アジェンダは、リソースの依存関係、動的参照、ネストされたスタック、クロスアカウント参照、スタック設置の 5 つです。早速、リソースの依存関係についてご説明いたします。 AWS クラウドフォーメーションを使用してテンプレートに記述したリソースを作成する際、あるリソースには 1 つ以上のリソースに依存する場合があります。これをリソースの依存関係といいます。なお、一般的にリソースの構築順はテンプレートで定義したリソース間の依存関係からクラウドフォーメーションが自動的に決定します。例えば、 EC インスタンスは、 EC インスタンスに使用するセキュリティグループに依存します。クラウドフォーメーションスタックにおいて、 EC インスタンスがセキュリティグループを参照するように記述することで、最初にセキュリティグループが作成され、次に EC インスタンスが作成され。

すまたテンプレートで定義したリソース間に依存関係がない場合、クラウドフォーメーションはリソースの作成を並行して開始します。なお、デペンジョン属性または組み込み関数を利用することで、リソースの依存関係を指定することが可能です。ディペンゾーン属性を使用してリソースの作成順序を定義した場合、クラウドフォーメーションはリソースを定義された順序で処理します。うん。また、 if 関数、 getat 関数などの組み込み関数を使用して、プロパティの値として別のリソースを参照すると、参照先のリソースから作成することが可能です。

では、ディペンゾーン特性及び組み込み関数によるリソースの依存関係について、もう少し詳しく見ていきましょう。まずはじめに、ディペンジョン属性による明示的な依存関係についてご説明いたします。 dipenzon 属性を使用することで、明示的にリソースの依存関係を定義することが可能です。たとえば、左下のテンプレートでは、上にエスリーバケットを、下に SNS トピックを定義しています。うん。

また、 SNS トピックのディベンゾーン属性にはエススリーバケットを定義しています。そうすることで、右下のイベントのように、クラウドフォーメーションスタックはエスリーバケットの作成が完了するのを待ってから、 SNS トピックの作成を開始します。

次に、組込み関数による暗黙的な依存関係についてご説明します。 liff 関数や getat 関数などの組み込み関数を使用して、別のリソースを参照することで、暗黙的に依存関係を定義することが可能です。まずは if 関数について、たとえを用いて説明いたします。AWS、SNS、サブスクリプションリソースタイプを作成する場合、プロパティのトピックアーンには、サブスクライブするトピックの Amazon リソースネームが必要となります。その際に、 DIF 関数を使用して AWS SNS トピックリソースタイプを参照すると、 SNS トピックの Amazon リソースネームを取得することが可能です。なお、クラウドフォーメーションは SIF 関数で Amazon リソース電話を取得するために SNS トピックの作成が完了するのを待ってから、 SNS トピックサブスクリプションの作成を開始します。このように lif 関数を用いることで、暗黙的にリソースの作成順序を定義することが可能です。うん。なお、スタックを削除する場合には、削除される順序が作成される順序とは逆になります。今回の場合では、最後に作成された SNS トピック、サブスクリプションが最初に削除されます。その後に SNS トピックが削除される運びとなります。

次にGET、 AT 関数について例を用いてご説明いたします。 getat 関数も if 関数同様に、参照されるリソースがクリエイトコンプリートステータスになってから、参照するリソースの作成が開始されます。はい。例えば AWS EC セキュリティグループイングレスリソースタイプを作成する場合には、プロパティのグループ ID にセキュリティグループの ID が必要となります。その際に、 AWS EC ツールセキュリティグループリソースタイプは、 GET AT 関数を使用してグループ ID 特性を参照すると、セキュリティグループの ID を参照することが可能です。なお、クラウドフォーメーションは getat 関数でセキュリティグループの ID を縮するために、セキュリティグループの作成が完了するのを待ってから、セキュリティグループイングレスの作成を開始します。はい、このように getat 関数を用いることでも、暗目的にリソースの作成順序を定義することが可能です。なお、スタックを削除する場合には、 LIF 関数と同様に削除される順序が作成される順番と逆になります。うん。今回の場合では最後に作成されたセキュリティグループイングレスが最初に削除されます。その後にセキュリティグループが削除される運びとなります。

