Red Hat OpenShift Service on AWS 4

チュートリアル

Red Hat OpenShift Service on AWS のチュートリアル

Last Updated: 2025-12-06

Red Hat OpenShift Service on AWS 4 チュートリアル

Red Hat OpenShift Service on AWS のチュートリアル

Legal Notice

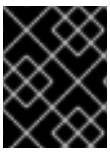
Abstract

初めての Red Hat OpenShift Service on AWS (ROSA) クラスター作成に関するチュートリアルです。

Table of Contents

第1章 チュートリアルの概要

Red Hat エキスパートによるステップバイステップのチュートリアルを使用して、マネージド OpenShift クラスターを最大限に活用してください。



重要

このコンテンツは Red Hat のエキスパートが作成したものですが、サポート対象のすべての設定でまだテストされていません。

第2章 チュートリアル: RED HAT OPENSHIFT SERVICE ON AWS のアクティブ化とアカウントのリンク

このチュートリアルでは、最初のクラスターをデプロイする前に、Red Hat OpenShift Service on AWS をアクティブ化し、AWS アカウントにリンクするプロセスを説明します。

重要

製品のプライベートオファーを受け取っている場合、このチュートリアルを始める前に、プライベートオファーに記載されている手順に従ってください。プライベートオファーは、製品がすでにアクティブ化されている場合 (この場合はアクティブなサブスクリプションを置き換えます)、また初めてアクティブ化する場合に利用できるように設計されています。

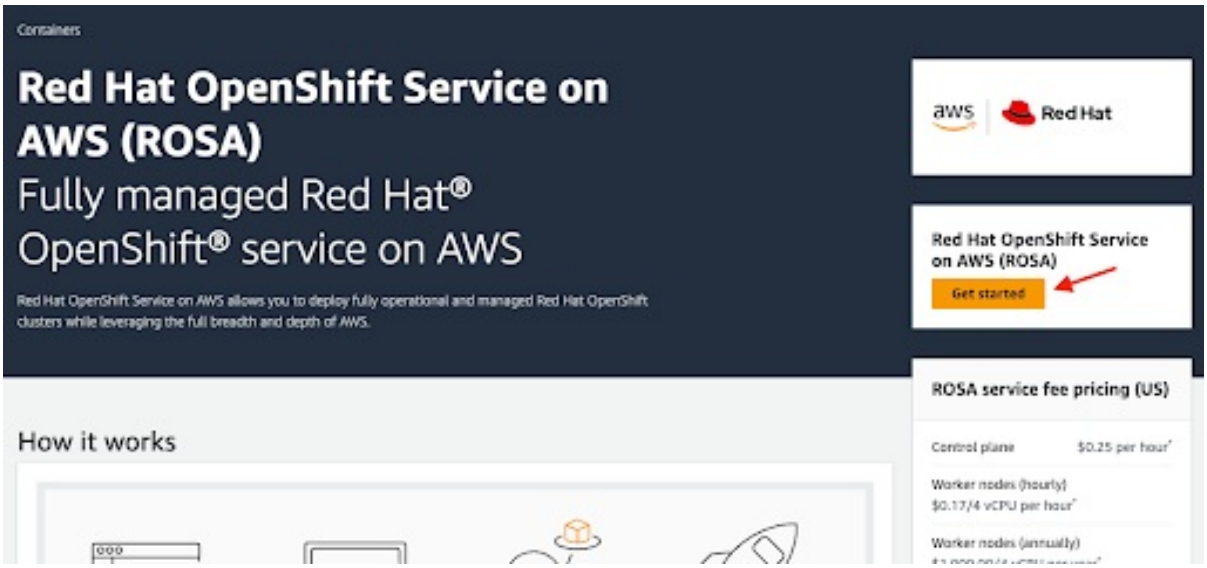
2.1. 前提条件

- Red Hat OpenShift Service on AWS 製品サブスクリプションをアクティブ化する AWS アカウントに関連付ける Red Hat アカウントにログインします。
- サービスの請求に使用する AWS アカウントは、1 つの Red Hat アカウントにのみ関連付けることができます。通常は、AWS 支払者のアカウントが、Red Hat OpenShift Service on AWS へのサブスクライブと、アカウントのリンクおよび請求に使用されます。
- 同じ Red Hat 組織に属するすべてのチームメンバーが、Red Hat OpenShift Service on AWS クラスターの作成中に、サービスの請求用にリンクされた AWS アカウントを使用できます。

2.2. サブスクリプションの有効化と AWS アカウントのセットアップ

1. AWS コンソールページ で **Get started** ボタンをクリックして、Red Hat OpenShift Service on AWS 製品をアクティブ化します。

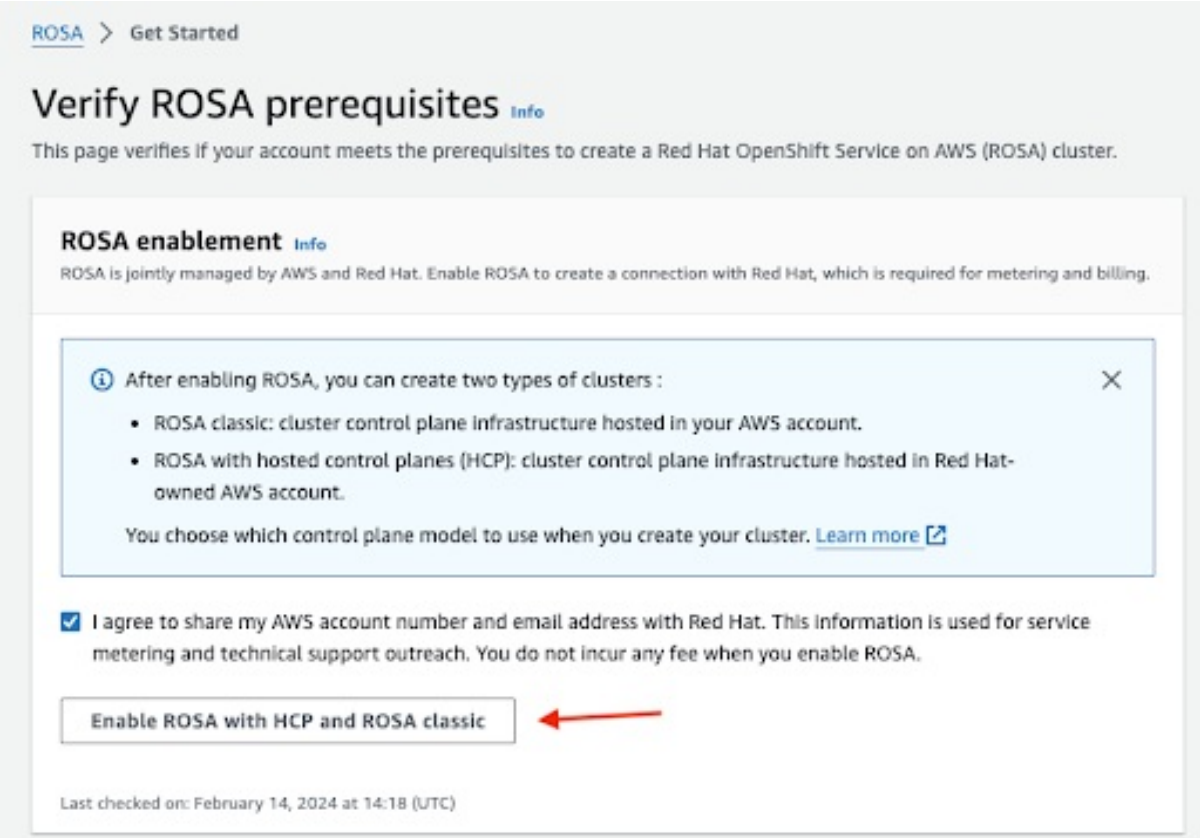
   図2.1 使用してみる

   以前に Red Hat OpenShift Service on AWS をアクティブ化したが、プロセスを完了していない場合は、ボタンをクリックして、次の手順で説明するようにアカウントのリンクを完了できます。

2. 連絡先情報を Red Hat と共有することを確認し、サービスを有効にします。

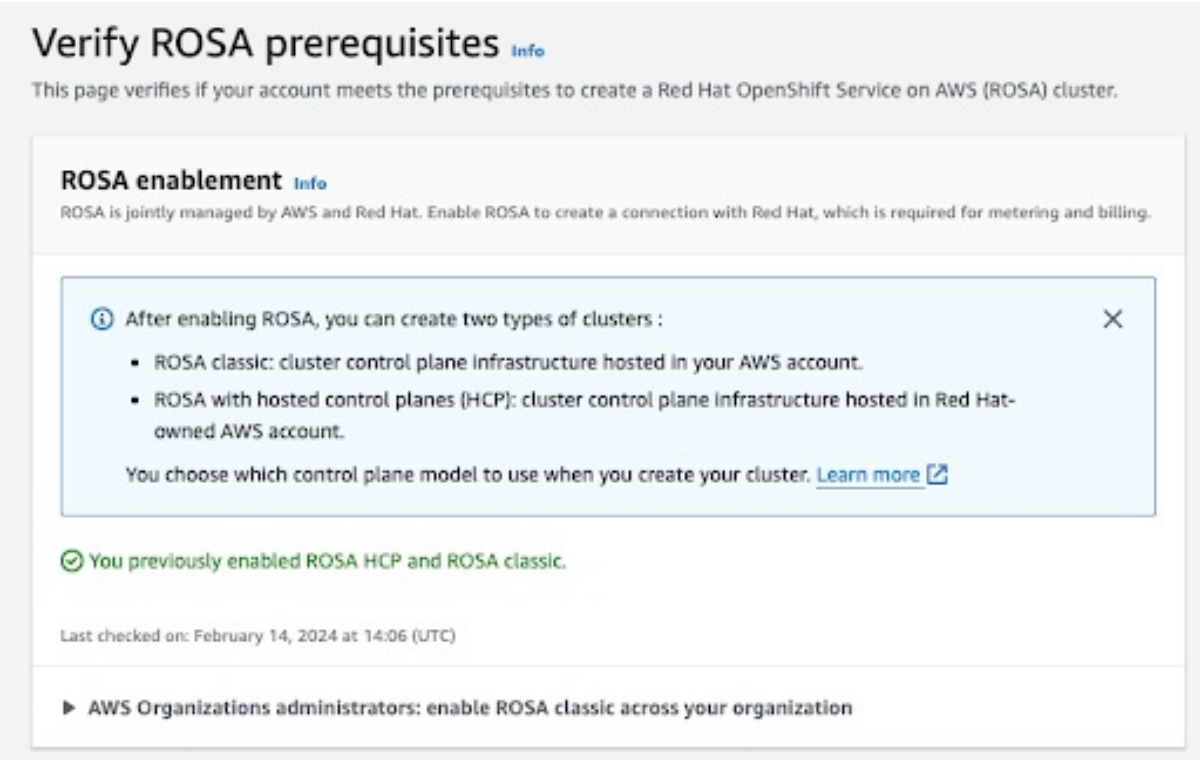
図2.2 Red Hat OpenShift Service on AWS の有効化



- このステップでサービスを有効にしても料金は発生しません。課金と計測が連携されるのは、最初のクラスターをデプロイした後のみです。これには数分かかる場合があります。

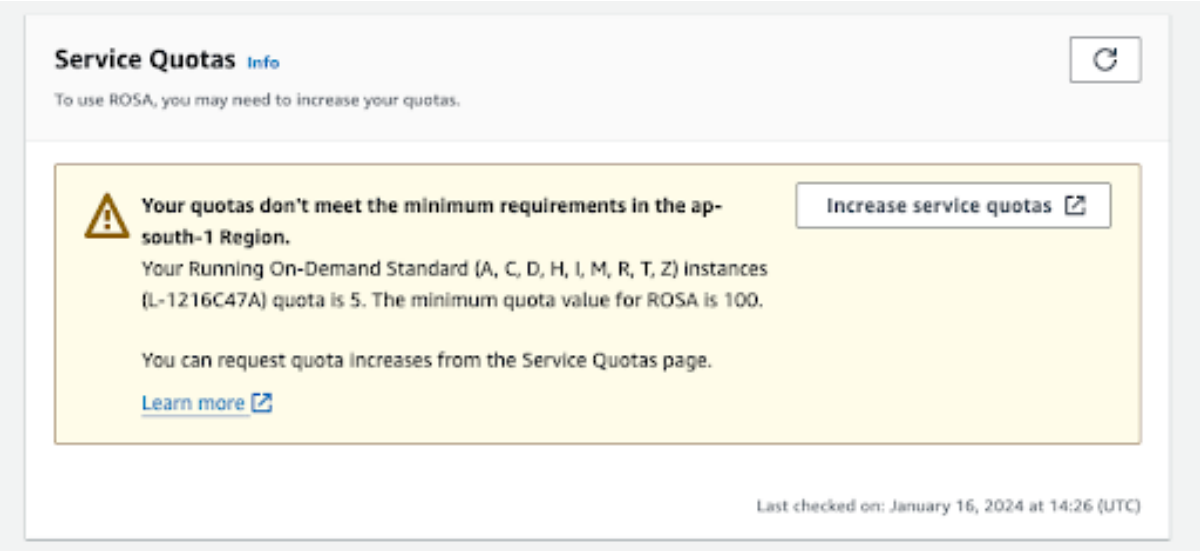
3. プロセスが完了すると、確認メッセージが表示されます。

図2.3 Red Hat OpenShift Service on AWS の有効化の確認



4. この検証ページの他のセクションには、追加の前提条件 (が満たされているかどうか) のステータスが表示されます。これらの前提条件のいずれかが満たされていない場合は、対応するメッセージが表示されます。選択したリージョンで割り当てが不十分な例を次に示します。

