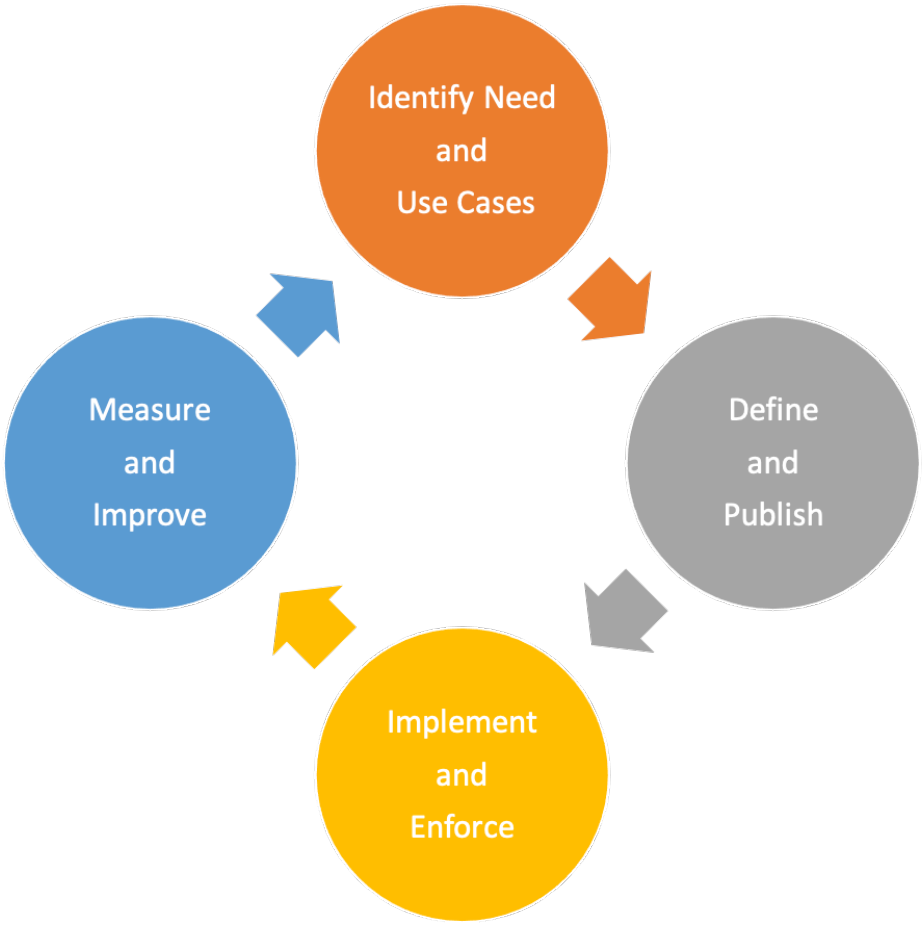
ユースケース	タグキー	根拠	許可された値
データ分類	example-inc:info-sec:data-classification	情報セキュリティ定義のデータ分類セット	sensitive , company-confidential , customer-identifiable
操作	example-inc:dev-ops:environment	テスト環境と開発環境のスケジューリングの実装	development , staging, quality-

ユースケース	タグキー	根拠	許可された値
			assurance , production
ディザスタリカバリディザスタリカバリ	example-inc:disaster-recovery:rpo	リソースの目標復旧時点 (RPO) の定義	6h, 24h
コスト配分	example-inc:cost-allocation:business-unit	財務チームには、各チームの使用状況と支出に関するコストレポートが必要です。	corporate , recruitment , support, engineering

タグはシンプルで柔軟性があります。タグのキーと値はどちらも可変長の文字列で、幅広い文字セットをサポートできます。長さと文字セットの詳細については、AWS 全般リファレンスの「AWS リソースのタグ付け」を参照してください。タグでは大文字と小文字を区別します。つまり、costCenter と costcenterとは別のタグキーです。国によって単語のスペルが異なる場合があり、それがキーに影響する可能性があります。たとえば、米国ではキーを costcenter と定義しても、英国では costcentre が優先される場合があります。リソースタグ付けの観点から見ると、これらは異なるキーです。タグ付け戦略の一環として、スペル、大文字と小文字、句読点を定義します。これらの定義は、リソースを作成または管理するすべての人の参考資料として使用してください。このトピックについては、次のセクション「タグ付け戦略の構築」でさらに詳しく説明します。

タグ付け戦略の構築

運用における多くのプラクティスと同様に、タグ付け戦略の実装は反復と改善のプロセスです。当面の優先事項から小さく始めて、必要に応じてタグ付けスキーマを拡張します。



タグ付け戦略の反復と改善サイクル

このプロセスを通じて、責任の所在が説明責任と進歩の鍵になります。タグはさまざまな目的に使用できるため、全体的なタグ付け戦略を組織内の各責任分野に分けることができます。タグ付けにより、リソースの特性に依存するアクティビティにプログラム的にアプローチできます。タグ付けによってメリットが得られる利害関係者の範囲は、組織の規模と運用方法によって異なります。大規模な組織では、タグ付け戦略の構築と実施に関わるチームの責任を明確に定義することでメリットが得られます。タグ付けのニーズの識別 (ユースケースの定義) を担当するステークホルダーもいれば、タグ付け戦略の維持、実装、改善を担当するステークホルダーもいます。

オーナーシップを割り当てることで、戦略の個々の側面が実施しやすくなります。必要に応じて、このオーナーシップをポリシーとして定式化し、責任マトリックス (RACI: 実行責任者、説明責任者、協業先、相談先、報告先など) に文書化することも、責任分担モデルとして文書化することもできま

す。小規模な組織では、要件の定義から実装、強制まで、チームがタグ付け戦略において複数の役割を果たすことがあります。

ニーズとユースケースの定義

まずは、メタデータを利用したいという根本的なニーズを持つ利害関係者に働きかけることから戦略の構築を始めましょう。これらのチームは、レポート、自動化、データ分類などのアクティビティをサポートするためにリソースにタグを付ける必要のあるメタデータを定義します。リソースをどのように整理し、どのポリシーにマッピングする必要があるかを概説します。これらの利害関係者が組織内で果たすことができる役割と機能の例には次のようなものがあります。

- 財務部門や事業部門は、投資の価値をコストと照らし合わせて把握し、非効率な状況に対処する際に取るべきアクションの優先順位を決めます。コストと生み出される価値を理解すると、うまくいっていない事業部門や製品提供を見極めることができます。その結果、サポートの継続、代替案の採用 (SaaSオファリングやマネージドサービスの使用など) 、または不採算のビジネスオファリングの廃止について、根拠のある決定を下すことができます。
- ガバナンスとコンプライアンスでは、データの分類 (公開、機密、極秘など) 、特定のワークロードが特定の標準や規制に対する監査の対象となるかどうか、および権限、ポリシー、監視などの適切な制御と監視を適用するためのサービスの重要性 (サービスまたはアプリケーションがビジネスに不可欠なかどうか) を理解する必要があります。
- 運用部門と開発部門は、ワークロードのライフサイクル、サポート対象製品の実装段階、リリース段階 (開発、テスト、生産分割など) の管理、および関連するサポートの優先順位付けと利害関係者の管理要件を理解する必要があります。バックアップ、パッチ適用、オブザーバビリティ、非推奨などの義務も定義し、理解する必要があります。
- 「情報セキュリティ (InfoSec)」と「セキュリティオペレーション (SecOps)」では、適用すべき統制と推奨する統制の概要を述べます。通常、InfoSec は統制の実装を定義し、それらの統制の管理は一般的に SecOps が担当します。

ユースケース、優先順位、組織の規模、運用方法によっては、財務 (調達を含む)、情報セキュリティ、クラウド支援、クラウド運用など、組織内のさまざまなチームの代表者が必要になる場合があります。また、パッチ適用、バックアップと復元、監視、ジョブスケジューリング、ディザスタリカバリなどの機能については、アプリケーションオーナーやプロセスオーナーの代表者も必要です。これらの担当者は、タグ付け戦略の定義、実装、効果の評価に関わります。その場合、利害関係者やそのユースケースからさかのぼって取り組み、部門横断的なワークショップを実施する必要があります。ワークショップでは、各自の視点やニーズを共有して、全体的な戦略を推し進めます。参加者の例とさまざまなユースケースへの関わり方については、このホワイトペーパーの後半で説明します。

また、利害関係者は必須タグのキーを定義して検証します。オプションタグの範囲を推奨することもできます。例えば、財務チームでは、社内のコストセンター、事業単位、あるいはその両方にリソースを関連付ける必要があるかもしれません。そのため CostCenter や BusinessUnit などの特定のタグキーを必須にする必要があるかもしれません。個々の開発チームが、EnvironmentName や OptIn、OptOut などの追加のタグを自動化目的で使用することを決定する場合があります。

主要利害関係者は、タグ付け戦略のアプローチについて合意し、次のようなコンプライアンスやガバナンス関連の質問に対する回答を文書化する必要があります。

- 取り組む必要があるユースケースは何か？
- リソースのタグ付け (実装) の責任者は誰か？
- タグはどのように適用され、どのような方法や自動化が使用されるのか (事前対応的か事後対応的か)？
- タグ付けの効果と目標はどのように測定されるのか？
- タグ付け戦略はどのくらいの頻度で見直すべきか？
- 誰が改善を推し進めるのか？ 改善はどのように行われるのか？