次に動的参照についてご説明いたします。クラウドフォーメーションの動的な参照を利用することで、 AWS システムズマネージャー、パラメーターストア、 AWS シークレッツ・マネージャーを含む AWS サービスに保存されている値を。参照することが可能です。はい。 AWS システムズマネージャーでは、パラメーターストアに格納されている平文で保存されているデータまたは暗号化されているデータを参照することが可能です。一方、 AWS シークレッツマネージャーでは、保存されているシークレット情報を参照することが可能です。うん。

それでは AWS システムズマネージャーおよび AWS シークレッツマネージャーにて動的参照を行う場合の考慮事項などについて見ていきましょう。まずは、 SSM パラメータストアにて保存されている平文のデータを自動的参照する場合についてご説明いたします。はい、例では、 SSM の動的な参照を使用して、 SD バケットのアクセス制御を SSM パラメーターストアに格納されているパラメータ値に設定します。

今回のテンプレートでは、クラウドフォーメーションはスタック操作および変更設定。操作にバージョン 2 のエスリーアクセスコントロールパラメータを使用します。また、特出する考慮事項といたしましては、 4 点ございます。中でも、初めに記載しておりますように、 SSM のパラメータ内容を変更した場合には、クラウドフォーメーションにパラメータの変更を反映させるためにスタックの更新が必要になること。および次に記載しておりますように、バージョンを指定しない場合、スタックを再作成・更新するたびに最新バージョンのパラメータを使用することにご留意ください。

次に、 SSM パラメータストアにて暗号化され保存されているデータを動的参照する場合についてご説明いたします。はい。例では IAM ユーザーのパスワードを SSM パラメータストアに格納されている安全の文字列に設定しています。うん。今回のテンプレートでは、クラウドフォーメーションはスタック操作および変更セット操作にバージョン中のIM、ユーザーパスワードパラメータを使用します。特筆する考慮事項といたしましては、 4 点ございます。また、記載がございますように、 SSM セキュアストリングパラメーターはサポートしているリソースプロパティに対してのみ使用可能となっております。こちらが 2023 年 11 月現在でサポートしているリソースの一覧でございます。

最後に AWS シークレッツマネージャーのシークレットを動的に参照する方法についてご説明いたします。はい。まずはじめに、 AWS シークレットマネージャーの場合には、 AWS システムズマネージャーとは違い、シークレットID、シークレットの値、キーバージョン、ステージ、および。バージョン ID を含む複数のセグメントで構成されています。うん。

なお、バージョンステージおよびバージョン ID を使用しない場合には、デフォルトで AWS カレントというバージョンが指定されます。それでは、同様に d を用いて AWS シークレッツマネージャーについて見ていきましょう。例では、 Myrds シークレットシークレットに保存されているユーザー名、パスワードの値を取得するために、シークレット ID セグメントと JSON キーセグメントを使用しています。なお、今回のテンプレートではバージョンステージ及びバージョン ID を指定していないため、取得されるシークレットのバージョンは AWS カレントの値となります。

次はネストされたスタックについてご説明いたします。ネストされたスタックを使用することで、大きいクラウドフォーメーションテンプレートを再利用可能な小さなテンプレートに分割することができます。そして、他のテンプレート内から分解した小さなテンプレートを参集することで、スタックを作成することが可能です。うん。なお、ネストされたスタックに対しても、他のスタックをネストすることが可能です。

次にネストサイトスタックにて使用される用語について説明します。リストされたすべてのスタックが属している最上位のスタックをルートスタックと呼びます。また、ネストされたスタックにはそれぞれ直接の親スタックが存在します。図を用いて説明いたしますと、 Stack B から見ると、 Stack A は親スタックであると同時に、ルートスタックでもあります。うん。また、スタック E から見ると、 Stack D が親スタックとなります。

具体的なニュースケースとしては、複数のスタックに使用しているリソースの構成がある場合、同じ構成のテンプレートを複数用意するのではなく、うん、使用している縫製の専用のテンプレートを作成し、再利用することが考えられます。また、大きなテンプレートを分割するため、リソース数の制限を回避することも可能です。