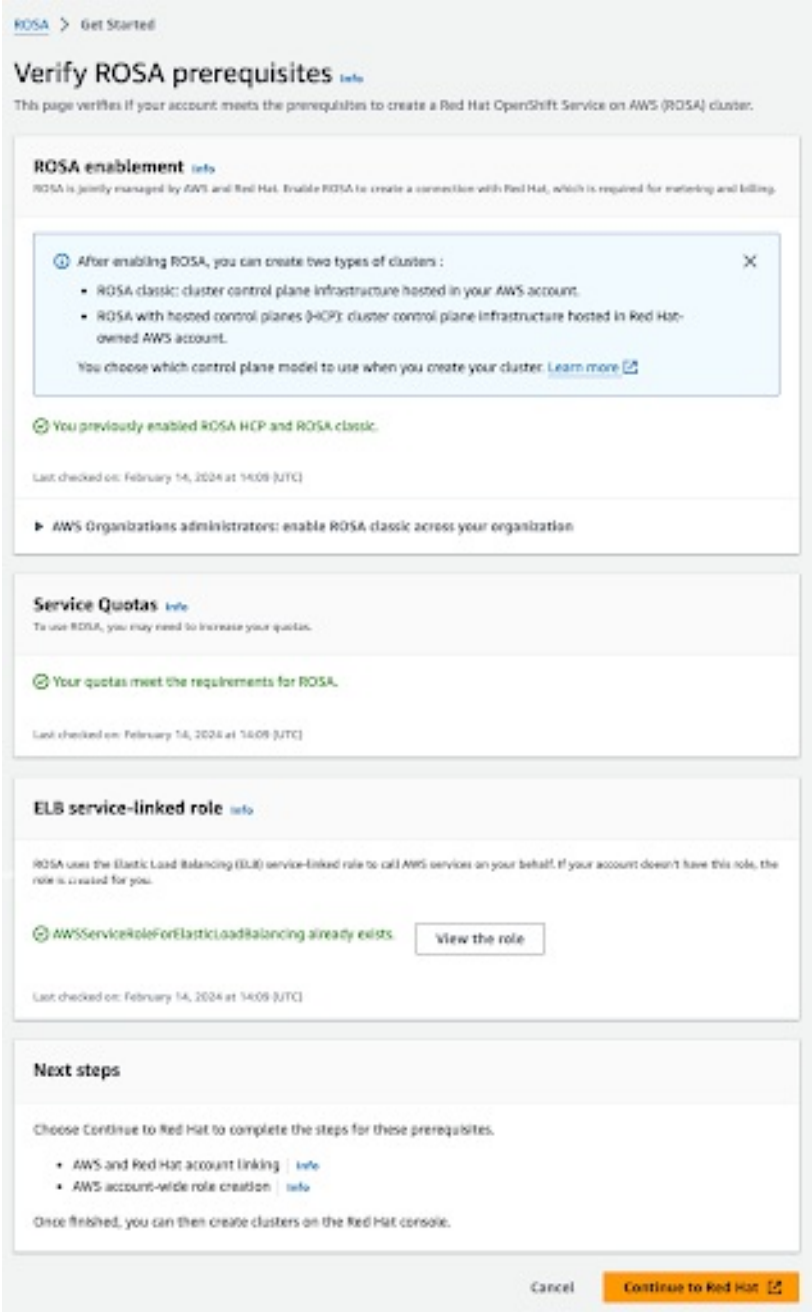
図2.4 サービスクォータ



   a. **Increase service quotas** ボタンをクリックするか、**Learn more** リンクを使用して、サービスクォータの管理方法に関する詳細情報を取得します。クォータが不十分な場合、クォータはリージョン固有であることに注意してください。Web コンソールの右上隅にあるリージョンスイッチャーを使用して、関心のあるリージョンのクォータチェックを再実行し、必要に応じてサービスクォータの増加リクエストを送信できます。

5. すべての前提条件が満たされている場合、ページは次のようになります。

図2.5 Red Hat OpenShift Service on AWS の前提条件の確認

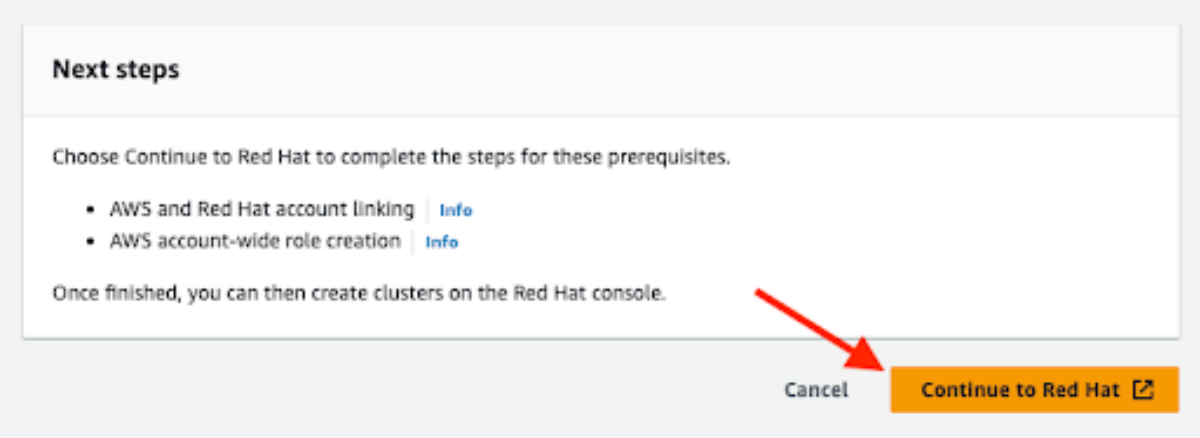


ELB サービスにリンクされたロールは自動的に作成されます。青色の小さな **Info** リンクをクリックすると、状況に応じたヘルプやリソースが表示されます。

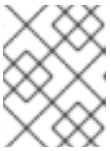
2.3. AWS と RED HAT のアカウントとサブスクリプションのリンク

1. オレンジ色の **Continue to Red Hat**ボタンをクリックして、アカウントのリンクを続行します。

図2.6 Red Hat に進む



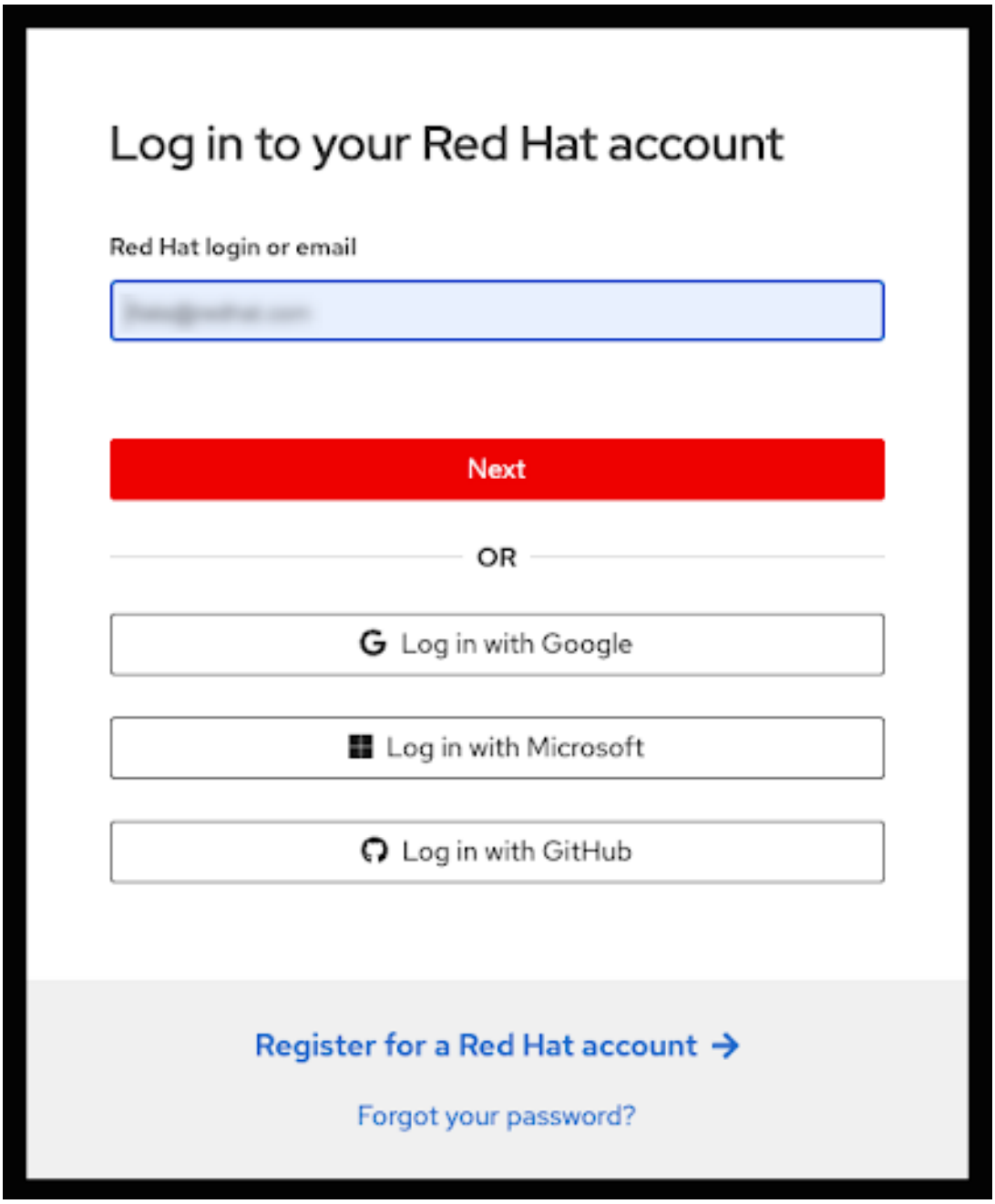
2. 現在のブラウザーのセッションで Red Hat アカウントにまだログインしていない場合は、アカウントにログインするように求められます。



注記

お使いの AWS アカウントが 1 つの Red Hat 組織にリンクされている必要があります。

図2.7 Red Hat アカウントへのログイン



- このページでは、新しい Red Hat アカウントに登録したり、パスワードをリセットしたりすることもできます。
- Red Hat OpenShift Service on AWS 製品サブスクリプションをアクティブ化した AWS アカウントに関連付ける Red Hat アカウントにログインします。
- サービスの請求に使用する AWS アカウントは、1 つの Red Hat アカウントにのみ関連付けることができます。通常は、AWS 支払者のアカウントが、Red Hat OpenShift Service on AWS へのサブスクライブと、アカウントのリンクおよび請求に使用されます。
- 同じ Red Hat 組織に属するすべてのチームメンバーが、Red Hat OpenShift Service on AWS クラスターの作成中に、サービスの請求用にリンクされた AWS アカウントを使用できます。

3. 利用規約を確認した後、Red Hat アカウントの連携を完了させます。

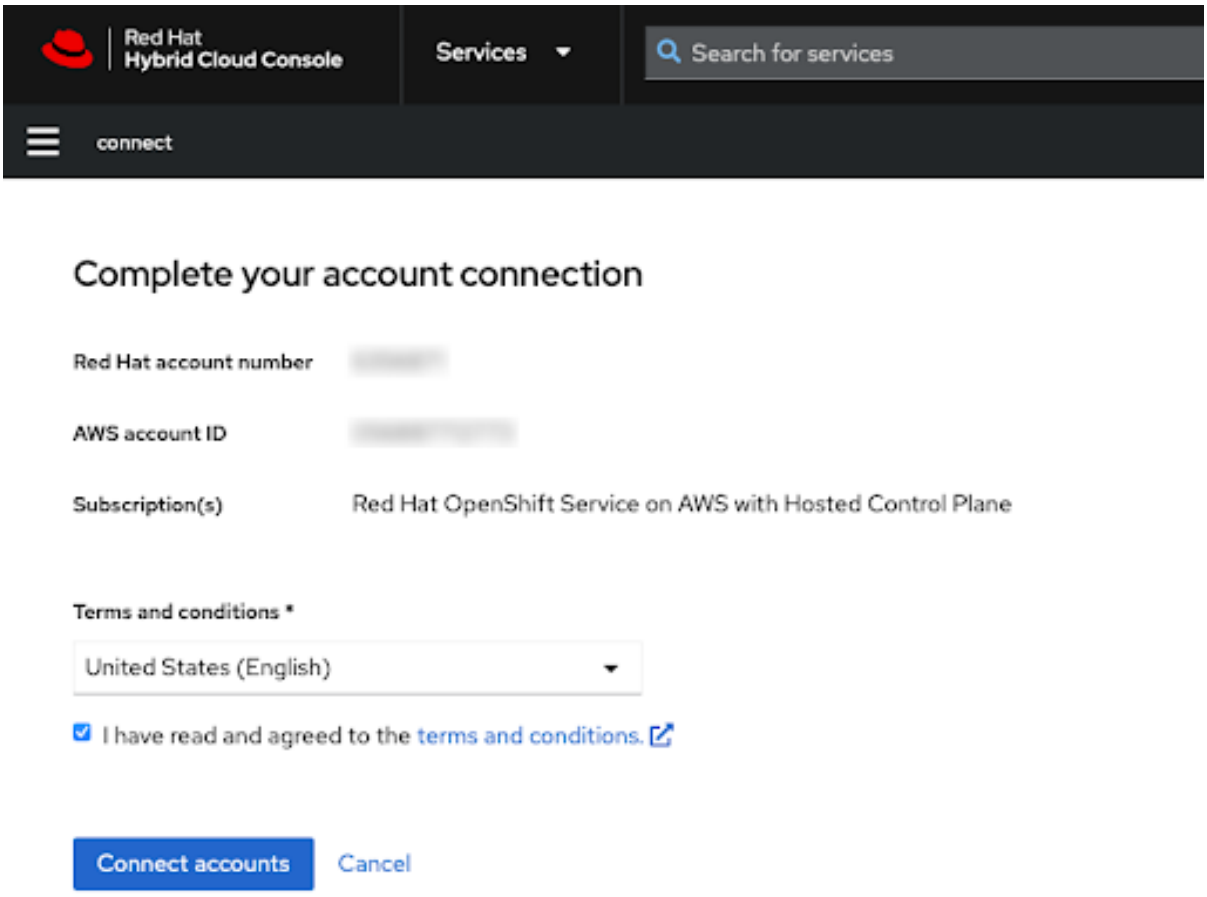
注記

この手順は、AWS アカウントが以前にどの Red Hat アカウントにもリンクされていない場合にのみ使用できます。

ユーザーがログイン中の Red Hat アカウントに AWS アカウントがすでにリンクされている場合、この手順はスキップされます。

AWS アカウントが別の Red Hat アカウントにリンクされている場合は、エラーが表示されます。トラブルシューティングは、Correcting Billing Account Information for HCP clusters を参照してください。