Cloud Enablement、Cloud Business Office、Cloud Platform Engineering などのビジネス部門が、進捗状況を測定し、障害を取り除き、重複する作業を減らすことで、タグ付け戦略の構築プロセスのファシリテーターの役割を果たし、タグ付け戦略の採用を促進し、適用の一貫性を確保することができます。

タグ付けスキーマの定義と公開

AWS リソースのタグ付けには、必須タグとオプションタグの両方に一貫したアプローチを採用します。包括的なタグ付けスキーマは、この一貫性を実現するのに役立ちます。始めるにあたって次の例を参照してお役立てください。

- 必須のタグキーについて合意する
- 使用できる値とタグ命名規則 (大文字または小文字、ダッシュまたはアンダースコア、階層など) を定義する
- 値が個人を特定できる情報 (PII) を構成しないことを確認する
- 新しいタグキーを定義して作成できる担当者を決める
- 新しい必須タグ値を追加する方法とオプションタグを管理する方法について合意する

以下のタグ付けカテゴリの表を見てください。この表は、タグ付けスキーマに含める内容のベースラインとして使用できます。さらに、タグキーに使用する規則と、それぞれで使用できる値を決定する必要があります。タグ付けスキーマは、使用環境に合わせてそれらを定義する文書です。

表 6 — 最終的なタグ付けスキーマの例 (パート 1)

ユースケース	タグキー	根拠	使用できる値 (リストまたは値のプレフィックス/サフィックス)	コスト配分に使用	リソースタイプ	範囲	必須
コスト配分	example-inc:cost-allocation: ApplicationId	事業部門ごとの発生コストと生成価値を対比して追跡する	DataLakeX , RetailSiteX	Y	すべて	すべて	必須
コスト配分	example-inc:cost-allocation: BusinessUnitId	ビジネスユニットごとにコストを監視する	Architecture , DevOps, Finance	Y	すべて	すべて	必須
コスト配分	example-inc:cost-allocation: CostCenter	コストセンターごとにコストを監視する	123-*	Y	すべて	すべて	必須
コスト配分	example-inc:cost-allocation :Owner	このワークロードに責任があるのはどの予算担当か	Marketing , RetailSupport	Y	すべて	すべて	必須
アクセスコントロール	example-inc:access-control: LayerId	役割に基づいてリソースへのアクセスを許可するサブコンポー	DB_Layer, Web_Layer , App_Layer	N	すべて	すべて	任意

表 6 — 最終的なタグ付けスキーマの例 (パート 2)

ユースケース	タグキー	根拠	使用できる値 (リストまたは値プレフィックス/サフィックス)	コスト配分に使用	リソースタイプ	範囲	必須
DevOps	`example-inc:operations: Owner`	リソースの作成とメンテナンスを担当するのはどのチーム/グループか	`Squad01`	N	すべて	すべて	必須
ディザスタリカバリ	`example-inc:disaster-recovery:rpo`	リソースに対する目標復旧時点 (RPO) を定義する	6h, 24h	N	S3、EBS	Prod	必須
データ分類	`example-inc:data:classification`	コンプライアンスとガバナンスのためにデータを分類する	`Public`, `Private`, `Confidential` , `Restricted`	N	S3、EBS	すべて	必須
コンプライアンス	`example-inc:compliance:framework`	ワークロードが適用されるコンプライアンスフレームワークを特定する	`PCI-DSS`, `HIPAA`	N	すべて	Prod	必須

タグ付けスキーマを定義したら、そのスキーマを関連するすべての利害関係者がアクセスして簡単に参照して、更新を追跡できようにバージョン管理されたリポジトリで管理します。このアプローチで効率が向上し、アジリティが高まります。

AWS Organizations - タグポリシー

AWS Organizations のポリシーを使用すると、組織内の AWS アカウント に新たなタイプの管理機能を追加して適用できます。タグポリシーとは、プラットフォームがタグ付けスキーマをレポートし、必要に応じて AWS 環境内で適用できるように、タグ付けスキーマを JSON 形式で表現する方法です。タグポリシーでは、特定のリソースタイプのタグキーに使用できる値を定義します。このポリシーは、値のリストでも、プレフィックスにワイルドカード文字 (*) が続く形式でもかまいません。単純なプレフィックス方式は、個別値のリストほど厳密ではありませんが、メンテナンスは少なくて済みます。

以下の例は、タグ付けポリシーを定義して特定のキーに許容される値を検証する方法を示したものです。わかりやすい表形式のスキーマの定義に基づいて、この情報を 1 つ以上のタグポリシーに転記できます。所有権の委任をサポートするために個別のポリシーを使用することもでき、また特定のシナリオにのみ適用されるポリシーもあります。

ExampleInc-CostAllocation.json

次に、コスト配分タグのレポートを行うタグポリシーの例を示します。

```
{
  "tags": {
    "example-inc:cost-allocation:ApplicationId": {
      "tag_key": {
        "@@assign": "example-inc:cost-allocation:ApplicationId"
      },
      "tag_value": {
        "@@assign": [
          "DataLakeX",
          "RetailSiteX"
        ]
      }
    },
    "example-inc:cost-allocation:BusinessUnitId": {
      "tag_key": {
        "@@assign": "example-inc:cost-allocation:BusinessUnitId"
      },
      "tag_value": {
```

```
        "@@assign": [
          "Architecture",
          "DevOps",
          "FinanceDataLakeX"
        ]
      }
    },
    "example-inc:cost-allocation:CostCenter": {
      "tag_key": {
        "@@assign": "example-inc:cost-allocation:CostCenter"
      },
      "tag_value": {
        "@@assign": [
          "123-*"
        ]
      }
    }
  }
}
```

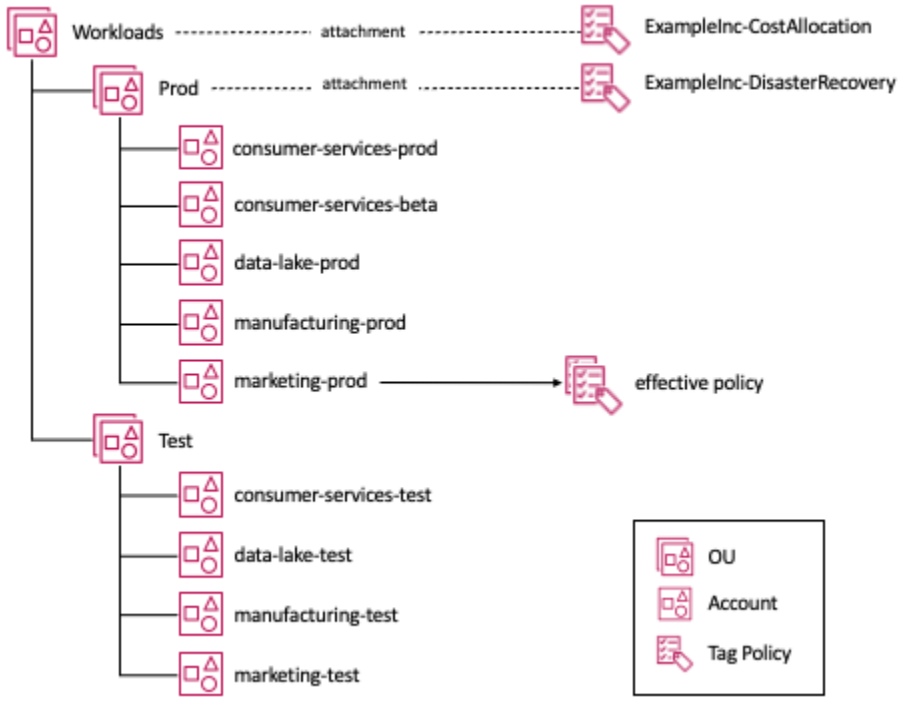
ExampleInc-DisasterRecovery.json

次に、ディザスタリカバリのレポートを行うタグポリシーの例を示します。

```
{
    "tags": {
        "example-inc:disaster-recovery:rpo": {
            "tag_key": {
                "@@assign": "example-inc:disaster-recovery:rpo"
            },
            "tag_value": {
                "@@assign": [
                    "6h",
                    "24h"
                ]
            }
        }
    }
}
```

この例では、ExampleInc-CostAllocation タグポリシーは Workloads OU にアタッチされているため、その Prod と子 OU Test の両方のすべてのアカウントに適用されます。同様

に、ExampleInc-DisasterRecovery タグポリシーは Prod OU にアタッチされているため、この OU の下位のアカウントにのみ適用されます。『複数のアカウントによる環境の整理』ホワイトペーパーでは、推奨される OU 構造について詳しく説明しています。



OU 構造へのタグポリシーのアタッチ

図の marketing-prod アカウントを見ると、両方のタグポリシーがこのアカウントに適用されるため、効果的なポリシーを考えました。それは、1 つのアカウントに適用される所定のタイプのポリシーのコンボリューションです。リソースを主に手動で管理している場合は、コンソールの Resource Groups & Tag Editor: タグポリシー にアクセスして有効なポリシーを確認できます。Infrastructure as Code (IaC) またはスクリプトでリソースを管理している場合は、AWS::Organizations::DescribeEffectivePolicy API 呼び出しを使用できます。