どう REST された Stack のテンプレートをあらかじめエススリーバケットに保存しておく必要があることにご留意ください。例えば、図にあるように、VPC、IAM。それぞれのリソースを作成する小さなテンプレートを用意し、あらかじめエスリーバケットに保存しておくことで、親となるスタックの一部のスタックとして作成することができます。それでは、親スタックのテンプレートにどのようにネストされたスタックを参照するのか見ていきましょう。まずは、 AWS クラウドフォーメーションスタックというソースタイプにてコスタックを定義する必要があります。うん。そして、テンプレート URL プロパティにて、レスリンにあらかじめ保存しておいたテストされた Stack のテンプレートの URL を指定します。または、パラメーターズプロパティを使用することで、ネストされた Stack のテンプレートにパラメータを渡すことが可能です。うん。

なお、 GET AT 関数を利用することで、他のリストされたスタックのテンプレートのアウトプットセクションで出力した値を参照することもできます。続いて、クロスタック参照についてご説明します。 Crodsstack 参照は、ある Stack でデプロイしたリソースを他のスタックで参照したい場合に使用される機能です。うん。なお、スタックを分けることで、スタックの役割やスタックの管理責任などを分けることにもつながります。例えば、 1 室インスタンスを作成したい場合に、セキュリティチームが管理する IOM 用のスタック。うん。ネットワークチームが管理する VPC 用のスタックを参照するといったことが可能です。ただし、クロススタック参照を利用する上で 3 つほど留意点がございます。 1 つ目に、スタックを分けて管理することが可能ですが、その分管理が煩雑になる可能性がございます。うん。

次に別のスタックによって参照されている出力値は、変更を削除することができません。最後に、スタックの出力値が参照されている場合、そのスタックを削除することができません。それでは早速クロスタック参照の使い方について見ていきましょう。まず、参照したいリソースを持っている参照される側のスタックでは、アウトプットセクションのエクスポートフィールドに値を指定します。次に参照する側のスタックでは、組み込み関数、インポートバリュー関数にて取得したい値を指定することで参照することが可能です。うん。

さて、これまでテストされたスタックおよびクラスタック参照についてご説明いたしました。ここで、一度ネストされたスタックとクロススタック参照の違いについて確認してみましょう。テストされたスタックは、リストされたスタックのグループ内でのみパラメータなどの情報を共有したい場合に使用することを推奨しております。うん。一方で、クロスタック参照では、テストされたスタックのグループ内に限らず、他のスタックと情報を共有したい場合に使用することを推奨しております。

うん、では最後にスタックセットについて見ていきましょう。まず、スタック設置の概要といたしましては、 1 つのテンプレートを使用して複数の AWS アカウント、複数のリージョンに対して同様のスタックを作成・更新・削除することが可能な機能です。うん。

次に、スタック設置でよく使用される用語を 4 つご説明いたします。 1 つ目に管理者アカウント、これはスタックセットを作成する AWS アカウントです。 2 つ目にターゲットアカウント。こちらはスタックセットの 1 つ以上のスタックを作成、更新、削除する AWS アカウントです。次に、スタックセットは、スタックの作成に使用するテンプレート及びパラメータースタックを作成する。ターゲットアカウントリージョンドなどの定義を指しています。

最後にスタックインスタンス。これはターゲットアカウントの Stack へのリファレンスです。スタックがない場合にも存在することが可能となっております。なお、スタックインスタンスは 1 つのスタックセットにのみ関連付けられます。ここからは、スタック設置のオペレーションについてご案内いたします。まず、セルフマネージド型またはサービスマネージド型どちらのアクセス許可モデルとするかを選択し、初回のみ設定を行う必要があります。セルフマネージドでは、管理アカウント、ターゲットアカウント双方に指定された名称・内容で IAM ロールを手動で作成する必要があります。なお、 IAM ロール作成用のクラウドフォーメーションテンプレートが配布されておりますので、必要に応じてご利用ください。一方、サービスマネージドの場合には、 AWS Organizations で Stacksets の信頼されたアクセスを有効化することで、管理アカウントとターゲットアカウントの双方に必要な IAM ロールが自動的に作成されます。

次のオペレーションは、スタックセットの作成です。 Stack の作成に使用するテンプレート、ターゲット、アカウント、リージョンの指定を行うことで、スタックセットを作成できます。スタックセットの作成は大きく分けると 4 つのステップがあります。 1 つ目はテンプレートの作成ですまず、初めにサービスマネージドもしくはセルフマネージドどちらのアクセス許可にするかを選択いただきます。そして、 Stack のテンプレートを指定します。次にスタックセットの詳細を指定します。そして、スタックセットオプションにてタグや実行設定を設定します。最後にデプロイオプションにて、デプロイ先やリージョン、自動デプロイオプションなどを設定します。ここで自動デプロイオプションについて解説させていただきます。