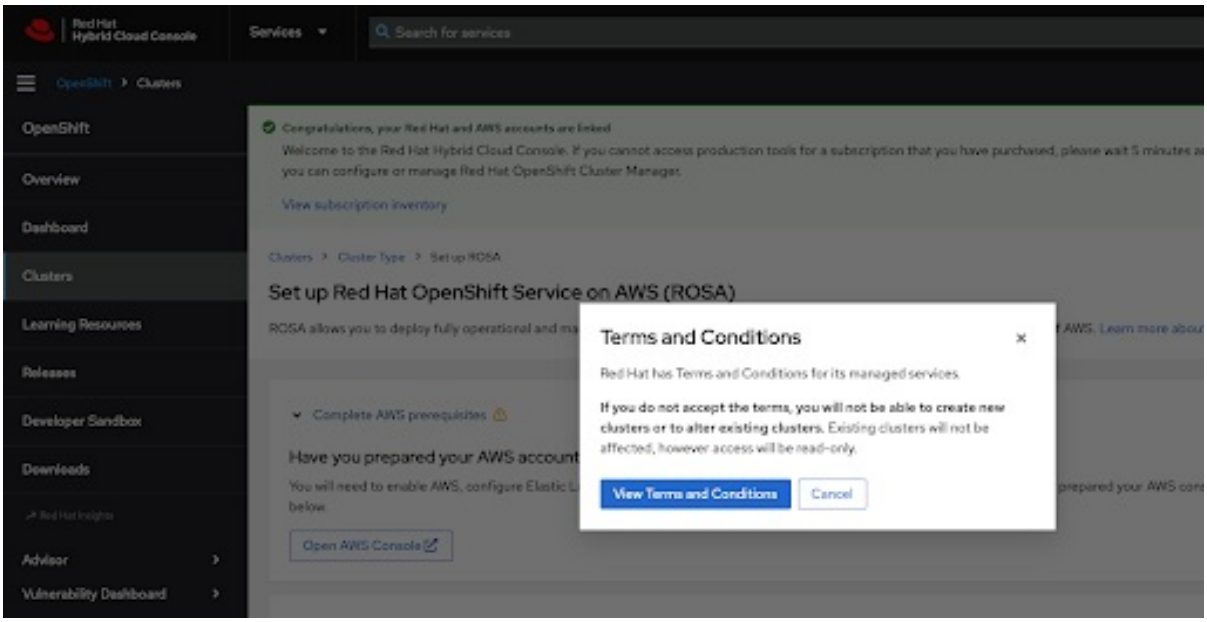
図2.8 アカウント連携の完了



Red Hat と AWS の両方のアカウント番号がこの画面に表示されます。

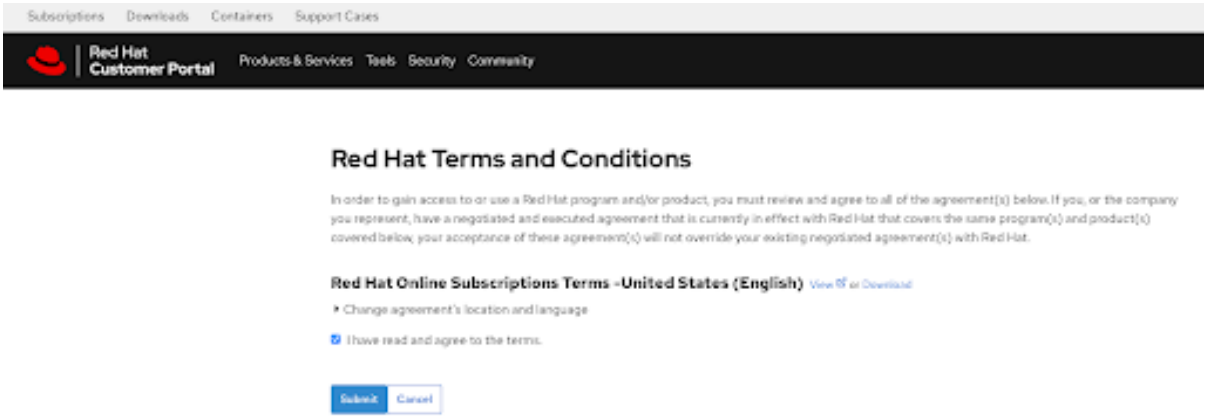
4. サービス規約に同意する場合は、**Connect accounts** ボタンをクリックします。
Red Hat Hybrid Cloud Console を初めて使用する場合は、最初のクラスターを作成する前に、マネージドサービスの一般利用規約に同意するよう求められます。

図2.9 利用規約



View Terms and Conditions ボタンをクリックすると、確認して同意する必要がある追加の利用規約が表示されます。

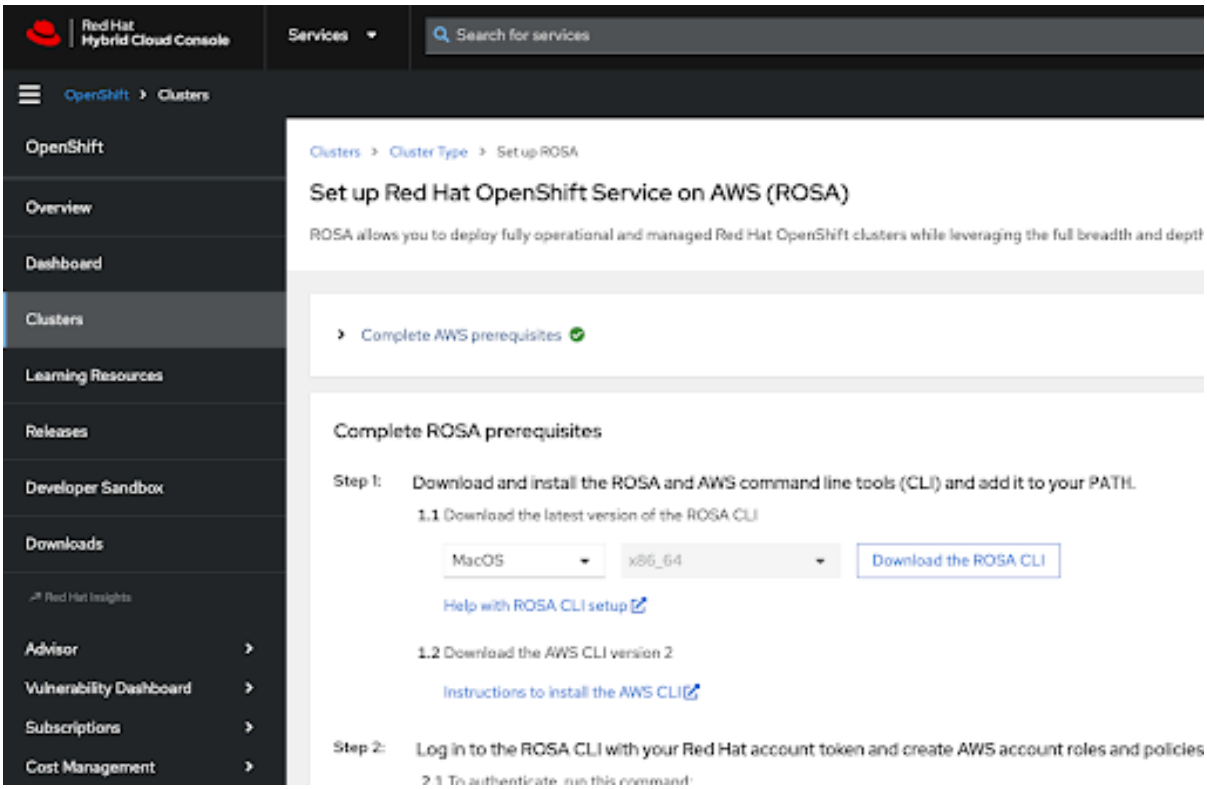
図2.10 Red Hat の利用規約



この時点で追加の条件を確認し、プロンプトが表示されたら同意を確定します。

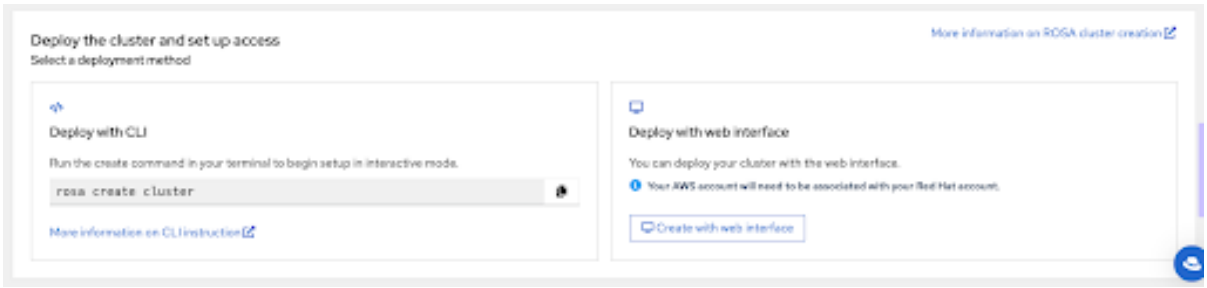
5. Hybrid Cloud Console に、AWS アカウントのセットアップ完了に関する確認メッセージと、クラスターのデプロイの前提条件が表示されます。

図2.11 Red Hat OpenShift Service on AWS の前提条件の完了



このページの最後のセクションでは、**rosa** CLI または Web コンソールを使用したクラスターデプロイメントオプションを示します。

図2.12 クラスターのデプロイとアクセスの設定



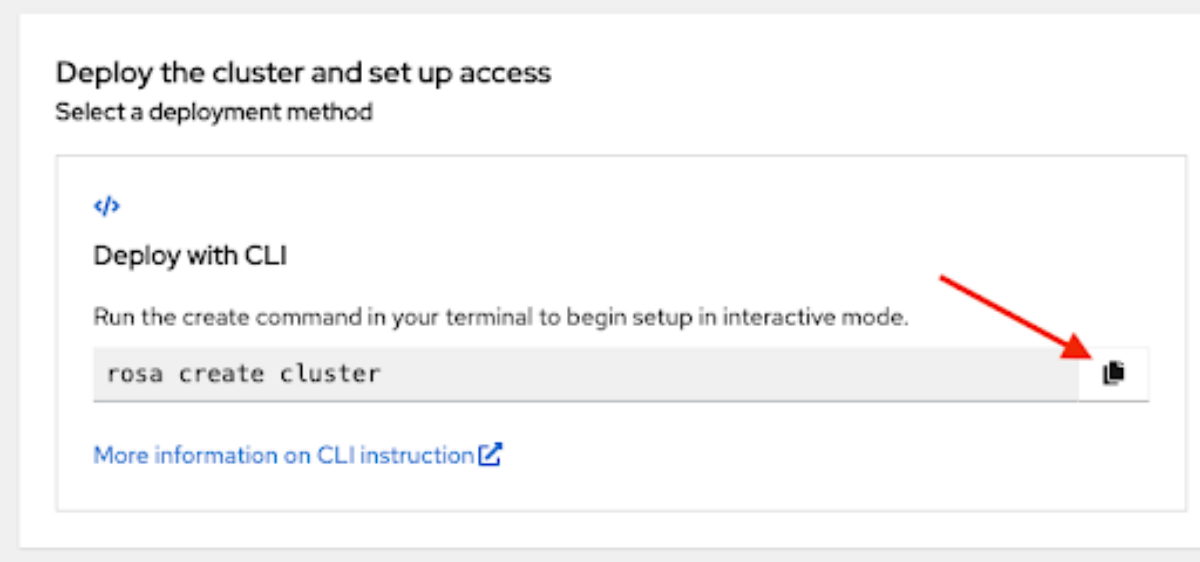
2.4. CLI を使用したクラスターデプロイ時に RED HAT OPENSHIFT SERVICE ON AWS の AWS 請求先アカウントを選択する

重要

最新の ROSA コマンドラインインターフェイス (CLI) と AWS CLI がインストールされており、前のセクションで説明した Red Hat OpenShift Service on AWS の前提条件が満たされていることを確認してください。詳細は、ROSA CLI セットアップのヘルプ および AWS CLI のインストール手順 を参照してください。

1. **rosa create cluster** コマンドを使用してクラスターのデプロイを開始します。コンソールの Set up Red Hat OpenShift Service on AWS (ROSA) ページ で **コピー** ボタンをクリックし、コマンドをターミナルに貼り付けることができます。これにより、クラスター作成プロセスが対話モードで起動します。

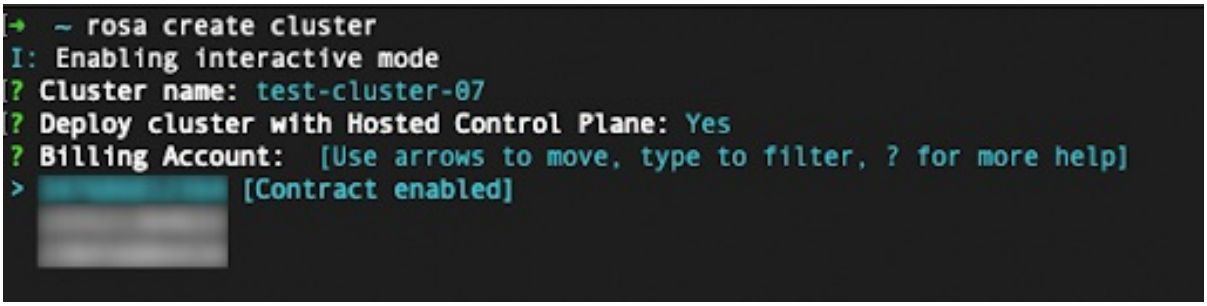
図2.13 クラスターのデプロイとアクセスの設定



2. カスタムの AWS プロファイルを使用するには、**~/.aws/credentials** で指定したデフォルト以外のプロファイルのいずれかを使用します。rosa create cluster コマンドに **–profile <profile_name>** のセレクターを追加すると、コマンドは rosa create cluster **–profile stage** のようになります。このオプションを使用して、AWS CLI プロファイルが指定されていない場合は、デフォルトの AWS CLI プロファイルにより、AWS インフラストラクチャーのプロファイルがどのクラスターにデプロイされるかが決定されます。請求用 AWS プロファイルは、次のいずれかの手順で選択します。

3. Red Hat OpenShift Service on AWS クラスターをデプロイする際に、請求先の AWS アカウントを指定する必要があります。

図2.14 請求先アカウントの指定



- ユーザーがログイン中の Red Hat アカウントにリンクされている AWS アカウントのみが表示されます。
- 指定した AWS アカウントに、Red Hat OpenShift Service on AWS サービスの使用料が課金されます。
- 特定の AWS 請求先アカウントに対して Red Hat OpenShift Service on AWS の契約が有効になっているかどうかが表示されます。
    - **Contract enabled** というラベルが表示されている AWS 請求先アカウントを選択した場合、前払い済みの契約の容量が消費されるまで、オンデマンドの消費料金は課金されません。
    - **Contract enabled** というラベルのない AWS アカウントには、該当するオンデマンド消費料金が課金されます。