タグ付けの実装と実施

このセクションでは、手動、Infrastructure as Code (IaC)、継続的インテグレーション/継続的デリバリー (CI/CD) といったリソース管理戦略に使用できるツールを紹介します。これらのアプローチの重要な側面は、デプロイ頻度がますます高まっていることです。

手動管理のリソース

これらは通常、導入の基盤段階または移行段階に分類されるワークロードです。多くの場合、これらのワークロードは従来の書面による手順で構築された単純でほとんど静的なワークロードや、オンプレミス環境から CloudEndure などのツールを使用してそのまま移行されたワークロードです。Migration Hub や CloudEndure などの移行ツールでは、移行プロセスの一部としてタグを適用できます。ただし、元の移行時にタグが適用されなかった場合や、それ以降にタグ付けスキーマが変更された場合は、タグエディタ (AWS Management Console の機能の一つ) で、さまざまな検索条件を利用してリソースを検索し、タグを一括で追加、変更、削除できます。検索条件には、特定のタグや値の有無にかかわらず、リソースを含めることができます。AWS リソースタギング API では、これらの機能をプログラムで実行できます。

これらのワークロードを最新化すると、Auto Scaling グループなどのリソースタイプが導入されます。これらのリソースタイプにより、柔軟性が高まり、レジリエンスが向上します。自動スケーリンググループはユーザーに代わって Amazon EC2 インスタンスを管理しますが、場合によっては、EC2 インスタンスに手動で作成したリソースで一貫したタグを付ける必要性が生ずる場合があります。Amazon EC2 起動テンプレートでは、自動スケーリングが作成するインスタンスに適用するタグを指定することができます。

ワークロードのリソースを手動で管理するときは、リソースのタグ付けを自動化すると便利です。さまざまなソリューションが利用できます。1 つのアプローチは、`required_tags` を確認してから Lambda 関数起動して適用できる、AWS Config ルール を使用することです。 AWS Config ルールについては、以降のセクションで詳しく説明します。

Infrastructure as code (IaC) 管理リソース

AWS CloudFormation では AWS 環境内のすべてのインフラストラクチャーリソースを共通言語でプロビジョニングできます。CloudFormation テンプレートは、AWS リソースを自動的に作成する JSON ファイルまたは YAML ファイルです。CloudFormation テンプレートで AWS リソースを作成するときは、サポート対象のリソースタイプに CloudFormation リソースタグプロパティでタグを適用できます。IaC でタグとリソースを管理すると、一貫性が保たれます。

AWS CloudFormation でリソースを作成すると、このサービスでは AWS CloudFormation テンプレートで作成されたリソースに定義済みの AWS タグのセットが自動的に適用されます。次のようなものがあります。

```
aws:cloudformation:stack-name
aws:cloudformation:stack-id
```

```
aws:cloudformation:logical-id
```

リソースグループは AWS CloudFormation スタックに基づいて簡単に定義できます。これらの AWS 定義済みタグは、スタックで作成されたリソースに継承されます。ただし、Auto Scaling グループ内の Amazon EC2 インスタンスの場合は、ユーザーの AWS CloudFormation テンプレート内の Auto Scaling グループの定義に AWS::AutoScaling::AutoScalingGroup TagProperty を設定する必要があります。また、Auto Scaling グループで EC2 起動テンプレート を使用している場合は、その定義でタグを定義できます。Auto Scaling グループまたは AWS コンテナサービスとともに EC2 起動テンプレート の使用を推奨します。これらのサービスは、Amazon EC2 インスタンスに一貫したタグ付けを行うのに役立ち、耐障害性を向上させ、コンピューティングコストを最適化する複数のインスタンスタイプと購入オプションにわたる自動スケーリング もサポートしています。

AWS CloudFormationフックでは、開発者はアプリケーションの重要部分と組織の標準との整合性を図ることができます。フックは警告を出す設定や、デプロイを妨げたる設定が可能です。この機能は、Auto Scaling グループが、起動するすべての Amazon EC2 インスタンスに顧客定義タグを適用するように設定されているかどうか、またはすべての Amazon S3 バケットが必要な暗号化設定で作成されていることを確認するなど、テンプレート内の主要な設定要素を確認するのに最適です。いずれの場合も、このコンプライアンスの評価は、デプロイ前に AWS CloudFormation フックでデプロイプロセスの早い段階にプッシュされます。

AWS CloudFormation では、テンプレートからプロビジョニングしたリソース (「ドリフト検出をサポートするリソース」を参照) が変更され、リソースが想定していたテンプレート構成と一致しなくなったことを検出できます。これをドリフトと呼びます。IaC で管理されているリソースに自動化でタグを適用すると、タグを変更することになり、ドリフトが生じます。IaC を使用するとき、現在、推奨されているのは、タグ付け要件を IaC テンプレートの一部として管理し、AWS CloudFormation フックを実装し、自動化で使用できる AWS CloudFormation ガードルールセットを公開することです。

CI/CD パイプラインの管理リソース

ワークロードの成熟度が高まるにつれて、継続的インテグレーションや継続的デプロイ (CI/CD) などの手法が採用される可能性が高くなります。これらの手法では、テストの自動化が進み、小さな変更を頻繁にデプロイしやすくなるため、デプロイリスクの軽減に役立ちます。デプロイによって生じる予期しない動作を検出するオブザーバビリティ戦略があれば、ユーザーへの影響を最小限に抑えながらデプロイを自動的にロールバックできます。1 日に何十回もデプロイする段階になると、さかのぼってタグを適用することはもはや現実的ではなくなります。すべてをコードまたは構成として表現し、バージョン管理し、可能な場合は本番環境にデプロイする前にテストと評価を行う必要がありま

す。複合開発運用（DevOps）モデルでは、多くのプラクティスでは運用上の配慮事項がコードとして扱われ、導入ライフサイクルの早い段階で検証されます。

理想を言えば、これらのチェックをプロセスのできるだけ早い段階で (AWS CloudFormation フックとともに表示されているように) プッシュして、AWS CloudFormation テンプレートが開発者のマシンを離れる前にポリシーを満たしていることを確信できるようにしたいものです。

AWS CloudFormationGuard 2.0 では、CloudFormation で定義できるあらゆるものに対して、予防的なコンプライアンスルールを記述することができます。テンプレートは開発環境のルールに照らして検証されます。この機能にはさまざまな用途があることは明らかですが、このホワイトペーパーでは、AWS::AutoScaling::AutoScalingGroupTagProperty が常に使用される例をいくつか見ていきます。

CloudFormation ガードルールの例を次に示します。

```
let all_asgs = Resources.*[ Type == 'AWS::AutoScaling::AutoScalingGroup' ]

rule tags_asg_automation_EnvironmentId when %all_asgs !empty {
    let required_tags = %all_asgs.Properties.Tags.*[
        Key == 'example-inc:automation:EnvironmentId' ]
    %required_tags[*] {
        PropagateAtLaunch == 'true'
        Value IN ['Prod', 'Dev', 'Test', 'Sandbox']
        <<Tag must have a permitted value
          Tag must have PropagateAtLaunch set to 'true'>>
    }
}

rule tags_asg_costAllocation_CostCenter when %all_asgs !empty {
    let required_tags = %all_asgs.Properties.Tags.*[
        Key == 'example-inc:cost-allocation:CostCenter' ]
    %required_tags[*] {
        PropagateAtLaunch == 'true'
        Value == /^123-/
        <<Tag must have a permitted value
          Tag must have PropagateAtLaunch set to 'true'>>
    }
}
```

このコード例では、タイプ AutoScalingGroup のすべてのリソースについてテンプレートをフィルタリングし、2 つのルールを設定しています。

- **tags_asg_automation_EnvironmentId** -このキーを持つタグが存在し、使用できる値リスト内の値があり、PropagateAtLaunch が次 true に設定されていることを確認します。
- **tags_asg_costAllocation_CostCenter** -このキーを持つタグが存在し、定義したプレフィックス値で始まる値を持ち、PropagateAtLaunch が true に設定されていることを確認します

強制

前述のように、Resource Groups & Tag Editor では、組織の OU に適用されるタグポリシーで定義されたタグ付け要件をリソースが満たしていない場所を識別することができます。組織のメンバーアカウント内から Resource Groups & Tag Editor コンソールツールにアクセスすると、そのアカウントに適用されるポリシーと、アカウント内でタグポリシーの要件を満たしていないリソースが表示されます。管理アカウントからアクセスした場合 (およびタグポリシーの AWS Organizations のサービスで [アクセス] が有効になっている場合)、組織内のリンクされているすべてのアカウントのタグポリシーコンプライアンスを確認できます。

タグポリシーそのもの中では、特定のリソースタイプへの強制機能を有効にできます。以下のポリシー例では、ec2:instance タイプと ec2:volume タイプのすべてのリソースにポリシーの遵守を義務付ける強制機能を追加しました。タグポリシーでリソースを評価するにはそのリソースにタグが必要であるなど、いくつかの制限が知られています。リストについては、「タグポリシーの強制をサポートするリソース」を参照してください。