自動デプラグオプションとは、 AWS Organizations の組織単位へアカウントを追加・削除した際の Stack インスタンスの挙動を設定する機能です。なお、自動デプロイオプションは、サービスマネージドのアクセス許可モデルの場合のみ設定することが可能です。

そう、加えて、後ほど説明するスタック設図のスタックに上書きしたパラメーターは、設定時に指定されたアカウントのみ適用され、自動デプロイオプションで追加されるアカウントには適用され。ないことをご留意ください。次に、デプロイオプションについてご説明いたします。まずはじめに、同時アカウントの最大数はスタックセットの作成、更新、削除に適用され、リージョン別に一度にオペレーションを実行するターゲットアカウントの最大数または割合を指定することが可能です。例えば、割合 50% で 2 つのリジオンの 10 個のターゲットアカウントにデプロイする場合を見てみましょう。うん。 10 個のターゲットアカウントの 50% のため、一度にオペレーションが実行されるターゲットのアカウント数は 5 つとなります。そして、リージョン別に実施されるため、図のような流れになります。

まずはじめに、リージョン A で 5 つのアカウント。次に同じリージョン A でその他の 5 つのアカウント。そのあとにリージョン B で 5 つのアカウント。最後に T ション B で、その他 5 つのアカウントといった形です。

次に、デプロイオプションの障害体制についてご説明いたします。こちらは、スタックセットの作成、更新、削除時に、各リージョンでスタックオペレーションの失敗を許容するターゲットアカウント数を設定することが可能です。例えば、割合 20% で 3 つのリージョンの 10 個のターゲットアカウントにデプロイした場合を想定してみましょう。うん、まずリージョン A ではデプロイに 1 つのターゲットアカウントで失敗しました。その場合は割合が 10% のため、次のリージョンへとストラックのオペレーションが進みます。ます。その次に、リージョンビリーでは、 3 つのターゲットアカウントでデプロイに失敗しました。その場合には、割合が 30% となり、指定した 20% を超えるため、アタックオペレーションは自動的に停止します。

最後に同時実行モードについてご説明いたします。同時実行モードは、スタックセット操作中の同時実行レベルの動作を選択できるパラメータです。なお、選択できるオプションは、厳密な障害体制とソフト障害体制の 2 つがあります。

うん。デフォルトで設定されている厳密な障害体制は、失敗したアカウントの数が障害体制で設定した数プラス 1 の値を超えないように、同時実行レベルが動的に下がります。つまり、デプロイに失敗した場合、同時実行レベルである。実際の同時実行数は、デプロイしたデプロイの数に比例して検証します。そのため、デプロイが失敗するたびにデプロイ速度が低下します。一方で、ソフト障害体制は、デプロイに失敗しても同時実行レベルである実際の同時実行数は一定です。これにより、デプロイの失敗数に限らず、スタックセットの操作を同時アカウントの最大数で実行することができます。うん。しかし、実際の障害数を考慮せずに同時にデプロイを実施するため、障害耐性で設定した値よりもデプロイに失敗したスタックインスタンスが多くなる可能性があることにご留意ください。そのため、厳密な障害耐性よりもデプロイ速度を優先したい場合に使用することを推奨いたします。

では、 A を用いて厳密な障害体制について説明いたします。例では、傷害耐性の値、用語、同時アカウントの最大数を 10 とします。その場合、実際の同時実行数は障害耐性の数の 5 に 1 を足した 6 となります。うん。なお、下の図は障害対数の値が同時アカウントの最大数の値に与える影響と、それらがスタックセット操作の実際の同時実効性に与える影響を示しています。