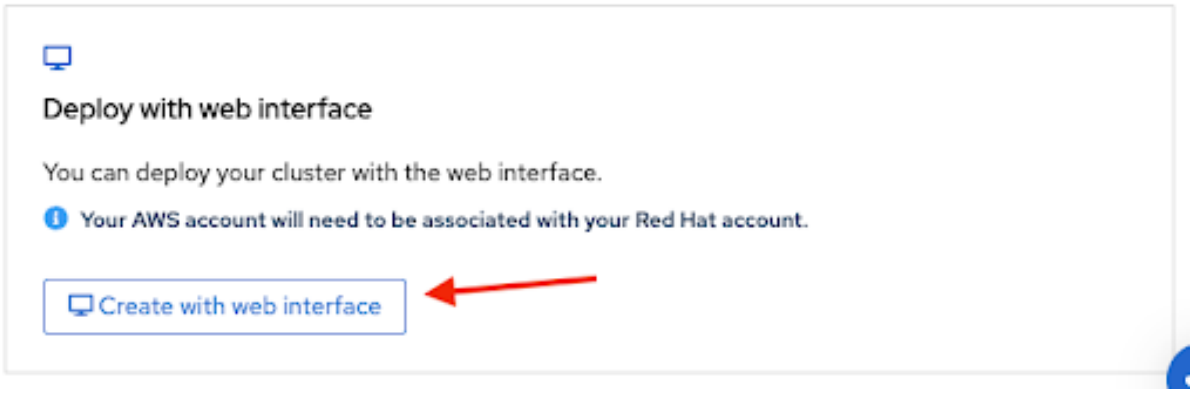
関連情報

- 詳細なクラスターデプロイ手順は、このチュートリアルの範囲外です。CLI を使用して Red Hat OpenShift Service on AWS クラスターのデプロイを完了する方法の詳細は、デフォルトのオプションを使用した Red Hat OpenShift Service on AWS クラスターの作成 を参照してください。

2.5. WEB コンソールを使用したクラスターデプロイ時に RED HAT OPENSHIFT SERVICE ON AWS の AWS 請求先アカウントを選択する

1. Web コンソールを使用してクラスターを作成するには、導入ページの **Set up Red Hat OpenShift Service on AWS** の下部にある 2 番目のオプションを選択します。

図2.15 Web インターフェイスでのデプロイ



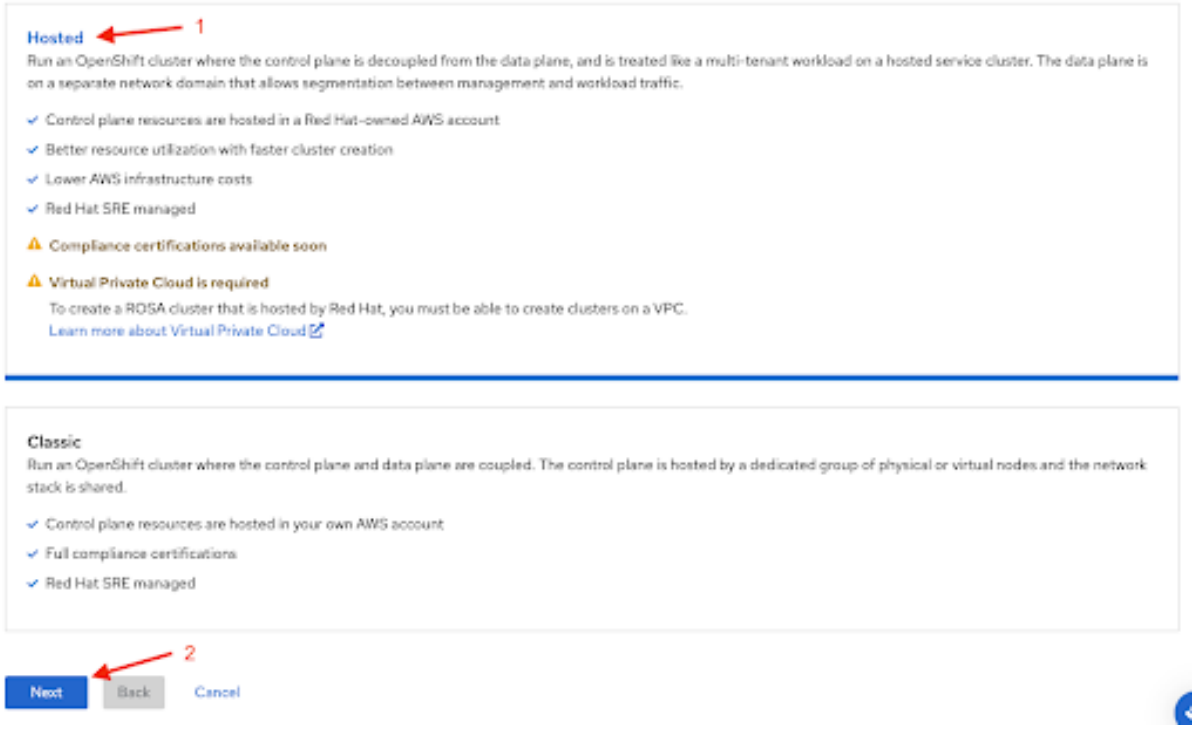
注記

Web コンソールでのデプロイプロセスを開始する前に、前提条件を完了してください。

アカウントロールの作成など、一部のタスクには **rosa** CLI が必要です。Red Hat OpenShift Service on AWS を初めてデプロイする場合は、Web コンソールのデプロイ手順を開始する前に、CLI の手順に従って **rosa whoami** コマンドまで実行してください。

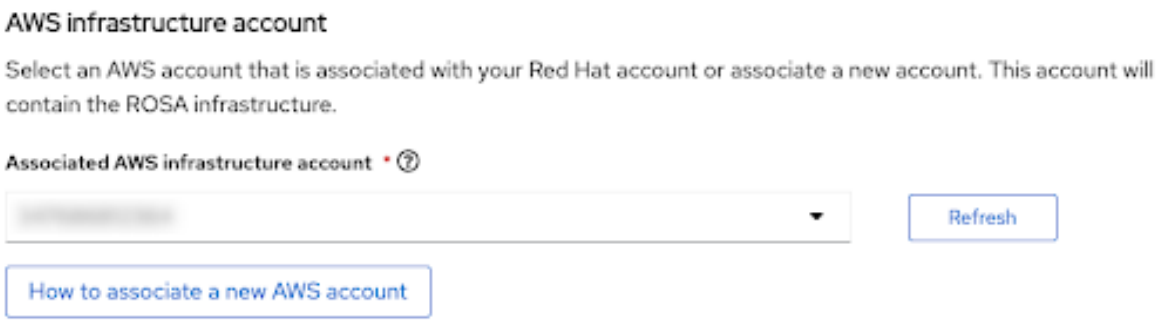
2. Web コンソールを使用して Red Hat OpenShift Service on AWS クラスターを作成する場合の最初のステップは、コントロールプレーンの選択です。**Next** ボタンをクリックする前に、**Hosted** オプションが選択されていることを確認してください。

図2.16 Hosted オプションの選択



3. 次のステップ **Accounts and roles**では、インフラストラクチャー AWS アカウントを指定できます。このアカウントが Red Hat OpenShift Service on AWS クラスターのデプロイ先となり、そこでリソースが消費および管理されます。

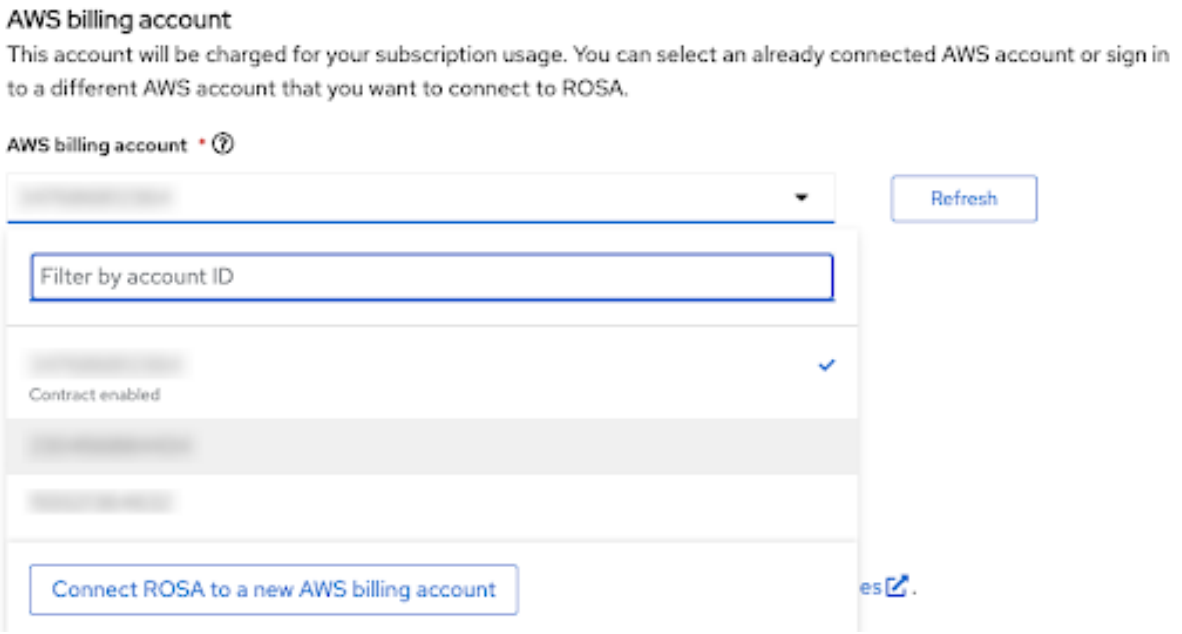
図2.17 AWS インフラストラクチャーアカウント



- Red Hat OpenShift Service on AWS クラスターのデプロイ先のアカウントが表示されない場合は、**How to associate a new AWS account**をクリックして、この関連付けに関して、アカウントロールの作成またはリンクの情報を参照してください。
- これには **rosa** CLI を使用します。
- 複数の AWS アカウントを使用しており、それらのプロファイルが AWS CLI 用に設定されている場合は、**rosa** CLI コマンドを使用するときに **--profile** セレクターを使用して AWS プロファイルを指定できます。

4. 請求先の AWS アカウントは、すぐ次のセクションで選択されます。

図2.18 AWS 請求先アカウント



- ユーザーがログイン中の Red Hat アカウントにリンクされている AWS アカウントのみが表示されます。
- 指定した AWS アカウントに、Red Hat OpenShift Service on AWS サービスの使用料が課金されます。
- 特定の AWS 請求先アカウントに対して Red Hat OpenShift Service on AWS の契約が有効になっているかどうかが表示されます。
    - **Contract enabled** というラベルが表示されている AWS 請求先アカウントを選択した場合、前払い済みの契約の容量が消費されるまで、オンデマンドの消費料金は課金されません。
    - **Contract enabled** というラベルのない AWS アカウントには、該当するオンデマンド消費料金が課金されます。

請求先の AWS アカウントの選択以降の手順は、このチュートリアルの範囲外です。

関連情報

- CLI を使用してクラスターを作成する方法は、CLI を使用した Red Hat OpenShift Service on AWS クラスターの作成 を参照してください。
- Web コンソールを使用してクラスターのデプロイを完了する方法の詳細は、こちらのラーニングパス を参照してください。

第3章 チュートリアル: RED HAT OPENSHIFT SERVICE ON AWS のプライベートオファーの承認と共有

このガイドでは、Red Hat OpenShift Service on AWS のプライベートオファーを承認する方法と、すべてのチームメンバーがプロビジョニングするクラスターでプライベートオファーを使用できるようにする方法を説明します。

Red Hat OpenShift Service on AWS のコストは、AWS インフラストラクチャーのコストと Red Hat OpenShift Service on AWS サービスのコストで構成されます。必要なワークロードを実行している EC2 インスタンスなどの AWS インフラストラクチャーのコストは、インフラストラクチャーがデプロイされている AWS アカウントに課金されます。Red Hat OpenShift Service on AWS サービスのコストは、クラスターをデプロイするときに "AWS 請求先アカウント" として指定した AWS アカウントに課金されます。

各コストの請求先を、別々の AWS アカウントにすることができます。Red Hat OpenShift Service on AWS サービスのコストと AWS インフラストラクチャーのコストの計算方法の詳細は、Red Hat OpenShift Service on AWS の料金ページ を参照してください。

3.1. プライベートオファーの承認

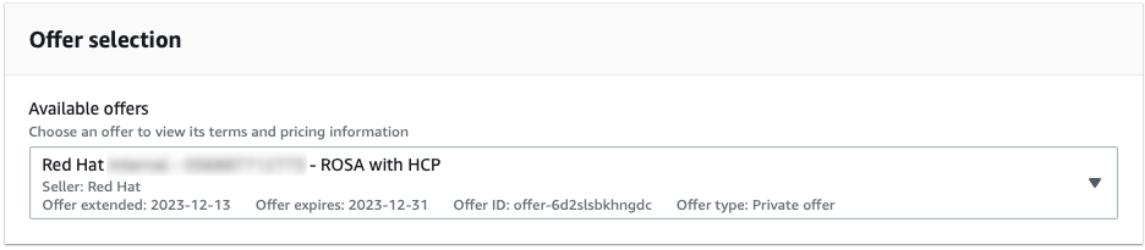
1. Red Hat OpenShift Service on AWS のプライベートオファーを承認すると、販売者が指定した特定の AWS アカウント ID でのみアクセスできる一意の URL が提供されます。

注記

購入者として指定された AWS アカウントを使用してログインしていることを確認してください。別の AWS アカウントを使用してオファーにアクセスしようとすると、以下のトラブルシューティングセクションの図 11 に示すように、"page not found" というエラーメッセージが表示されます。