ExampleInc-Cost-Allocation.json

次に、コスト配分タグのレポートを行うタグポリシーと強制するタグポリシーの両方またはいずれかの例を示します。

```
{
  "tags": {
    "example-inc:cost-allocation:ApplicationId": {
      "tag_key": {
        "@@assign": "example-inc:cost-allocation:ApplicationId"
      },
      "tag_value": {
        "@@assign": [
          "DataLakeX",
          "RetailSiteX"
        ]
      },
      "enforced_for": {
```

```
        "@@assign": [
          "ec2:instance",
          "ec2:volume"
        ]
      }
    },
    "example-inc:cost-allocation:BusinessUnitId": {
      "tag_key": {
        "@@assign": "example-inc:cost-allocation:BusinessUnitId"
      },
      "tag_value": {
        "@@assign": [
          "Architecture",
          "DevOps",
          "FinanceDataLakeX"
        ]
      },
      "enforced_for": {
        "@@assign": [
          "ec2:instance",
          "ec2:volume"
        ]
      }
    },
    "example-inc:cost-allocation:CostCenter": {
      "tag_key": {
        "@@assign": "example-inc:cost-allocation:CostCenter"
      },
      "tag_value": {
        "@@assign": [
          "123-*"
        ]
      },
      "enforced_for": {
        "@@assign": [
          "ec2:instance",
          "ec2:volume"
        ]
      }
    }
  }
}
```

AWS Config (**required_tag**)

AWS Config では、AWS リソースの設定を査定、監査、評価することができます。(AWS Config によるサポート対象のリソースタイプ を参照してください)。タグ付けの場合は、required_tags ルールを使用して特定のキーを持つタグがないリソースの識別に使用できます (「required_tags がサポートするリソースタイプ」を参照してください)。前の例から、すべての Amazon EC2 インスタンスにキーが存在するかどうかをテストできます。キーが存在しない場合、インスタンスは非準拠として登録されます。この AWS CloudFormation テンプレートには、表に記載されている必須キーが Amazon S3 バケット、Amazon EC2 インスタンス、Amazon EBS ボリュームに存在するかどうかをテストする AWS Config ルールが記載されています。

```
Resources:
 MandatoryTags:
   Type: AWS::Config::ConfigRule
   Properties:
     ConfigRuleName: ExampleIncMandatoryTags
     Description: These tags should be in place
     InputParameters:
       tag1Key: example-inc:cost-allocation:ApplicationId
       tag2Key: example-inc:cost-allocation:BusinessUnitId
       tag3Key: example-inc:cost-allocation:CostCenter
       tag4Key: example-inc:automation:EnvironmentId
     Scope:
       ComplianceResourceTypes:
       - "AWS::S3::Bucket"
       - "AWS::EC2::Instance"
       - "AWS::EC2::Volume"
     Source:
       Owner: AWS
       SourceIdentifier: REQUIRED_TAGS
```

リソースを手動で管理する環境では、AWS Lambda 関数による自動修復を使用して、不足しているタグキーをリソースに自動的に追加するように AWS Config ルールを拡張できます。これは静的なワークロードではうまく機能しますが、IaC やデプロイパイプラインを介してリソースを管理し始めると、次第に効果が低下します。

AWS Organizations - サービスコントロールポリシー (SCP) は、組織の権限の管理に使用できる組織ポリシーの一種です。SCP では、組織または組織単位 (OU) のすべてのアカウントで使用可能な最大権限を一元的に制御できます。SCP は、組織の一部であるアカウントが管理するユーザーとロールのみに反映されます。リソースには直接反映されませんが、アクションにタグを付ける権限を含

め、ユーザーとロールの権限が制限されます。タグ付けに関しては、SCP はどのようなタグポリシーを提供できるかという機能に加えて、さらにきめ細かいタグ適用を行うことができます。

次の例では、`example-inc:cost-allocation:CostCenter` タグが存在しない `ec2:RunInstances` リクエストはポリシーによって拒否されます。