では早速、デプロイが失敗した場合の挙動について見ていきましょう。 2 つ目の図では、スタックセットが 2 つのスタックインスタンスのデプロイに成功し、 1 つのデプロイに失敗しています。その場合、 3 番目の図のように実際の同時実行数は 6 に検証します。その後、さらに 2 つのスタックインスタンスのデプロイに失敗すると、実際の同時実行数は 5 に減少します。なお、失敗したスタックインスタンスの合計は 3 つとなります。続いて、スタックセットが 3 つのデプロイに失敗すると、失敗したスタックインスタンスの合計は 6 つとなります。その場合、失敗したスタックインスタンスの数が障害体制で設定していたと等しくなるため、スタックセットは操作に失敗します。今回の例では、スタックセットは操作を停止するまでに 9 つのスタックインスタンスをデプロイし、 6 つのスタックインスタンスにて失敗しました。

次に、ソフト障害体制について例を用いてご説明いたします。例では、傷害体制の値を後、同時アカウントの最大数を 10 としています。なお、ソフト障害体制では、実際の同時実行数は同時アカウントの最大数と同じ 10 となります。

うん。では、ソフト障害体制にてデプロイが失敗した場合の挙動について見ていきましょう。厳密な障害体制と異なり、ソフト障害。障害耐性の場合は、スタックの操作が失敗しても実際の同時実行数は 10 のまま変わりありません。さらに、スタックインスタンスのデプロイが 2 回失敗しても、実際の同時実行数に変化はございません。うん。加えて、 6 つのスタックの操作に失敗し、障害耐性の値である 5 に達した場合には、スタックセットは操作に失敗します。うん。しかし、同時軸呼吸に残りの操作がある場合には、操作を完了するまでスタックセットはデプロイを続けます。うん。そのため、最終的には障害体制数に達した際に残っていた 7 つのスタックインスタンスにデプロイを実施し、合計 8 つのスタックインスタンスが失敗しております。このように、ソフト障害体制は、障害が発生したスタックインスタンスの数が障害対数の数より多くなる可能性がございます。

次に、スタックセットの更新方法についてご説明します。スタックセットを更新する場合、 4 つの手順にて更新することが可能です。まずはじめに、更新したいスタックセットを選択し、スタックセットの詳細を編集を押下します。そして、現在のテンプレートを使用するか、新しくテンプレートを置き換えるかを選択します。次に、スタックセットの説明やパラメータなどの詳細を変更します。次に、作成時と同様にタグや実行設定を指定することができます。最後にこちらも作成時と同様にデプロイ先やリージョンなどを指定することが可能です。

ここで、サービス Monaged のアクセス許可モデルの場合に設定できるアカウントフィルタータイプについてご説明します。ガルンとフィルタータイプにはなし、共通集合、差分和集合の 4 種類があり、これらを適宜に用いることで、指定された組織単位を使用してデプロイを個々のターゲットアカウントに制限することなどが可能となります。

続いて、 Stack インスタンスのパラメータの上書きについてご説明します。こちらはアカウントおよびリージョン別に Stack インスタンスのパラメータを上書きすることが可能です。 3 つの手順にて Stack インスタンスのパラメータを上書きすることが可能です。はじめに任意のスタックセットを選択し、アクションからスタックセットのパラメータを上書きを押下します。うん。そしてデプロイ先やリージョン、デプロイオプションを指定します。そして、上書きしたいパラメーターを選択し、アクションからスタックセット値の上書きを謳歌して内容を入力することで上書きできます。また、留意点としては、上書きされたパラメータをスタックセットで指定された値に戻すためには、スタックセット値に設定を選択する必要がございます。なお、上書きされたパラメータは設定したアカウントのみ適用されることもご留意ください。

次に、スタックの削除について説明いたします。 Stack の削除は、指定したターゲットアカウントからスタックを削除を実行することが可能です。なおあわせて対象のスタックインスタンスも削除されます。実際に削除する方法としては、はじめに削除するスタックを選択し、アクションのスタックセットからスタックを削除を押下します。次に、デプロイ先やデプロイオプションなどを選択することで削除することができます。最後にスタックセットの削除についてご説明いたします。なお、スタックセットはスタックセット内にスタックインスタンスがない場合にのみ削除することが可能です。はじめに削除するスタックセットを選択し、アクションからスタックセットの削除を押下します。次に削除ボタンを押下することで、スタックセットを削除することができ。きます。以上でこのセミナーを終了します。

最後に、 AWS ブラックベルトオンラインセミナーの説明と内容についての注意を表示します。本セミナーへの感想は XQ Twitter までお寄せください。