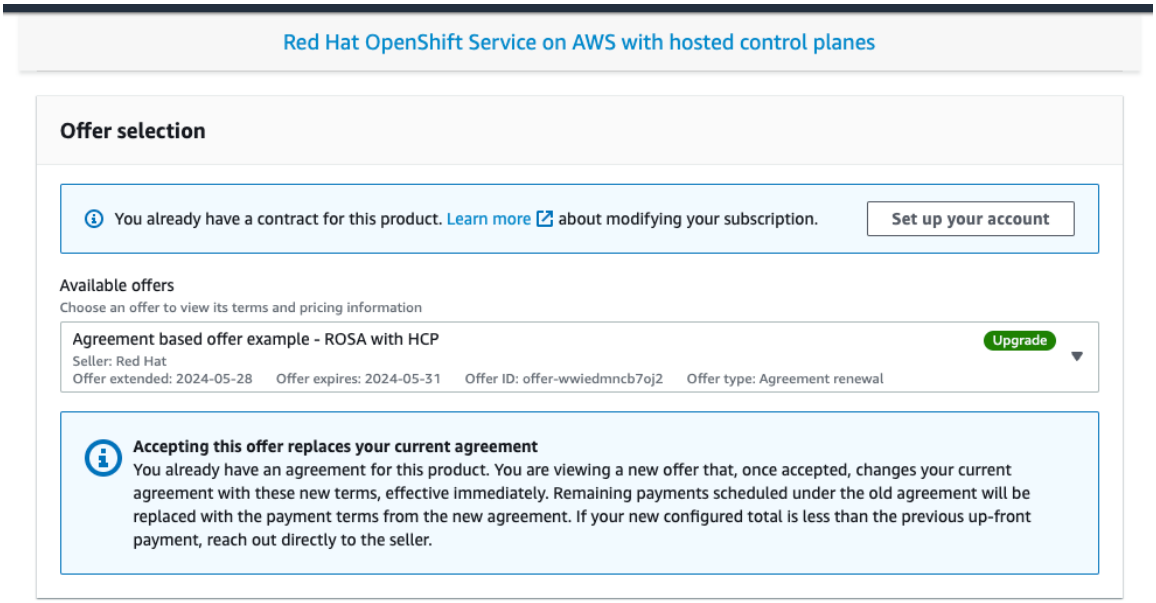
   a. 図 1 はオファー選択ドロップダウンメニューを示しています。通常のプライベートオファーが事前に選択されています。このタイプのオファーは、パブリックオファーまたは別のプライベートオファーを使用する前に Red Hat OpenShift Service on AWS がアクティブ化されていない場合にのみ承認することができます。

図3.1 通常のプライベートオファー



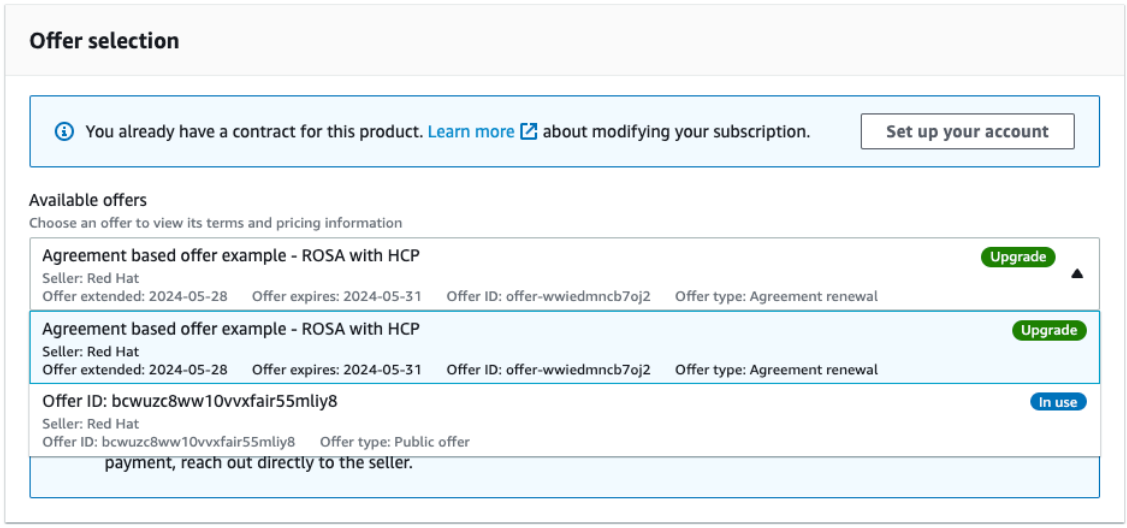
   b. 図 2 は、以前にパブリックオファーを使用して Red Hat OpenShift Service on AWS をアクティブ化した AWS アカウント用に作成されたプライベートオファーを示しています。製品名が表示され、"Upgrade" というラベルの付いたプライベートオファーが選択されています。このオファーを承認すると、現在有効な Red Hat OpenShift Service on AWS の契約が置き換えられます。

図3.2 プライベートオファー選択画面



c. ドロップダウンメニューを使用して、複数のオファーから選択できます (利用可能な場合)。以前にアクティブ化されたパブリックオファーは、図 3 のように、"Upgrade" というラベルが付いた、新しく提供された同意ベースのオファーと一緒に表示されます。

図3.3 プライベートオファー選択ドロップダウン

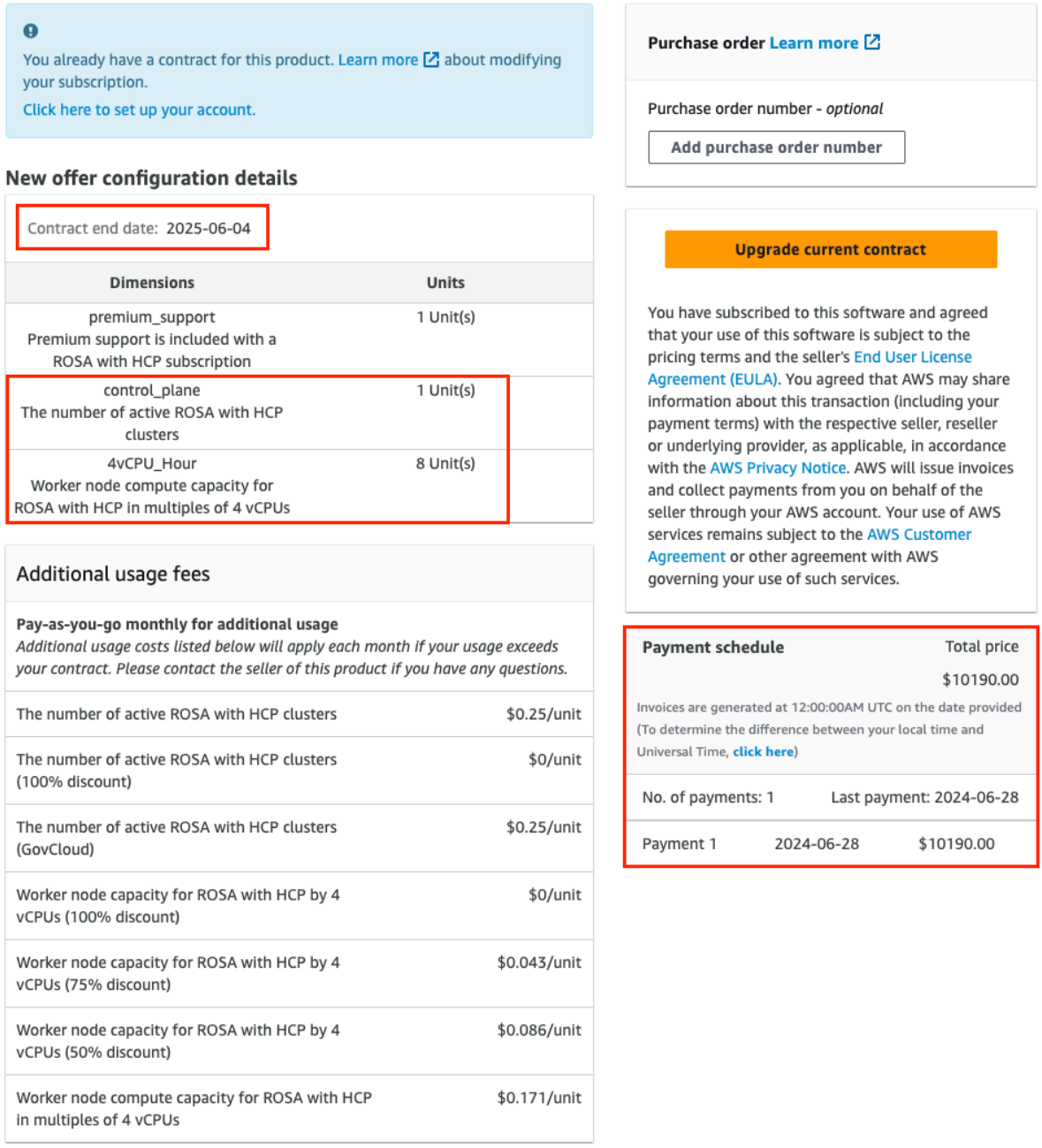


2. オファー設定が選択されていることを確認します。図 4 は、オファーの詳細が記載されたオファーページの下部を示しています。

注記

契約終了日、オファーに含まれるユニット数、および支払いスケジュールが表示されます。この例には、1 つのクラスターと、4 つの仮想 CPU を使用する最大 3 つのノードが含まれています。

図3.4 プライベートオファーの詳細



3. オプション: 購入するサブスクリプションに 独自の注文書 (PO) 番号を追加 して、後の AWS 請求書にその番号を含めることができます。また、"New offer configuration details" の範囲を超える使用量に対して課金される "Additional usage fees" を確認します。

注記

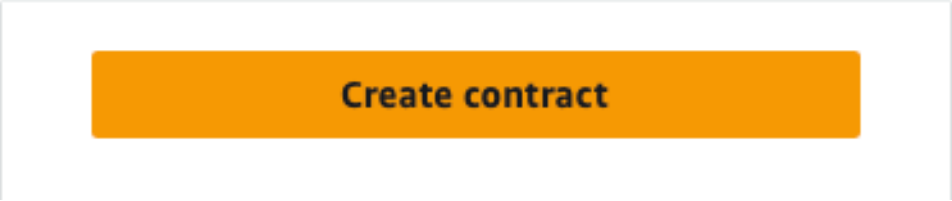
プライベートオファーには、利用可能な設定がいくつかあります。

- 承認するプライベートオファーに、固定された将来の開始日が設定されている可能性があります。
- プライベートオファー、パブリックオファー、または古いプライベートオファーのエンタイトルメントを承認する時点で、別のアクティブな Red Hat OpenShift Service on AWS サブスクリプションがない場合は、プライベートオファー自体を承認し、指定されたサービス開始日以降にアカウントのリンク手順とクラスターのデプロイ手順を続行します。

これらの手順を完了するには、アクティブな Red Hat OpenShift Service on AWS エンタイトルメントが必要です。サービス開始日は、常に UTC タイムゾーンで表示されます。

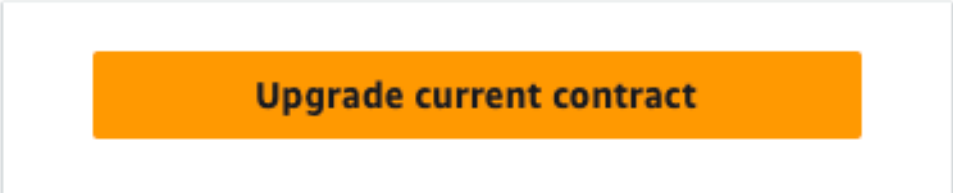
4. 契約を作成またはアップグレードします。
   a. Red Hat OpenShift Service on AWS をまだアクティブ化しておらず、このサービス用の契約を初めて作成する AWS アカウントによってプライベートオファーを承認する場合は、**Create contract** ボタンをクリックします。

      図3.5 Create contract ボタン

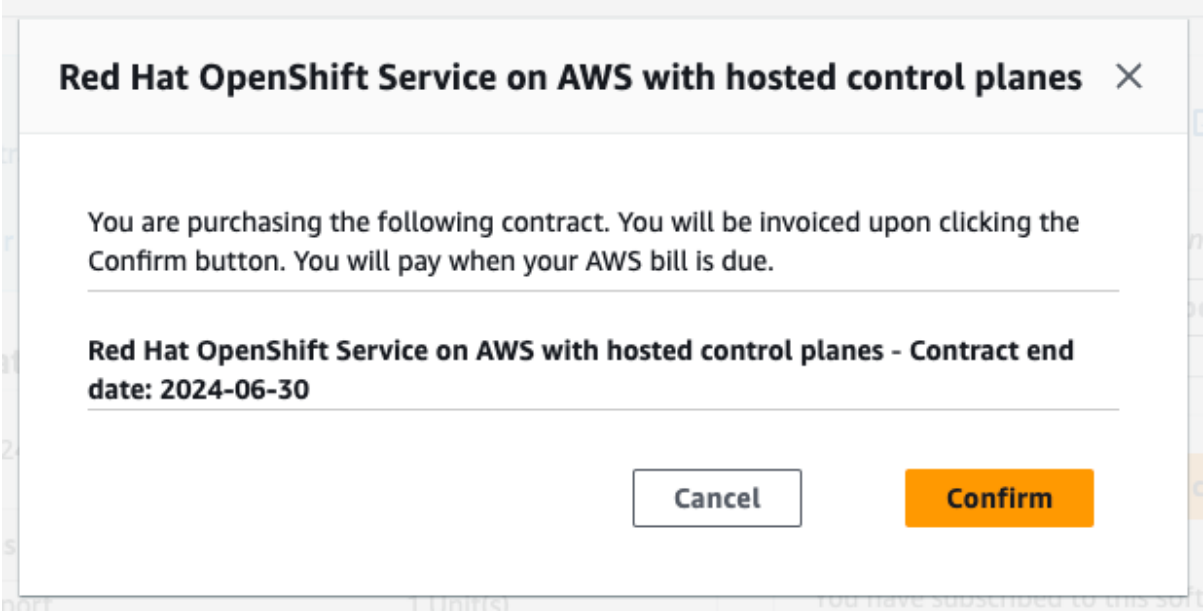
   b. 同意ベースのオファーの場合は、**Upgrade current contract** ボタンをクリックします (図 4 および 6 を参照)。