以下は拒否 SCP です。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyRunInstanceWithNoCostCenterTag",
      "Effect": "Deny",
      "Action": "ec2:RunInstances",
      "Resource": [
        "arn:aws:ec2:*:*:instance/"
      ],
      "Condition": {
        "Null": {
          "aws:RequestTag/example-inc:cost-allocation:CostCenter": "true"
        }
      }
    }
  ]
}
```

設計上、連結アカウントに適用される有効なサービスコントロールポリシーを取得することは不可能です。SCP でタグ付けを強制する場合、開発者が自分のアカウントに適用されているポリシーにリソースを適合させられるように、開発者がドキュメントを入手できる環境を整える必要があります。アカウント内の CloudTrail イベントへの読み取り専用アクセス権限を提供すると、開発者がリソースを順守できない場合のデバッグ作業を支援できます。

タグ付けの有効性の評価と改善の推進

タグ付け戦略を実装したら、対象となるユースケースに対してその効果を評価することが重要です。効果の尺度はユースケースによって異なります。例:

- コスト属性-AWS Cost Explorer または AWS コストと使用状況レポートなどのツールを使用すると、支出に基づいてリソースのタグ付け範囲を評価できます。例えば、主に、特定のタグキーを監

視して、タグ付けされたリソースとタグ付けされていないリソースで、料金が発生しているリソースの割合を追跡できます。

- 自動化 - 予想したした結果が得られたかどうかを確認したい場合もあります。例えば、本番環境以外の Amazon EC2 インスタンスが営業時間外に停止されているかどうかを確認するには、インスタンスの起動時間と停止時間を監査します。

管理アカウント内のResource Groups & Tag Editor には、組織内のすべての連結アカウントのタグポリシーコンプライアンスを分析する追加機能があります。

タグ付けの有効性を測定した結果に基づいて、ユースケース定義、タグ付けスキーマの実装や強制などのステップで改善や変更が必要かどうかを確認します。必要な変更を加え、望ましい効果が得られるまでこのサイクルを繰り返します。コスト属性を使った例では、改善率を確認できます。

リソースに実際にタグを付ける必要があるのは開発者と運用者なので、彼らに所有権を与えることが重要です。AWS 導入の過程でチームに一般に課せられる新しい責任はこれだけではありません。アプリケーションの開発と運用のセキュリティとコストに対する責任の増加も重要です。組織では、新しいプラクティス採用の動機付けの手段として目標やターゲットが使用されることが多いので、ここでも同じことが言えます。

タグ付けのユースケース

トピック

コスト配分と財務管理のタグ

組織が最初に取り組むタグ付けのユースケースの 1 つは、コストと使用量の可視化と管理です。これには通常、次のようないくつかの理由があります。

- 一般に、このシナリオは誰もが理解していることであり、要件は明らかです。たとえば、財務チームのニーズは、複数のサービス、機能、アカウント、またはチームにまたがるワークロードとインフラストラクチャの総コストを確認することです。このようなコストの可視化を実現する 1 つの方法として、リソースに一貫したタグ付けをします。
- タグとその値は明確に定義されています。通常、コスト配分メカニズムは、コストセンター、事業単位、チーム、または組織機能ごとの追跡など、組織の財務システムに既にあるメカニズムです。
- 迅速で実証可能な投資収益率。例えば、適切なサイズに設定されたリソース、自動スケーリングされたリソース、スケジュール設定されたリソースなど、リソースに一貫したタグが付けられていると、コスト最適化の傾向を経時的に追跡できます。

AWS でコストがどのように発生するかを理解することで、情報に基づいた財務上の意思決定を行うことができます。リソース、ワークロード、チーム、または組織レベルでコストが発生した場所がわかれば、達成したビジネス成果と比較して、該当するレベルでもたらされる価値をより深く理解できます。

エンジニアリングチームには、リソースの財務管理の経験がない場合もあります。AWS 財務管理の専門スキルがあり、エンジニアリングチームと開発チームに対して AWS 財務管理の基本をトレーニングし、財務とエンジニアリングの関係を築いて FinOps の文化を育むことができる人を配置すれば、ビジネスに測定可能な成果が得られ、コストを念頭に置いて構築することがチームに奨励されます。優れた財務慣行の確立については、Well-Architected フレームワークの「コスト最適化の柱」で詳しく説明していますが、このホワイトペーパーではいくつかの基本原則に触れます。

コスト配分タグ

コスト配分とは、発生したコストを、定められたプロセスを経て、そのコストの利用者または受益者に割り当て、あるいは配分することです。このホワイトペーパーでは、コスト配分をショーバックとチャージバックの 2 種類に分けています。

ショーバックツールとメカニズムは、コスト意識の向上に便利です。チャージバックはコスト回収に役立ち、コスト意識の向上に便利です。ショーバックとは、ビジネスユニット、アプリケーション、ユーザー、コストセンターなど、特定のエンティティが負担する料金の表示、計算、レポートに関する概念です。例:「インフラストラクチャエンジニアリングチームは先月 AWS を X ドル使用しました」。チャージバックとは、財務システムや仕訳伝票などの組織の内部会計プロセスを通じて、発生した費用をそれらの事業体に実際に請求することです。例:「インフラストラクチャエンジニアリングチームの AWS の予算から X ドルが差し引かれました。」 いずれの場合も、リソースに適切にタグを付けると、エンティティへのコスト配分に便利です。唯一の違いは、誰かが支払いを求められるかどうかです。

組織の財務ガバナンスでは、アプリケーション、ビジネスユニット、コストセンター、チームレベルで発生するコストの透明性の高い会計処理が求められる場合があります。コスト配分タグでサポートされているコスト属性を実行すると、適切にタグ付けされたリソースに対してエンティティが発生させたコストを正確に関連付けるために必要なデータが得られます。

- 説明責任 - コストは必ずリソース使用量の責任者に配分してください。支出のレビューや報告の責任を負うことができるのは 1 つのサービス拠点またはグループです。
- 財務の透明性 - 経営陣向けの効果的なダッシュボードと有意義なコスト分析の作成によって、IT 部門に対する資金配分を明確に把握できます。
- 情報に基づいた IT 投資 - 目的は、プロジェクト、アプリケーション、または事業分野に基づいて ROI を追跡し、チームがより良いビジネス上の意思決定を行うことです。例えば、収益を生み出すアプリケーションにより多くの資金を割り当てるといったことが可能です。

以上のことから、コスト配分タグは次の事項の把握に便利です。

- 支出の所有者と最適化の責任者は誰か？
- 支出の原因になっているワークロード、アプリケーション、または製品は何か? どの環境またはステージか?
- 最も急速に増加しているのはどの支出分野か?
- 過去の傾向に基づいて、AWS 予算からどのくらいの支出を差し引けるか?

- 特定のワークロード、アプリケーション、または製品におけるコスト最適化の取り組みにはどのような影響があったか?

コスト配分のためにリソースタグを有効にしておくと、AWS 使用状況を可視化して支出の責任の透明性を高めるために使用できる測定方法を組織内で定義するのに便利です。それによって、コストと使用状況の可視性が適切なレベルで細分化され、コスト配分レポートや KPI 追跡を通じてクラウドの利用行動に影響を与えることもできます。

コスト配分戦略の構築

コスト配分モデルの定義と実装

AWS にデプロイするリソースのアカウント構造とコスト構造を作成します。AWS に対する支出によるコストと、そのコストがどのように発生したか、そのコストを誰が、または何が負担したかといった関連性を確立します。一般的なコスト構造は AWS Organizations、AWS アカウント、環境と組織内の事業部門や作業負荷などのエンティティに基づいています。したがって、個々のワークロードのコストをその対象となる事業部門に積み上げるなど、コストはさまざまな方法や詳細レベルで調べることができます。

想定した結果に合致するコスト構造を選択するときは、実装のしやすさと望ましい精度を対比しながらコスト配分メカニズムを評価してください。これには、説明責任、ツールの利用可能性、文化の変化に関する考慮事項が関係する場合があります。AWS 顧客は通常、次の 3 つの一般的なコスト配分モデルから開始します。

- アカウントベース - このモデルは労力が最小限で済み、ショーバックやチャージバックの精度も高く、アカウント構造が定義されている組織に適しています (また、「Organizing Your AWS Environment Using Multiple Accounts」ホワイトペーパーの推奨事項にも合っています)。これにより、アカウント毎のコストが明確に可視化されます。コストの可視化と配分についてはAWS Cost Explorer、Cost and Usage Reports だけでなく、AWSBudgets でコストの監視と追跡を行うことができます。これらのツールには、AWS アカウント 毎のフィルターやグループ化のオプションがあります。コスト配分の観点からは、このモデルが個々のリソースの正確なタグ付けに依存する必要はありません。
- ビジネスユニットまたはチームベース — 企業内のチーム、ビジネスユニット、または組織にコストを配分できます。このモデルの操作にはそこそこの手間が必要とされますが、ショーバックやチャージバックの精度も高く、さまざまなチーム、アプリケーション、ワークロードタイプに分離された明確なアカウント構造 (通常は AWS Organizations を使用) を持つ組織に適しています。これにより、チームやアプリケーション全体でコストを明確に把握でき、さらなる利点とし

て、1 回の AWS アカウント 以内にAWSサービスクォータに達するリスクが軽減されます。例えば、各チームに 5 つのアカウント (prod、staging、test、dev、sandbox) があっても、2 つのチームやアプリケーションが同じアカウントを共有することはないでしょう。このような構造では、AWSCost Categories にはアカウントやその他のタグ (「メタタギング」) をカテゴリにグループ化する機能 (「メタタギング」があり、前の例で説明したツールで追跡できます。アカウントや組織単位 (OU) に AWS Organizations でタグを付けができる点には注意が必要ですが、これらのタグはコスト配分や請求レポートには適用できません (したがって、OU ごとの AWS Cost Explorer コストのグループ化や、フィルタリングはできません)。AWSそのような目的には Cost Categories を使用してください。

- タグベース - このモデルは前の 2 つに比べて手間がかかりますが、要件や最終目標に応じてショーバックやチャージバックの精度も高くなります。「Organizing Your AWS Environment Using Multiple Accounts」ホワイトペーパーにある方法の採用を強くお勧めしますが、現実には、顧客のアカウント構造が混在した複雑な構造をしていることがあり、移行に時間がかかることはめずらしくありません。このシナリオでは、厳密で効果的なタグ付け戦略を実装することが重要であり、続いて、請求情報とコスト管理コンソールでコスト配分に関連するタグを有効化します (AWS Organizations では、コスト配分のタグは管理支払者のアカウントからのみ有効化できます)。コスト配分でタグを有効にすると、前の方法で説明したコストの可視化と配分のためのツールをショーバックやチャージバックに使用できます。コスト配分タグはさかのぼることはなく、コスト配分が有効化された後にのみ、請求レポートやコストトラッキングツールに表示されるので注意してください。