      図3.6 Upgrade contract ボタン

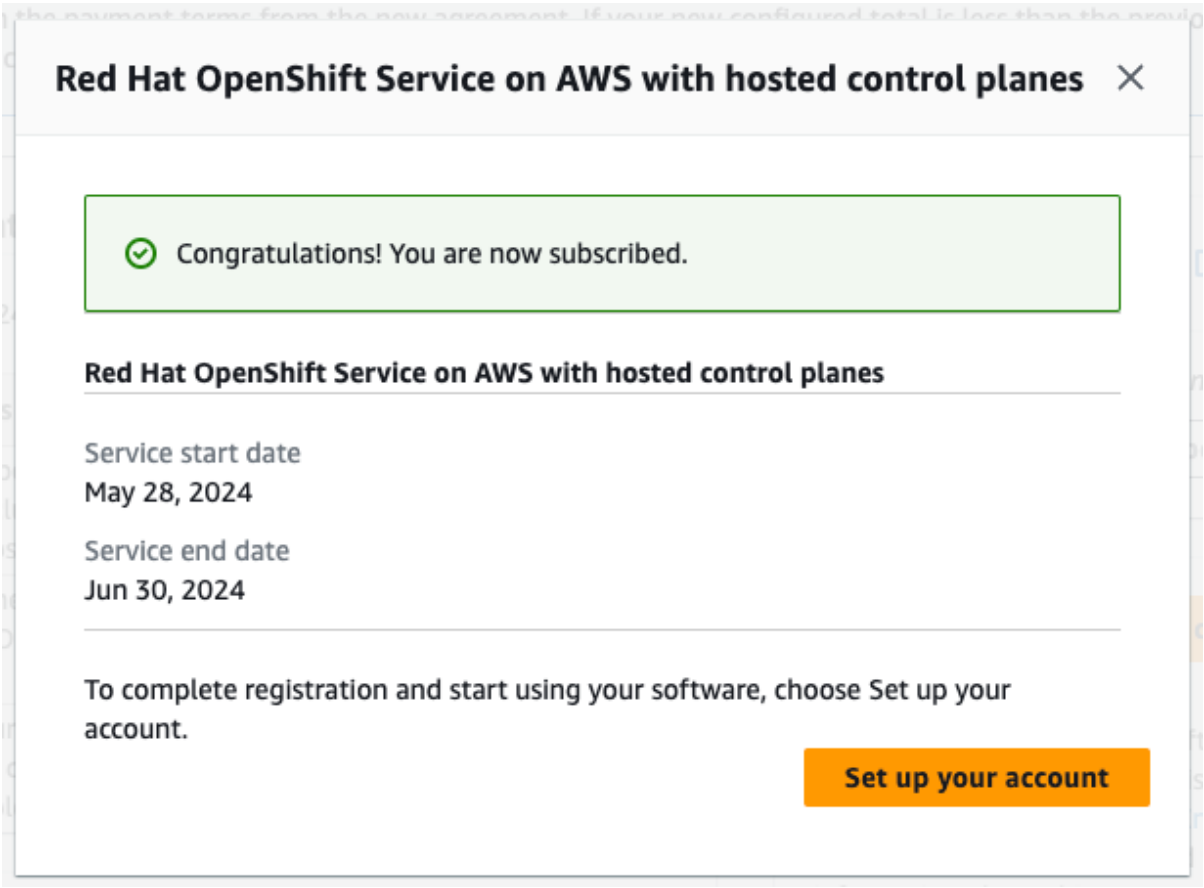
5. **Confirm** をクリックします。

図3.7 プライベートオファー承認の確認ウィンドウ



6. 承認するプライベートオファーのサービス開始日がオファー承認の直後に設定されている場合、確認モーダルウィンドウで **Set up your account**ボタンをクリックします。

図3.8 サブスクリプションの確認



7. 承認するプライベートオファーに将来の開始日が指定されている場合は、サービス開始日後にプライベートオファーページに戻り、**Setup your account**ボタンをクリックして、Red Hat と AWS アカウントのリンクを続行します。

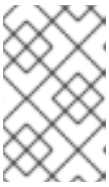
注記

同意が有効でない場合、以下に説明するアカウントリンクがトリガーされず、"アカウントのセットアップ" プロセスを "Service start date" 以降まで実行できません。

これらの日付は常に UTC タイムゾーンの日付です。

3.2. プライベートオファーの共有

1. 前の手順で **Set up your account** ボタンをクリックすると、AWS と Red Hat のアカウントをリンクする手順に進みます。この時点で、オファーを承認した AWS アカウントですでにログインしています。Red Hat アカウントでログインしていない場合は、ログインするように求められます。
   Red Hat OpenShift Service on AWS のエンタイトルメントは、Red Hat 組織アカウントを通じて他のチームメンバーと共有されます。同じ Red Hat 組織内の既存の全ユーザーは、上記の手順に従って、プライベートオファーを承認した請求 AWS アカウントを選択できます。Red Hat 組織管理者としてログインしている場合は、Red Hat 組織内のユーザーを管理 し、新しいユーザーを招待したり作成したりできます。



注記

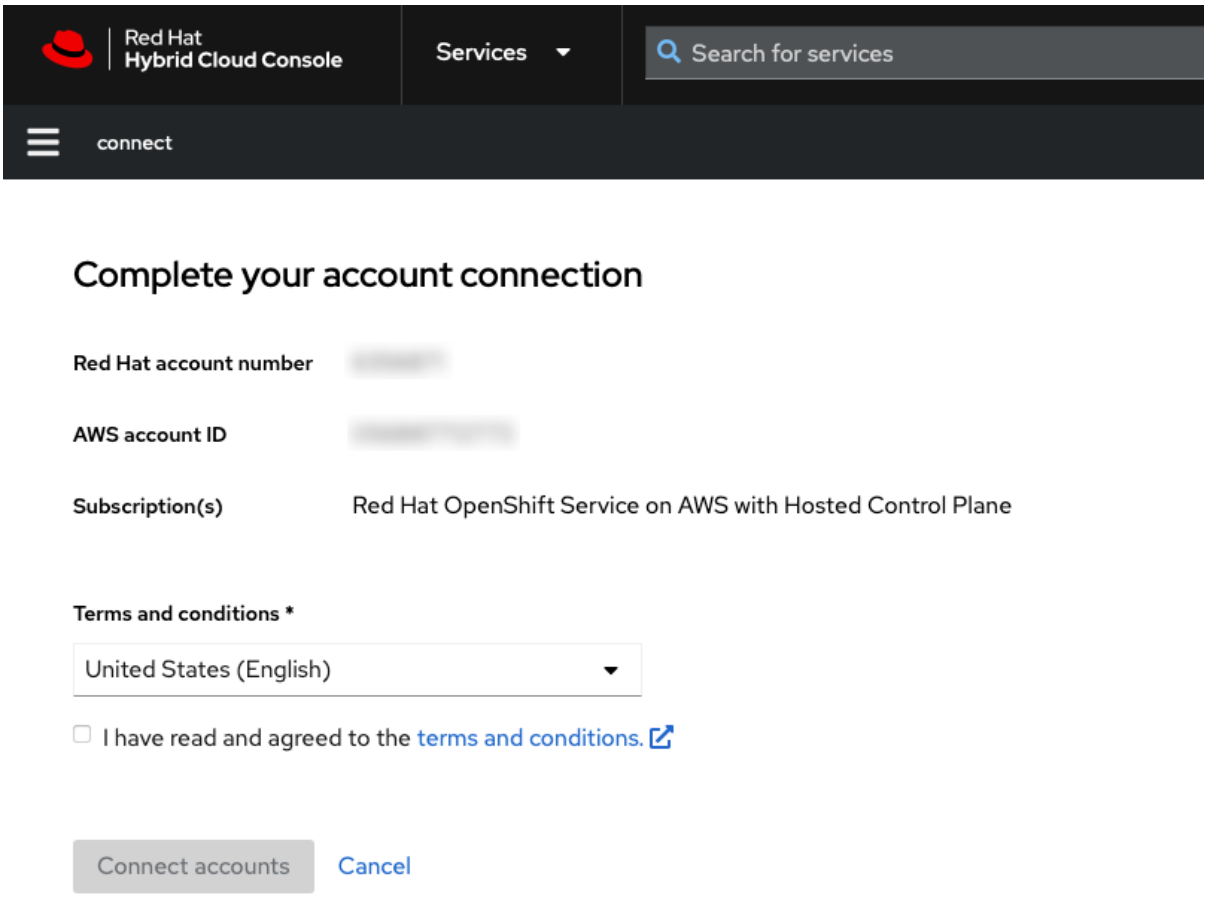
Red Hat OpenShift Service on AWS のプライベートオファーは、AWS License Manager を通じて AWS にリンクされたアカウントと共有することはできません。

2. Red Hat OpenShift Service on AWS クラスターをデプロイするユーザーを追加します。Red Hat アカウントのユーザー管理タスクの詳細は、こちらのユーザー管理に関する FAQ を参照してください。

3. すでにログインしている Red Hat アカウントに、承認されたプライベートオファーを利用し、Red Hat OpenShift Service on AWS クラスターをデプロイするユーザーがすべて含まれていることを確認します。

4. Red Hat アカウント番号と AWS アカウント ID が、リンクさせる目的のアカウントであることを確認します。このリンクは一意であり、Red Hat アカウントと連携できるのは 1 つの AWS (請求先) アカウントだけです。

図3.9 AWS と Red Hat アカウントの連携



5. AWS アカウントを、このページの図 9 に示されているもの以外の Red Hat アカウントにリンクする場合は、アカウントを連携する前に Red Hat Hybrid Cloud Console からログアウトし、すでに承認されているプライベートオファー URL に戻ってアカウントの設定手順を再度実行します。
AWS アカウントと連携できるのは、1 つの Red Hat アカウントだけです。Red Hat アカウントと AWS アカウントを連携すると、ユーザーはこれを変更できなくなります。変更が必要な場合、ユーザーはサポートチケットを作成する必要があります。

6. 利用規約に同意して、**Connect accounts** をクリックします。

3.3. AWS 請求先アカウントの選択

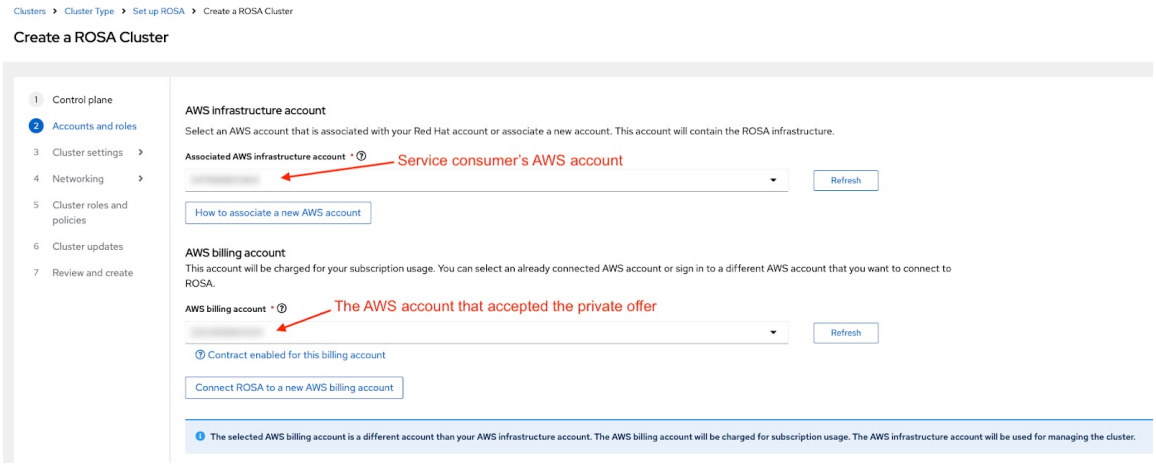
- Red Hat OpenShift Service on AWS クラスターをデプロイする際に、プライベートオファーを承認した AWS 請求先アカウントをエンドユーザーが選択していることを確認してください。

- Web インターフェイスを使用して Red Hat OpenShift Service on AWS をデプロイする場合、"Associated AWS infrastructure account" は、通常、作成するクラスターの管理者が使用する AWS アカウント ID に設定します。

  - これは、請求 AWS アカウントと同じ AWS アカウントにすることができます。

  - このアカウントに AWS リソースがデプロイされます。これにより、そのリソースに関連するすべての請求が処理されます。

図3.10 Red Hat OpenShift Service on AWS クラスターのデプロイ時のインフラストラクチャーおよび請求先 AWS アカウントの選択



- 購入したクォータを作成中のクラスターで使用することを想定している場合は、上記のスクリーンショットの AWS 請求先アカウントのドロップダウンを、プライベートオファーを承認した AWS アカウントに設定する必要があります。インフラストラクチャーと請求先の "ロール" として別々の AWS アカウントを選択した場合は、図 10 のような青色の注記が表示されます。

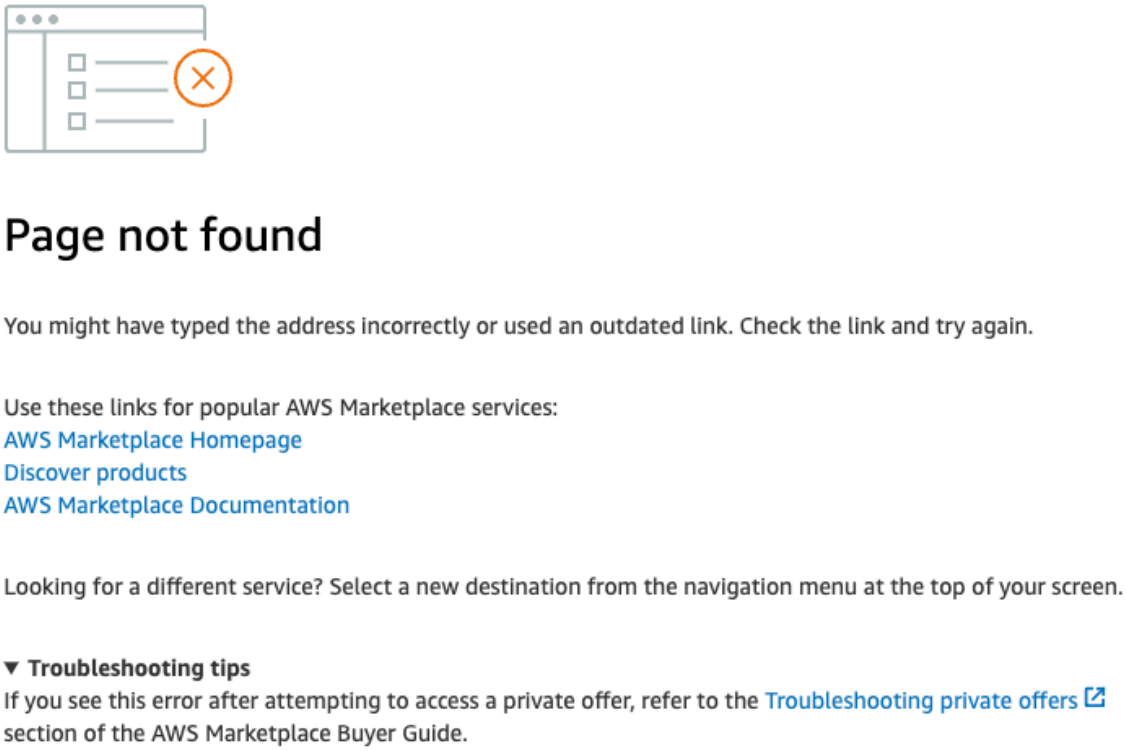
3.4. トラブルシューティング

プライベートオファーの承認と Red Hat アカウントのリンクに関連する最もよくある問題を紹介します。

3.4.1. 別の AWS アカウントを使用してプライベートオファーにアクセスする

- オファーで定義されていない AWS アカウント ID でログインしてプライベートオファーにアクセスしようとし、図 11 のようなメッセージが表示された場合は、目的の AWS 請求先アカウントとしてログインしていることを確認してください。

図3.11 プライベートオファー URL 使用時の HTTP 404 エラー

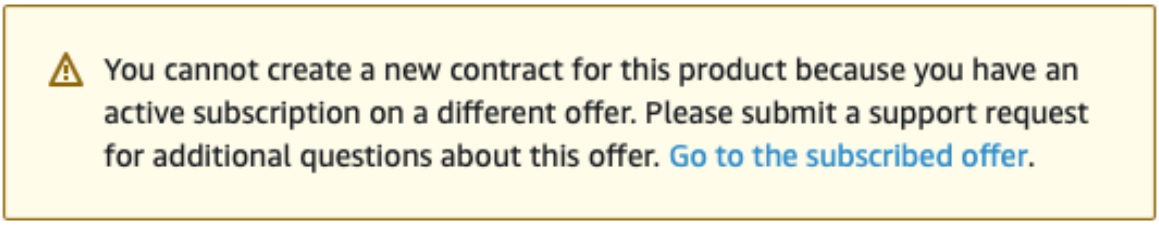


- プライベートオファーを別の AWS アカウントに拡張する必要がある場合は、販売者に問い合わせてください。

3.4.2. アクティブなサブスクリプションのため、プライベートオファーを承認できない

- 別のパブリックオファーまたはプライベートオファーを使用して Red Hat OpenShift Service on AWS をすでにアクティブ化している状態で、Red Hat OpenShift Service on AWS を初めてアクティブ化するために作成されたプライベートオファーにアクセスしようとして次の通知が表示された場合は、オファーを提供した販売者に問い合わせてください。
販売者は、以前のサブスクリプションをキャンセルする必要なく、新しいオファーを提供し、現在の同意をシームレスに置き換えることができます。

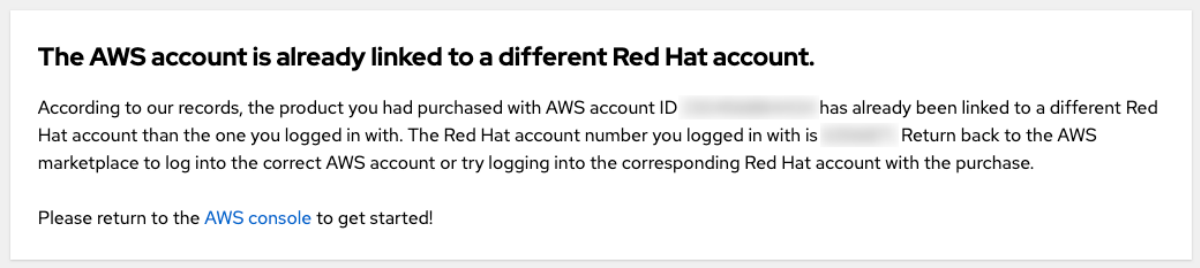
図3.12 既存のサブスクリプションによりプライベートオファーの承認が妨げられる



3.4.3. AWS アカウントがすでに別の Red Hat アカウントにリンクされている

- プライベートオファーを承認した AWS アカウントを現在ログイン中の Red Hat ユーザーに接続しようとしたときに、"AWS account is already linked to a different Red Hat account" というエラーメッセージが表示された場合、その AWS アカウントはすでに別の Red Hat ユーザーと連携しています。

図3.13 AWS アカウントがすでに別の Red Hat アカウントにリンクされている



- 別の Red Hat アカウントか別の AWS アカウントを使用すると、ログインできます。
  - ただし、このガイドはプライベートオファーに関するものであるため、購入者として指定された AWS アカウントでログインし、すでにプライベートオファーを承認していることを想定しています。そのため、このアカウントを請求先アカウントとして使用することを想定しています。プライベートオファーを承認した後、別の AWS アカウントとしてログインすることは想定していません。
- プライベートオファーを承認した AWS アカウントにすでに連携している別の Red Hat ユーザーでログインすることもできます。同じ Red Hat 組織に属する他の Red Hat ユーザーは、図 10 に示すように、クラスターを作成するときに、リンクされた AWS アカウントを Red Hat OpenShift Service on AWS の AWS 請求先アカウントとして使用できます。
- 既存のアカウントのリンクが正しくないと思われる場合は、以下の "チームメンバーが別の Red Hat 組織に属している" という質問を参照して、対処方法のヒントを確認してください。

3.4.4. チームメンバーが別の Red Hat 組織に属している

- AWS アカウントと連携できるのは、1 つの Red Hat アカウントだけです。クラスターを作成し、この AWS アカウントに付与されたプライベートオファーを利用するユーザーは、同じ Red Hat アカウントに属している必要があります。これを実現するには、ユーザーを同じ Red Hat アカウントに招待し、新しい Red Hat ユーザーを作成します。

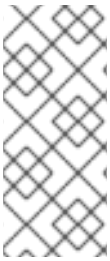
3.4.5. クラスターの作成時に間違った AWS 請求先アカウントを選択した

- ユーザーが間違った AWS 請求先アカウントを選択した場合、これを修正する最も早い方法は、クラスターを削除し、正しい AWS 請求先アカウントを選択して新しいクラスターを作成することです。
- 簡単に削除できない実稼働クラスターの場合は、Red Hat サポートに問い合わせて、既存のクラスターの請求先アカウントを変更してください。この問題を解決するには、ある程度の時間がかかることが予想されます。

第4章 チュートリアル: カスタム DNS リゾルバーを使用した RED HAT OPENSHIFT SERVICE ON AWS のデプロイ

カスタムの DHCP オプションセットを使用 すると、独自の DNS サーバー、ドメイン名などを使用して VPC をカスタマイズできます。Red Hat OpenShift Service on AWS クラスターは、カスタムの DHCP オプションセットの使用をサポートしています。デフォルトでは、Red Hat OpenShift Service on AWS クラスターの場合、クラスターの作成と操作を正常に行うために、"domain name servers" オプションを **AmazonProvidedDNS** に設定する必要があります。DNS 解決にカスタム DNS サーバーを使用するお客様は、Red Hat OpenShift Service on AWS クラスターの作成と操作を正常に実行できるように、追加の設定を行う必要があります。

このチュートリアルでは、特定の DNS ゾーン (詳細は後述) の DNS ルックアップを Amazon Route 53 Inbound Resolver に転送するように DNS サーバーを設定します。



注記

このチュートリアルでは、オープンソースの BIND DNS サーバー (**named**) を使用して、Red Hat OpenShift Service on AWS クラスターのデプロイ先の VPC にある Amazon Route 53 Inbound Resolver に DNS ルックアップを転送するために必要な設定を示します。ゾーン転送を設定する方法は、希望する DNS サーバーのドキュメントを参照してください。

4.1. 前提条件

- ROSA CLI (**rosa**)
- AWS CLI (**aws**)
- 手動で作成した AWS VPC
- カスタム DNS サーバーを参照するように設定され、VPC のデフォルトとして設定されている DHCP オプションセット

4.2. 環境の設定

1. 次の環境変数を設定します。

```
$ export VPC_ID=<vpc_ID> 1
$ export REGION=<region> 2
$ export VPC_CIDR=<vpc_CIDR> 3
```

1. **<vpc_ID>** は、クラスターをインストールする VPC の ID に置き換えます。
2. **<region>** は、クラスターをインストールする AWS リージョンに置き換えます。
3. **<vpc_CIDR>** は、VPC の CIDR 範囲に置き換えます。

2. 次のセクションに進む前に、すべてのフィールドが正しく出力されていることを確認してください。

```
$ echo "VPC ID: ${VPC_ID}, VPC CIDR Range: ${VPC_CIDR}, Region: ${REGION}"
```

4.3. AMAZON ROUTE 53 INBOUND RESOLVER の作成

次の手順を使用して、クラスターのデプロイ先の VPC に Amazon Route 53 Inbound Resolver をデプロイします。

警告

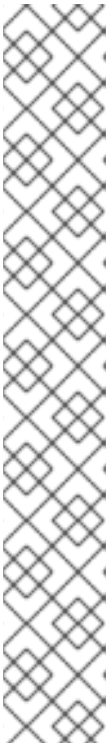
この例では、クラスターが使用するのと同じ VPC に Amazon Route 53 Inbound Resolver をデプロイします。別の VPC にデプロイする場合は、**クラスターの作成を開始してから**、以下に詳述するプライベートホストゾーンを手動で関連付ける必要があります。クラスター作成プロセスを開始する前にゾーンを関連付けることはできません。クラスター作成プロセス中にプライベートホストゾーンを関連付けないと、クラスター作成が失敗します。

1. セキュリティーグループを作成し、VPC からポート **53/tcp** および **53/udp** へのアクセスを許可します。

```
$ SG_ID=$(aws ec2 create-security-group --group-name rosa-inbound-resolver --description "Security group for ROSA inbound resolver" --vpc-id ${VPC_ID} --region ${REGION} --output text)
$ aws ec2 authorize-security-group-ingress --group-id ${SG_ID} --protocol tcp --port 53 --cidr ${VPC_CIDR} --region ${REGION}
$ aws ec2 authorize-security-group-ingress --group-id ${SG_ID} --protocol udp --port 53 --cidr ${VPC_CIDR} --region ${REGION}
```

2. VPC に Amazon Route 53 Inbound Resolver を作成します。

```
$ RESOLVER_ID=$(aws route53resolver create-resolver-endpoint \
  --name rosa-inbound-resolver \
  --creator-request-id rosa-$(date '+%Y-%m-%d') \
  --security-group-ids ${SG_ID} \
  --direction INBOUND \
  --ip-addresses $(aws ec2 describe-subnets --filter Name=vpc-id,Values=${VPC_ID} --region ${REGION} | jq -jr '.Subnets | map("SubnetId=\(.SubnetId) ") | .[]') \
  --region ${REGION} \
  --output text \
  --query 'ResolverEndpoint.Id')
```



注記

上記のコマンドは、動的に割り当てられた IP アドレスを使用して、指定した VPC 内の **すべてのサブネット** に Amazon Route 53 Inbound Resolver エンドポイントをアタッチします。サブネットや IP アドレスを手動で指定する場合は、代わりに次のコマンドを実行します。

```
$ RESOLVER_ID=$(aws route53resolver create-resolver-endpoint \
  --name rosa-inbound-resolver \
  --creator-request-id rosa-$(date '+%Y-%m-%d') \
  --security-group-ids ${SG_ID} \
  --direction INBOUND \
  --ip-addresses SubnetId=<subnet_ID>,Ip=<endpoint_IP> SubnetId=
<subnet_ID>,Ip=<endpoint_IP> \ 1
  --region ${REGION} \
  --output text \
  --query 'ResolverEndpoint.Id')
```

1 **<subnet_ID>** はサブネット ID に、**<endpoint_IP>** はインバウンドリゾルバーエンドポイントを追加する静的 IP アドレスに置き換えます。

3. DNS サーバー設定で指定するインバウンドリゾルバーエンドポイントの IP アドレスを取得します。

```
$ aws route53resolver list-resolver-endpoint-ip-addresses \
  --resolver-endpoint-id ${RESOLVER_ID} \
  --region=${REGION} \
  --query 'IpAddresses[*].Ip'
```

出力例

```
[
    "10.0.45.253",
    "10.0.23.131",
    "10.0.148.159"
]
```

4.4. DNS サーバーの設定

次の手順を使用して、必要なプライベートホストゾーンを Amazon Route 53 Inbound Resolver に転送するように DNS サーバーを設定します。

4.4.1. Red Hat OpenShift Service on AWS

Red Hat OpenShift Service on AWS クラスターでは、2 つのプライベートホストゾーンの DNS 転送を設定する必要があります。

- **<cluster-name>.hypershift.local**
- **rosa.<domain-prefix>.<unique-ID>.p3.openshiftapps.com**

これらの Amazon Route 53 プライベートホストゾーンは、クラスターの作成中に作成されます。**cluster-name** と **domain-prefix** はお客様が指定する値ですが、**unique-ID** はクラスターの作成中

にランダムに生成され、事前に選択することはできません。そのため、クラスター作成プロセスが開始してから、**p3.openshiftapps.com** プライベートホストゾーンの転送を設定する必要があります。

1. クラスターを作成する前に、**<cluster-name>.hypershift.local** のすべての DNS 要求を Amazon Route 53 Inbound Resolver エンドポイントに転送するように DNS サーバーを設定します。BIND DNS サーバーの場合は、任意のテキストエディターで **/etc/named.conf** ファイルを編集し、以下の例を使用して新しいゾーンを追加します。

   例

```
zone "<cluster-name>.hypershift.local" { (1)
  type forward;
  forward only;
  forwarders { (2)
    10.0.45.253;
    10.0.23.131;
    10.0.148.159;
  };
};
```

   (1) **<cluster-name>** は、実際の Red Hat OpenShift Service on AWS クラスター名に置き換えます。

   (2) 上記で取得したインバウンドリゾルバーエンドポイントの IP アドレスに置き換えます。各 IP アドレスの後に必ず **;** を付けます。

2. クラスターを作成します。

3. クラスターの作成プロセスが開始したら、新しく作成されたプライベートホストゾーンを特定します。

```
$ aws route53 list-hosted-zones-by-vpc \
  --vpc-id ${VPC_ID} \
  --vpc-region ${REGION} \
  --query 'HostedZoneSummaries[*].Name' \
  --output table
```

   出力例

```
----------------------------------------------
|            ListHostedZonesByVPC           |
+--------------------------------------------+
|  rosa.domain-prefix.lkmb.p3.openshiftapps.com. |
|  cluster-name.hypershift.local.            |
+--------------------------------------------+
```