まとめると、ビジネスユニットごとにコストを追跡する必要がある場合は、AWSCost Categories で、AWS 組織内の連結アカウントを内容に従ってグループ化すれば、そのグループを請求レポートに表示できます。本番環境と非本番環境で別々のアカウントを作成すると、AWS Cost Explorerなどのツールで環境に関連するコストのフィルタリングや、Budgets を利用して、それらのコストの追跡もできます。AWS最後に、個々のワークロードやアプリケーション別など、より詳細なコスト追跡が必要なユースケースでは、それらのアカウント内のリソースに適切なタグを付け、管理アカウントでコスト配分用のタグキーを有効にすれば、請求レポートツールでそのコストをタグキーでフィルタリングできます。

コストレポートとモニタリングプロセスの確立

まず、社内の利害関係者にとって重要なコストのタイプ (たとえば、日次支出、アカウント別コスト、X 単位コスト、償却コストなど) の特定から始めます。そうすれば、AWS 請求書の確定を待つよりも早く想定外の支出や異常な支出に伴う予算上のリスクを軽減できます。タグにはこうしたレポートシナリオを可能にする属性があります。レポートから得られた情報は、財政予算における異常な想定外の支出による影響軽減のための対策に役立ちます。想定外のコスト増が発生した場合は、提

供した価値が予想よりも急増したかどうかを判定して、必要な対策が必要かどうか、またどのような対策が必要かの判断が重要です。

コスト配分に役立てるためのタグ付け戦略を策定する際は、以下の要素を念頭に置いてください。

- AWS Organizations - 複数のアカウント内でのコスト配分は、アカウント、アカウントグループ、またはそれらのアカウントのリソース用に作成したタグのグループごとに実行できます。AWS Organizations の個々のアカウントにあるリソース用に作成されたタグは、管理アカウントからのみコスト配分に使用できます。
- AWSアカウント - 1 つの AWS アカウント 内でのコスト配分は、サービスや地域などの追加ディメンションで実行できます。アカウント内のリソースにさらにタグを付け、そのようなリソースタグのグループと連携することができます。
- コスト配分タグ - 必要に応じて、ユーザーが作成したタグと AWS が生成したタグの両方を有効化してコスト配分を行うことができます。（AWS Organizations の管理アカウントの）請求コンソールでコスト配分用のタグを有効にすると、ショーバックやチャージバックに便利です。
- Cost Categories - AWS Cost Categories を使用すると、AWS 組織内のアカウントをグループ化し、タグをグループ化（「メタタギング」）できます。これにより、AWS Cost Explorer、AWS Budgets や AWS コストと使用状況レポートなどのツールで、これらのカテゴリに関連するコストをさらに分析できます。

企業内の事業部門、チーム、または組織のショーバックとチャージバックを行います。

コスト構造とコスト配分タグに裏付けられたコスト配分プロセスを使用してコストを特定します。タグは、直接費用を支払う責任はないものの、その費用を発生させた責任があるチームにショーバックを提供するために使用できます。このアプローチで、チームの支出への貢献度と、そのコストがどのように発生しているかを把握できます。コストに直接の責任があるチームにチャージバックを実施して、消費したリソースの費用を回収し、それらのコストとその発生状況を把握することができます。

KPI の効率性または値の測定と循環

ここでは、クラウド財務管理への投資の影響を測定するために、単位コストや KPI の指標について理解していただきます。この演習では、テクノロジーとビジネスの利害関係者の間で共通の言語を構築し、絶対的な総支出のみに焦点を当てたストーリーではなく、効率に基づいたストーリーをお話しします。その他の情報については、「ユニットメトリックがビジネス機能間の調整にどのように役立つか」を説明したこのブログをチェックしてください。

割り当てられない支出の配分

組織の会計方法によっては、請求の種類が異なれば処理も異なる場合があります。タグ付けできないリソースやコストカテゴリを把握してください。使用するサービスと使用予定のサービスに応じて、このような割り当て不可能な支出をどのように処理し、評価するかについてのメカニズムについて理解していただきます。例えば、AWS Resource Groups とタグエディタがサポートしているリソースのリストについては、 と『AWS Resource Groupsタグユーザーガイド』を参照してください。

タグ付けできないコストカテゴリの一般的な例としては、リザーブドインスタンス (RI) や Savings Plans (SP) などのコミットメントベースの割引にかかる手数料があります。サブスクリプション料金や未使用の SP や RI 料金は、請求レポートツールに表示される前にタグ付けすることはできませんが、表示後であれば、RI と SP の割引が AWS Organizations でアカウント、リソース、およびそれらのタグにどのように適用されるかを追跡できます。たとえば、AWS Cost Explorer では償却コストを確認すること、その支出を関連するタグキーでグループ化すること、ユースケースに関連するフィルターを適用できます。AWS コストと使用状況レポート (CUR) では、RI と SP の割引の対象になる使用量に対応する行を除外し (詳細は CUR ドキュメントの「ユースケース」セクションを参照)、自分だけに関連する列を選択することができます。コスト配分対象として有効になった各タグキーは、月次コスト配分レポートなど、他の従来の請求レポートに表示される場合と同様に、CUR レポートの最後に個別の列に表示されます。その他の参考資料としては、AWSWell-Architected Labs で CUR データからコストと使用状況を把握する例を確認してください。

レポート作成

ショーバックやチャージバックを支援する AWS ツール以外にも、タグ付けされたリソースのコストを監視し、タグ付け戦略の有効性を測定するのに役立つ、さまざまな AWS 作成済みソリューションやサードパーティソリューションがあります。それぞれの組織の要件と最終目的によっては、カスタマイズされたソリューションの構築に時間とリソースを投資するか、AWS クラウドManagement Tools Competency Partners が提供するツールを購入するかを選択できます。ビジネスに関連する制御されたパラメータを備えた独自の単一の信頼できるコスト配分ツールを作成する場合は、AWS Cost and Usage Report (CUR) を使用するとコストと使用状況に関する最も詳細なデータが得られ、カスタマイズされた最適化ダッシュボードを作成できます。これにより、アカウント、サービス、コストカテゴリ、コスト配分タグ、その他複数の要素でフィルタリングやグループ化ができます。必要であれば、AWS が開発した、これらのツールの 1 つとして使用できる CUR ベースのソリューションとして、AWS Well-Architected Labsの Web サイトのクラウドインテリジェンスダッシュボードを確認してください。

運用とサポート用のタグ

AWS 環境には、運用要件の異なる複数のアカウント、リソース、およびワークロードがあります。タグを使用すると、運用チームによるサービス管理の強化をサポートするコンテキストやガイダンスが得られます。タグは、管理対象リソースの運用ガバナンスの透明性の向上にも使用できます。

運用タグの一貫した定義の作成を促がす主な作業には、次のようなものがあります。

- 自動インフラストラクチャーアクティビティ中にリソースをフィルタリングする。例えば、リソースをデプロイ、更新、削除する場合などです。もう 1 つは、コストの最適化と時間外使用量の削減を目的としたリソースのスケーリングです。実際の例については、AWSインスタンススケジューラのソリューションを参照してください。
- 孤立したリソースや廃止予定のリソースを特定する。定義された有効期間を過ぎたリソースや、内部メカニズムによって隔離のフラグが立てられたリソースには、サポート担当者が調査できるように、適切なタグを付ける必要があります。廃止予定のリソースは、分離、アーカイブ、削除の前にタグ付けする必要があります。
- リソースグループのサポート要件。一般に、リソースにはさまざまなサポート要件があり、それらはチーム間の交渉や、アプリケーションの重要度の一部として設定することができます。運用モデルに関する詳細なガイダンスは、オペレーショナルエクセレンスの柱に掲載されています。
- インシデント管理プロセスを強化する。インシデント管理プロセスの透明性を高めるタグをリソースにタグ付けると、サポートチームやエンジニア、重大インシデント管理 (MIM) チームによるイベント管理の効果化が図れます。
- バックアップ。タグは、リソースのバックアップが必要な頻度、バックアップコピーの保存先、バックアップの復元先の特定にも使用できます。AWS でのバックアップとリカバリのアプローチに関する規範的なガイダンス。
- パッチ適用。AWS で実行中の可変インスタンスにパッチを適用することは、包括的なパッチ戦略とゼロデイ脆弱性のパッチ適用の両方において重要です。より広範なパッチ戦略に関するより詳細なガイダンスについては、『規範的ガイダンス』を参照してください。ゼロデイ脆弱性のパッチについては、このブログで説明しています。
- 運用上の可観測性。運用 KPI 戦略をリソースタグに変換しておくと、運用チームは目標が達成されているかどうかをより正確に追跡してビジネス要件を強化できます。KPI 戦略の策定は独立したトピックですが、そこでは定常的に運営されている事業や、変化の影響と成果をどこで測定すべきかに焦点が当てられる傾向があります。KPI ダッシュボード (AWS Well-Architected Labs) とオペレーション KPI ワークショップ (AWSエンタープライズサポートの事前対応型サービス) はどちらも、定常状態での測定パフォーマンスを対象としています。AWS エンタープライズ戦略のブロ

グ記事「トランスフォーメーションの成功を測定する」では、IT の近代化や、オンプレミスから AWS への移行など、トランスフォーメーションプログラムの KPI 測定が解説されています。

自動インフラストラクチャーアクティビティ

タグは、インフラストラクチャーを管理する際のさまざまな自動化アクティビティに使用できます。たとえば、AWS Systems Manager を使用すると、作成した定義済みのキーと値のペアで指定したリソースの自動化やランブックを管理できます。管理ノードには、タグのセットを定義してオペレーティングシステムや環境毎のノードの追跡やターゲット化ができます。その後、グループ内のすべてのノードに対する更新スクリプトの実行や、それらのノード状態の確認ができます。Systems Manager リソースにタグを付けて、自動化されたアクティビティの絞り込みや、追跡もできます。

環境リソースの開始と停止のライフサイクルの自動化は、どの組織にとっても大幅なコスト削減につながります。AWS 上のインスタンススケジューラーは、Amazon EC2 インスタンスと Amazon RDS インスタンスが不要なときに起動、停止できるソリューションの一例です。例えば、週末に実行する必要のない Amazon EC2 や Amazon RDS インスタンスを利用している開発者環境では、それらのインスタンスのシャットダウンによるコスト削減の可能性は活かされていません。チームとその環境のニーズを分析し、これらのリソースに適切にタグを付けて管理を自動化すれば、効果的に予算を活用できます。

Amazon EC2 インスタンスのインスタンススケジューラーが使用するスケジュールタグの例:

```
{
    "Tags": [
        {
            "Key": "Schedule",
            "ResourceId": "i-1234567890abcdef8",
            "ResourceType": "instance",
            "Value": "mon-9am-fri-5pm"
        }
    ]
}
```

ワークロードのライフサイクル

裏付けとなる運用データの正確性を確認します。ワークロードのライフサイクルに関連するタグを定期的に見直し、そのレビューに適切な利害関係者が関わっていることを確認してください。

表 7 — ワークロードライフサイクルの一環としての運用タグの見直し

ユースケース	タグキー	根拠	値の例
アカウント所有者	example-inc:account-owner:owner	アカウントの所有者とそれに含まれるリソース。	ops-center , dev-ops, app-team
アカウントオーナーのレビュー	example-inc:account-owner:review	アカウント所有権の詳細内容が最新で正しいことを確認する。	<タグ付けライブラリ内で定義されている日付が正しいフォーマットであることを確認する>
データ所有者	example-inc:data-owner:owner	アカウントにあるデータのデータ所有者。	bi-team, logistics , security
データ所有者レビュー	example-inc:data-owner:review	データ所有権の詳細が最新で正確であることを確認する。	<タグ付けライブラリ内で定義されている日付が正しいフォーマットであることを確認する>

停止 OU に移行する前に、停止アカウントにタグを割り当てる

『Organizing Your AWS Environment Using Multiple Accounts』ホワイトペーパーにあるように、アカウントを一時停止して停止 OU に移行する前に、アカウントのライフサイクルの内部追跡と監査ができるタグをアカウントに追加する必要があります。例えば、組織の ITSM チケットシステム上の相対 URL やチケット参照には、一時停止中のアプリケーションの監査記録が表示されます。

表 8 - ワークロードのライフサイクルが新たな段階に入った際の運用タグの追加

ユースケース	タグキー	根拠	値の例
アカウント所有者	example-inc:account-owner:owner	アカウントの所有者とそれに含まれるリソース。	ops-center , dev-ops, app-team

ユースケース	タグキー	根拠	値の例
データ所有者	example-inc:data-owner:owner	アカウントにあるデータのデータ所有者。	bi-team, logistics , security
停止日	example-inc:suspension:date	アカウントが停止された日付	<タグ付けライブラリ内で定義された正しいフォーマットの停止日>
一時停止の承認	example-inc:suspension:approval	アカウント停止の承認へのリンク	workload/deprecation

インシデント管理

タグは、インシデントの記録から、優先順位付け、調査、コミュニケーション、解決、終了に至るまで、インシデント管理のあらゆる段階で重要な役割を果たします。

タグには、インシデントを記録すべき場所、インシデントについて知らせるべきチーム、定義済みのエスカレーション優先度を詳細に記述できます。タグは暗号化されないので、タグにどのような情報を保存するかをよく考えてください。また、組織、チーム、報告部門では責任の所在が移り変わるので、情報をより効果的に管理できる安全なポータルへのリンクを保存することを検討してください。これらのタグは排他的である必要はありません。例えば、アプリケーション ID で、IT サービス管理ポータルのエスカレーションパスを検索できます。運用上の定義では、このタグが複数の目的で使用されていることを明確にしてください。

インシデントマネージャーや運用担当者がインシデントやイベントに対応して目標を絞り込むのに役立つように、運用要件タグも詳細に記述できます。

ランブックやプレイブックの (ナレッジシステムのベース URL への) 相対リンクをタグとして含めると、対応チームが対応するプロセス、手順、文書を特定しやすくなります。

表 9 - 運用タグをインシデント管理の情報伝達に使用する

ユースケース	タグキー	根拠	値の例
インシデント管理	example-inc:incident-management:escalationlog	サポートチームがインシデントを記録するために使用しているシステム	jira, servicenow , zendesk
インシデント管理	example-inc:incident-management:escalationpath	エスカレーションの経路	ops-center , dev-ops, app-team
コスト配分とインシデント管理	example-inc:cost-allocation:CostCenter	コストセンターごとにコストを監視します。これは、コストセンターをインシデント記録のアプリケーションコードとして使用するデュアルユースタグの例です。	123-*
バックアップスケジュール	example-inc:backup:schedule	リソースのバックアップスケジュール	Daily
プレイブック/インシデント管理	example-inc:incident-management:playbook	文書化されたプレイブック	webapp/incident/playbook

パッチ適用

組織で、AWS Systems Manager Patch Manager と AWS Lambda を使用すれば、可変コンピューティング環境へのパッチ適用戦略を自動化し、可変インスタンスをそのアプリケーション環境の定義済みパッチベースライに一致させることができます。これらの環境内の可変インスタンスのタグ付け戦略は、そのインスタンスをパッチグループとメンテナンスウィンドウに割り当てれば管理できま

す。Dev → Test → Prod の分割については、以下の例を参照してください。 可変インスタンスのパッチ管理については、AWS の規範的なガイダンスが用意されています。

表 10 - 環境によって異なる運用タグ

開発	ステージング	本番稼働用
{ "Tags": [{ "Key": "Maintenance Window", "ResourceId": "i-012345678ab9ab111", "ResourceType": "instance", "Value": "cron(30 23 ? * TUE#1 *)" }, { "Key": "Name", "ResourceId": "i-012345678ab9ab222", "ResourceType": "instance", "Value": "WEBAPP" }, { "Key": "Patch Group", "ResourceId": "i-012345678ab9ab333", "ResourceType": "instance", "Value": "WEBAPP-DEV-AL2" }] }	{ "Tags": [{ "Key": "Maintenance Window", "ResourceId": "i-012345678ab9ab444", "ResourceType": "instance", "Value": "cron(30 23 ? * TUE#2 *)" }, { "Key": "Name", "ResourceId": "i-012345678ab9ab555", "ResourceType": "instance", "Value": "WEBAPP" }, { "Key": "Patch Group", "ResourceId": "i-012345678ab9ab666", "ResourceType": "instance", "Value": "WEBAPP-TEST-AL2" }] }	{ "Tags": [{ "Key": "Maintenance Window", "ResourceId": "i-012345678ab9ab777", "ResourceType": "instance", "Value": "cron(30 23 ? * TUE#3 *)" }, { "Key": "Name", "ResourceId": "i-012345678ab9ab888", "ResourceType": "instance", "Value": "WEBAPP" }, { "Key": "Patch Group", "ResourceId": "i-012345678ab9ab999", "ResourceType": "instance", "Value": "WEBAPP-PROD-AL2" }] }

ゼロデイ脆弱性は、パッチ戦略を補完するタグの定義によっても管理できます。詳細なガイダンスについては、「AWS Systems Manager による同日のセキュリティパッチ適用によるゼロデイ脆弱性の回避」を参照してください。

運用上のオブザーバビリティ

環境パフォーマンスに対して実行可能な見通しを立て、問題の検出と調査に役立てるには、オブザーバビリティが必要です。また、重要業績評価指標 (KPI) や稼働時間などのサービスレベル目標 (SLO) を定義して測定できるという副次的な目的もあります。ほとんどの組織にとっては、インシデント発生時の平均検出時間 (MTTD) とインシデントからの平均回復時間 (MTTR) が重要な運用 KPI です。

オブザーバビリティでは、データが収集されてから関連するタグが収集されるため、コンテキストが重要です。対象になるサービス、アプリケーション、またはアプリケーション層に関係なく、その特定のデータセットを絞り込んで分析できます。CloudWatch Alarms へのオンボーディングをタグで自動化できるため、特定のメトリクスのしきい値を超えた場合に適切なチームがアラートを受け取ることができます。例えば、タグキー `example-inc:ops:alarm-tag` とその値が CloudWatch アラームの作成を示している可能性があります。これを示すソリューションについては、「Amazon EC2 インスタ–スに対する Amazon CloudWatch アラームの作成と維持のためにタグを使用する」を参照してください。

設定したアラームの数が多すぎると、オペレーターが個々のアラームを手動で重要度を判定して優先順位付けをしている間に、大量のアラームや通知でオペレーターに負担がかかり、全体的な効果を低下して、アラートストームが発生しやすくなります。アラームの追加コンテキストはタグの形で提供でき、Amazon EventBridge 内でルールを定義できるため、ダウンストリームの依存関係ではなくアップストリームの問題に的を絞ることができます。

DevOps と並行して運用が果たす役割は見過ごされがちですが、多くの組織では、中央運用チームが依然として通常の営業時間外に重要な初対応を行っています。(このモデルの詳細については、「オペレーショナルエクセレンスのホワイトペーパー」を参照してください。) ワークロードを担当する DevOps チームとは異なり、通常、運用チームはそれほど深い知識を持っていないため、タグでダッシュボードやアラートで得られるコンテキストを基に、問題の正しいランブックへの誘導や、自動ランブックの開始ができます (ブログ記事「Automating Amazon CloudWatch Alarmswith AWS Systems Manager」を参照)。

データセキュリティ、リスク管理、アクセス制御用のタグ

データの保存と処理の適切な取り扱いについて、組織にはさまざまなニーズや責任があります。データ分類は、アクセス制御、データ保持、データ分析、コンプライアンスなど、いくつかのユースケースでは、重要な前提条件です。

データセキュリティとリスク管理

AWS 環境内には、コンプライアンス要件やセキュリティ要件が異なるアカウントが存在する可能性があります。例えば、開発者用サンドボックスと、支払い処理などの規制の厳しいワークロード用の本番環境をホストするアカウントがあるとします。それらを異なるアカウントに分離して、個別のセキュリティ制御を適用し、機密データへのアクセスを制限すれば、規制対象のワークロードの監査範囲を狭めることができます。

すべてのワークロードに同じ標準を採用することは、問題の発生につながるおそれがあります。多くの制御機能が環境全体に等しく適用されていますが、特定の制御機能の枠組みを満たす必要のないアカウントや、個人を特定できるデータが存在しないアカウント (開発者用サンドボックスやワークロード開発アカウントなど) には、その存在が過剰な制御や無関係な制御もあります。そのような状況は、一般に、優先順位を付けて何もしないでクローズしなければならないセキュリティ上の誤検出につながるため、本来の調査すべき発見に費やされる労力が無駄に奪われます。

表 11 — データセキュリティタグとリスク管理タグの例

ユースケース	タグキー	根拠	値の例
インシデント管理	`example-inc:incident-management:escalationlog`	サポートチームがインシデントを記録するために使用しているシステム	`jira`, `servicenow` , `zendesk`
インシデント管理	`example-inc:incident-management:escalationpath`	エスカレーションの経路	`ops-center` , `dev-ops`, `app-team`
データ分類	`example-inc:data:classification`	コンプライアンスとガバナンスのためのデータの分類	`Public`, `Private`, `Confidential` , `Restricted`
コンプライアンス	`example-inc:compliance:framework`	ワークロードが対象となるコンプライアンスフレームワークを特定します。	`PCI-DSS`, `HIPAA`

AWS 環境全体でさまざまな制御を手動で管理するのは時間がかかり、エラーも発生しやすくなります。次のステップでは、適切なセキュリティ制御の導入を自動化し、そのアカウントの分類に基づいてリソースの検査を設定します。アカウントとその中のリソースにタグを適用すると、制御の導入を自動化し、ワークロードに合わせて適切に設定できます。

例:

ワークロードには、値 `Private` のタグ `example-inc:data:classification` が付いた Amazon S3 バケットが含まれます。このセキュリティツールの自動化では AWS Config ルール `s3-bucket-public-read-prohibited` がデプロイされます。このルールでは、Amazon S3 バケットの ブロックパブリックアクセス設定、バケットポリシー、バケットアクセスコントロールリスト (ACL) が確認され、バケットの構成がデータ分類に対して適切かどうかを確認できます。バケットの内容が分類と一致していることを確認するために、Amazon Macie は個人識別情報 (PII) をチェックするように設定できます。ブログ「Amazon Macie を使用して S3 バケットデータ分類を検証する」では、このパターンについて詳しく説明しています。

保険や医療などの特定の規制環境では、必須のデータ保持ポリシーが適用される場合があります。タグを使用したデータ保持と Amazon S3 ライフサイクルポリシーを組み合わせると、オブジェクトを別のストレージ階層で調べるための効果的で簡単な方法が得られます。Amazon S3 ライフサイクルルールで、必須の保持期間の終了後にデータを削除するオブジェクトを期限切れにすることもできます。このプロセスの詳細なガイドについては、「Amazon S3 ライフサイクルでオブジェクトタグを使用してデータライフサイクルを簡素化する」を参照してください。

さらに、セキュリティ上の検出結果の優先順位付けや対処を行う際に、調査担当者にはタグでリスクの見極めに役立つ重要なコンテキストが得られ、適切なチームを関わらせて調査結果や軽減に役立てることができます。

ID 管理とアクセス制御用のタグ

AWS IAM Identity Center で AWS 環境全体でアクセス制御を管理するとき、タグを使用すれば、スケーリングで複数のパターン化が可能になります。適用できる委任パターンはいくつかあり、中にはタグ付けに基づくものもあります。それぞれの説明と、それらを参照するためのリンクを用意しました。

個々のリソースの ABAC

IAM Identity Center のユーザーと IAM ロールは、属性ベースのアクセス制御 (ABAC) をサポートし、タグに基づいて操作とリソースへのアクセスを定義します。ABAC で、権限ポリシーを更新する必要性が減り、社内ディレクトリで従業員属性からベースアクセスを行うのに便利です。すでにマル

チアカウント戦略を採用している場合は、ロールベースのアクセス制御 (RBAC) に加えて ABAC を使用すれば、同じアカウントで業務を行う複数のチームに、さまざまなリソースに対するきめ細かいアクセス権限を与えることができます。例えば、IAM Identity Center のユーザーまたは IAM ロールには、特定の Amazon EC2 インスタンスへのアクセス権限を制限する条件を含めることができます。そうしない場合、それらのインスタンスにアクセスするためには各ポリシーに明示的に記載する必要があります。

ABAC 認証モデルはオペレーションやリソースへのアクセス権限をタグに依存するため、想定外のアクセスを防ぐためのガードレールを設けることが重要です。SCP では、特定の条件下でのみタグの変更を許可して、組織全体のタグを保護するために使用できます。実装方法についてはブログ「Seecuring resource tags used for authorization using a service control policy in AWS Organizations」と「Permissions boundaries for IAM entities」を参照してください。

運用期間の長い Amazon EC2 インスタンスで古くからの運用方法をサポートしている場合、このアプローチを利用できます。ブログ「Configure IAM Identity Center ABAC for Amazon EC2 instances and Systems Manager Session Manager」では、この形式の属性ベースのアクセス制御についてさらに詳しく説明しています。前述のように、すべてのリソースタイプがタグ付けをサポートしているわけではなく、サポートしているリソースタイプの中には、すべてのリソースタイプがタグポリシーを使用した強制をサポートしているわけではないため、この戦略を AWS アカウント に実装する前に、これを評価することをお勧めします。

ABAC をサポートするためにサービスについては、「IAM と動作する AWS サービス」を参照してください。

結論

AWS リソースには、コスト配分戦略の実装から自動化のサポート、AWS リソースへのアクセスの承認まで、さまざまな目的でタグを付けることができます。タグ付け戦略の実装は、関係するステークホルダーグループの数が多く、データソーシングやタグガバナンスなどの考慮事項があるため、一部の組織にとっては簡単ではありません。

このホワイトペーパーでは、運用慣行、定義済みのユースケース、プロセスに関与する利害関係者、AWS によって提供されるツールやサービスに基づいて、組織におけるタグ付け戦略の設計と実施に関する推奨事項の概要を述べました。タグ付け戦略は、反復と改善のプロセスです。当面の優先事項から小さく始めて、組織全体で関連するユースケースを特定し、必要に応じてタグ付けスキーマを実装して拡張し、効果を継続的に測定して改善します。すでに指摘したように、組織内でタグを明確に定義しておくと、組織の戦略と価値に合わせて、そのタグが存在するリソースとビジネス目的を担当するチームに AWS の用途や使用状況を関連付けることができます。

寄稿者

本ドキュメントの寄稿者は次のとおりです。

- クリス・ペイツ、Amazon Web Services、シニアスペシャリストテクニカルアカウントマネージャー
- ビジェイ・シェカール・ラオ、Amazon Web Services、エンタープライズサポートリード
- ナタリヤ・ゴドゥノック、Amazon Web Services、シニアスペシャリストテクニカルアカウントマネージャー
- Yogish Kutkunje Pai、Amazon Internet Services、シニアソリューションアーキテクト
- ジェイミー・イブ、Amazon Web Services、シニアスペシャリストテクニカルアカウントマネージャー

詳細情報

詳細については、以下を参照してください。

- AWS re:Invent 2020: Working backwards: Amazon's approach to innovation
- AWS 規範ガイダンス: AWS Systems Manager によるハイブリッドクラウドにおける可変インスタンスへの自動パッチ適用
- AWS アーキテクチャセンター

AWS Well-Architected

- AWS Well-Architected フレームワーク
- オペレーショナルエクセレンスの柱 - AWS Well-Architected フレームワーク
- ディザスタリカバリ (DR) 計画 - AWS Well-Architected 信頼性の柱
- コスト最適化の柱 - AWS Well-Architected フレームワーク
- AWSWell-Architected ラボ:AWS-生成されたコスト配分タグを有効にする
- AWSWell-Architected ラボ:タグポリシー
- AWSWell-Architected ラボ:AWSCUR クエリライブラリ

AWSブログ

- AWS HealthAware - 組織アカウントと個人 AWS アカウントの AWS Health アラートをカスタマイズ
- How to Automatically Tag Amazon EC2 Resources in Response to API Events
- AWS生成 vs ユーザー定義のコスト配分タグ
- AWS Organizations によるコストタグ付けとレポート
- Patching your Windows EC2 instances using AWS Systems Manager Patch Manager
- Avoid zero-day vulnerabilities with same-day security patching using AWS Systems Manager

AWS ドキュメント

- コスト配分タグの使用-AWS Billing and Cost Management コスト管理とコスト管理
- AWS コストと使用状況レポートとは

- AWS Resource Groups API リファレンス
- How can I use IAM policy tags to restrict how an EC2 instance or EBS volume can be created?
- Mutable vs immutable update models

その他

- Bryar, C. and Carr, B. (2021). Working Backwards: Insights, Stories, and Secrets from Inside Amazon. London Macmillan.
- AWS CloudFormationガード (GitHub)

ドキュメントの改訂

このホワイトペーパーの更新に関する通知を受け取るには、RSS フィードにサブスクライブしてください。

変更	説明	日付
マイナーな更新	ID 管理の更新	2023 年 3 月 30 日
マイナーな改訂	ABACの個々のリソースのリファレンスを更新しました。	2023 年 2 月 24 日
マイナーな改訂	IAM のベストプラクティスに合わせてガイドを更新しました。詳細については、「IAM のセキュリティのベストプラクティス」を参照してください。	2023 年 2 月 6 日
主な改訂	AWS Config ルール `required_tags` でサポートされるリソースタイプに関するより具体的なリファレンスを追加しました。	2023 年 1 月 18 日
主な改訂	特にアイデンティティの分野における最新のプラクティスとサービス機能を新たに追加して更新しました。	2022 年 9 月 29 日
マイナーな更新	PDF 版の表フォーマットを修正しました。	2022 年 4 月 25 日
主な改訂	文書構造を更新し、「タグ付け戦略」と「ユースケース」セクションを拡張しました。最新のツール、手法、利用可能なリソースに基づいた、よ	2022 年 4 月 22 日

	り規範的なガイダンスを追加しました。	
初版発行	ホワイトペーパーの初回発行。	2018 年 12 月 1 日

> **Note**
>
> RSS の更新をサブスクライブするには、使用しているブラウザで RSS プラグインを有効にする必要があります。

注意

お客様は、本書に記載されている情報を独自に評価する責任を負うものとします。本書は、(a) 情報提供のみを目的とし、(b) AWS の現行製品と慣行について説明しており、これらは予告なしに変更されることがあり、(c) AWS およびその関連会社、サプライヤー、またはライセンサーからの契約上の義務や保証をもたらすものではありません。AWS の製品やサービスは、明示または黙示を問わず、一切の保証、表明、条件なしに「現状のまま」提供されます。お客様に対する AWS の責任は AWS 契約によって規定されます。本書は、AWS とお客様との間で締結されるいかなる契約の一部でもなく、その内容を修正するものでもありません。

AWS 用語集

AWS の最新の用語については、「AWS の用語集リファレンス」の「AWS 用語集」を参照してください。