注記

クラスター作成プロセスで Route 53 にプライベートホストゾーンが作成されるまでに、数分かかる場合があります。**p3.openshiftapps.com** ドメインが表示されない場合は、数分待ってからコマンドを再度実行してください。

4. クラスタードメインの一意の ID を調べたら、**rosa.<domain-prefix>.<unique-ID>.p3.openshiftapps.com** のすべての DNS 要求を Amazon Route 53 Inbound Resolver エンドポイントに転送するように DNS サーバーを設定します。BIND DNS サーバーの場合は、任意のテキストエディターで **/etc/named.conf** ファイルを編集し、以下の例を使用して新しいゾーンを追加します。

   例

```
zone "rosa.<domain-prefix>.<unique-ID>.p3.openshiftapps.com" { (1)
  type forward;
  forward only;
  forwarders { (2)
    10.0.45.253;
    10.0.23.131;
    10.0.148.159;
  };
};
```

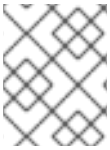
(1) **<domain-prefix>** はクラスターのドメイン接頭辞に、**<unique-ID>** は上記で取得した一意の ID に置き換えます。

(2) 上記で取得したインバウンドリゾルバーエンドポイントの IP アドレスに置き換えます。各 IP アドレスの後に必ず **;** を付けます。

第5章 チュートリアル: AWS WAF と AMAZON CLOUDFRONT を使用した RED HAT OPENSHIFT SERVICE ON AWS ワークロードの保護

AWS WAF は Web アプリケーションファイアウォールです。保護対象の Web アプリケーションリソースに転送される HTTP および HTTPS 要求を監視できます。

Amazon CloudFront を使用して、Web Application Firewall (WAF) を Red Hat OpenShift Service on AWS ワークロードに追加できます。外部ソリューションを使用すると、WAF の処理によるサービス拒否から Red Hat OpenShift Service on AWS リソースを保護できます。



注記

WAFv1 (WAF クラシック) はサポートされなくなりました。WAFv2 を使用してください。

5.1. 前提条件

- Red Hat OpenShift Service on AWS クラスター。
- OpenShift CLI (**oc**) にアクセスできる。
- AWS CLI (**aws**) にアクセスできる。

5.1.1. 環境設定

- 環境変数を準備します。

```
$ export DOMAIN=apps.example.com 1
$ export AWS_PAGER=""
$ export CLUSTER_NAME=$(oc get infrastructure cluster -o=jsonpath="
{.status.infrastructureName}"  | sed 's/-[a-z0-9]\{5\}$//')
$ export REGION=$(oc get infrastructure cluster -o=jsonpath="
{.status.platformStatus.aws.region}")
$ export AWS_ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
$ export SCRATCH="/tmp/${CLUSTER}/cloudfront-waf"
$ mkdir -p ${SCRATCH}
$ echo "Cluster: ${CLUSTER}, Region: ${REGION}, AWS Account ID:
${AWS_ACCOUNT_ID}"
```

1 **IngressController** に使用するカスタムドメインに置き換えます。

注記

前のコマンドからの "Cluster" の出力は、クラスターの名前、クラスターの内部 ID、またはクラスターのドメイン接頭辞である可能性があります。別の識別子を使用する場合は、次のコマンドを実行して手動でこの値を設定できます。

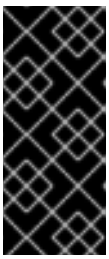
```
$ export CLUSTER=my-custom-value
```

5.2. セカンダリー INGRESS CONTROLLER の設定

標準 (およびデフォルト) クラスター Ingress Controller からの WAF 保護対象の外部トラフィックをセグメント化するために、セカンダリー Ingress Controller を設定する必要があります。

前提条件

- カスタムドメイン用の公的に信頼された SAN 証明書またはワイルドカード証明書 (例: **CN=*.apps.example.com**)



重要

Amazon CloudFront は HTTPS を使用してクラスターのセカンダリー Ingress Controller と通信します。Amazon CloudFront のドキュメント で説明されているように、CloudFront とクラスター間の HTTPS 通信には自己署名証明書を使用できません。Amazon CloudFront は、証明書が信頼できる認証局によって発行されたことを確認します。

手順

1. 秘密鍵と公開証明書から新しい TLS シークレットを作成します。**fullchain.pem** は完全なワイルドカード証明書チェーン (中間証明書を含む)、**privkey.pem** はワイルドカード証明書の秘密鍵です。

   例

   ```
   $ oc -n openshift-ingress create secret tls waf-tls --cert=fullchain.pem --key=privkey.pem
   ```

2. 新規の **IngressController** リソースを作成します。

   waf-ingress-controller.yaml の例

   ```
   apiVersion: operator.openshift.io/v1
   kind: IngressController
   metadata:
     name: cloudfront-waf
     namespace: openshift-ingress-operator
   spec:
     domain: apps.example.com (1)
     defaultCertificate:
       name: waf-tls
     endpointPublishingStrategy:
       loadBalancer:
         dnsManagementPolicy: Unmanaged
         providerParameters:
           aws:
             type: NLB
           type: AWS
         scope: External
       type: LoadBalancerService
     routeSelector: (2)
       matchLabels:
        route: waf
   ```

1. **IngressController** に使用するカスタムドメインに置き換えます。
2. Ingress Controller によって提供されるルートのセットをフィルタリングします。このチュートリアルでは **waf** ルートセレクターを使用しますが、値を指定しないと、フィルター処理は行われません。

3. **IngressController** を適用します。

例

```
$ oc apply -f waf-ingress-controller.yaml
```

4. IngressController が外部ロードバランサーを正常に作成したことを確認します。

```
$ oc -n openshift-ingress get service/router-cloudfront-waf
```

出力例

```
NAME                  TYPE          CLUSTER-IP      EXTERNAL-IP
PORT(S)                      AGE
router-cloudfront-waf   LoadBalancer   172.30.16.141
a68a838a7f26440bf8647809b61c4bc8-4225395f488830bd.elb.us-east-1.amazonaws.com
80:30606/TCP,443:31065/TCP   2m19s
```

5.2.1. AWS WAF の設定

AWS WAF サービスは Web アプリケーションファイアウォールです。Red Hat OpenShift Service on AWS などの保護対象の Web アプリケーションリソースに転送される HTTP および HTTPS 要求を監視、保護、制御できます。

1. Web ACL に適用する AWS WAF ルールファイルを作成します。

```
$ cat << EOF > ${SCRATCH}/waf-rules.json
[
  {
    "Name": "AWS-AWSManagedRulesCommonRuleSet",
    "Priority": 0,
    "Statement": {
      "ManagedRuleGroupStatement": {
        "VendorName": "AWS",
        "Name": "AWSManagedRulesCommonRuleSet"
      }
    },
    "OverrideAction": {
      "None": {}
    },
    "VisibilityConfig": {
      "SampledRequestsEnabled": true,
      "CloudWatchMetricsEnabled": true,
      "MetricName": "AWS-AWSManagedRulesCommonRuleSet"
    }
  },
  {
```

```
    "Name": "AWS-AWSManagedRulesSQLiRuleSet",
    "Priority": 1,
    "Statement": {
      "ManagedRuleGroupStatement": {
        "VendorName": "AWS",
        "Name": "AWSManagedRulesSQLiRuleSet"
      }
    },
    "OverrideAction": {
      "None": {}
    },
    "VisibilityConfig": {
      "SampledRequestsEnabled": true,
      "CloudWatchMetricsEnabled": true,
      "MetricName": "AWS-AWSManagedRulesSQLiRuleSet"
    }
  }
]
EOF
```

これにより、コア (共通) および SQL AWS マネージドルールセットが有効になります。

2. 上記で指定したルールを使用して、AWS WAF の Web ACL を作成します。

```
$ WAF_WACL=$(aws wafv2 create-web-acl \
  --name cloudfront-waf \
  --region ${REGION} \
  --default-action Allow={} \
  --scope CLOUDFRONT \
  --visibility-config
SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=${CLUSTER}-
waf-metrics \
  --rules file://${SCRATCH}/waf-rules.json \
  --query 'Summary.Name' \
  --output text)
```

5.3. AMAZON CLOUDFRONT の設定

1. 新しく作成されたカスタム Ingress Controller の NLB ホスト名を取得します。

```
$ NLB=$(oc -n openshift-ingress get service router-cloudfront-waf \
  -o jsonpath='{.status.loadBalancer.ingress[0].hostname}')
```

2. 証明書を Amazon Certificate Manager にインポートします。**cert.pem** はワイルドカード証明書、**fullchain.pem** はワイルドカード証明書のチェーン、**privkey.pem** はワイルドカード証明書の秘密鍵です。

注記

この証明書は、クラスターがデプロイされているリージョンに関係なく **us-east-1** にインポートする必要があります。Amazon CloudFront はグローバル AWS サービスであるためです。

例

```
$ aws acm import-certificate --certificate file://cert.pem \
  --certificate-chain file://fullchain.pem \
  --private-key file://privkey.pem \
  --region us-east-1
```

3. AWS コンソール にログインして、CloudFront ディストリビューションを作成します。
4. 次の情報を使用して、CloudFront ディストリビューションを設定します。

注記

以下の表でオプションが指定されていない場合は、デフォルトのままにしておきます (空白でも構いません)。

オプション	値
Origin domain	前のコマンドからの出力 [1]
名前	rosa-waf-ingress [2]
Viewer protocol policy	Redirect HTTP to HTTPS
Allowed HTTP methods	GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE
Cache policy	CachingDisabled
Origin request policy	AllViewer
Web Application Firewall (WAF)	Enable security protections
Use existing WAF configuration	true
Choose a web ACL	**cloudfront-waf**
Alternate domain name (CNAME)	*.apps.example.com [3]
Custom SSL certificate	上のステップでインポートした証明書を選択 [4]

1. origin domain を取得するには、**echo ${NLB}** を実行します。
2. 複数のクラスターがある場合は、オリジンの名前が一意であることを確認してください。
3. これは、カスタム Ingress Controller の作成に使用したワイルドカードドメインと同じである必要があります。
4. これは、上で入力した alternate domain name と同じである必要があります。

5. Amazon CloudFront ディストリビューションエンドポイントを取得します。

```
$ aws cloudfront list-distributions --query "DistributionList.Items[?Origins.Items[?
DomainName=='${NLB}']].DomainName" --output text
```

6. CNAME を持つカスタムのワイルドカードドメインの DNS を、前述のステップの Amazon CloudFront ディストリビューションエンドポイントに更新します。

例

```
*.apps.example.com CNAME d1b2c3d4e5f6g7.cloudfront.net
```

5.4. サンプルアプリケーションのデプロイ

1. 次のコマンドを実行して、サンプルアプリケーション用の新しいプロジェクトを作成します。

```
$ oc new-project hello-world
```

2. Hello World アプリケーションをデプロイします。

```
$ oc -n hello-world new-app --image=docker.io/openshift/hello-openshift
```

3. カスタムドメイン名を指定してアプリケーションのルートを作成します。

例

```
$ oc -n hello-world create route edge --service=hello-openshift hello-openshift-tls \
--hostname hello-openshift.${DOMAIN}
```

4. ルートにラベルを付けて、カスタム Ingress Controller へのアクセスを許可します。

```
$ oc -n hello-world label route.route.openshift.io/hello-openshift-tls route=waf
```

5.5. WAF のテスト

1. アプリが Amazon CloudFront の背後でアクセスできることをテストします。

例

```
$ curl "https://hello-openshift.${DOMAIN}"
```

出力例

```
Hello OpenShift!
```

2. WAF が不正な要求を拒否することをテストします。

例

```
$ curl -X POST "https://hello-openshift.${DOMAIN}" \
  -F "user='<script><alert>Hello></alert></script>'"
```

出力例

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<HTML><HEAD><META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-
8859-1">
<TITLE>ERROR: The request could not be satisfied</TITLE>
</HEAD><BODY>
<H1>403 ERROR</H1>
<H2>The request could not be satisfied.</H2>
<HR noshade size="1px">
Request blocked.
We can't connect to the server for this app or website at this time. There might be too much
traffic or a configuration error. Try again later, or contact the app or website owner.
<BR clear="all">
If you provide content to customers through CloudFront, you can find steps to troubleshoot
and help prevent this error by reviewing the CloudFront documentation.
<BR clear="all">
<HR noshade size="1px">
<PRE>
Generated by cloudfront (CloudFront)
Request ID: nFk9q2yB8jddI6FZOTjdliexzx-FwZtr8xUQUNT75HThPlrALDxbag==
</PRE>
<ADDRESS>
</ADDRESS>
</BODY></HTML>
```

予期される結果は **403 ERROR** エラーです。このエラーが返されれば、アプリケーションは AWS WAF によって保護されています。

5.6. 関連情報

- YouTube の Adding Extra Security with AWS WAF, CloudFront and ROSA | Amazon Web Services

第6章 チュートリアル: AWS WAF と AWS ALB を使用した RED HAT OPENSHIFT SERVICE ON AWS ワークロードの保護

AWS WAF は Web アプリケーションファイアウォールです。保護対象の Web アプリケーションリソースに転送される HTTP および HTTPS 要求を監視できます。

AWS Application Load Balancer (ALB) を使用して、Red Hat OpenShift Service on AWS ワークロードに Web Application Firewall (WAF) を追加できます。外部ソリューションを使用すると、WAF の処理によるサービス拒否から Red Hat OpenShift Service on AWS リソースを保護できます。

重要

ALB ベースのソリューションを絶対に使用する必要がある場合を除き、より柔軟な CloudFront メソッド を使用することを推奨します。

6.1. 前提条件

- 複数のアベイラビリティーゾーン (AZ) にまたがる Red Hat OpenShift Service on AWS クラスター。

注記

AWS ドキュメントによると、AWS ALB には複数の AZ にまたがる 2 つ以上の **パブリック** サブネットが必要です。このため、ALB を使用できるのは、複数の AZ にまたがる Red Hat OpenShift Service on AWS クラスターに限られます。

- OpenShift CLI (**oc**) にアクセスできる。
- AWS CLI (**aws**) にアクセスできる。

6.1.1. 環境設定

- 環境変数を準備します。

```
$ export AWS_PAGER=""
$ export CLUSTER=$(oc get infrastructure cluster -o=jsonpath="
{.status.infrastructureName}")
$ export REGION=$(oc get infrastructure cluster -o=jsonpath="
{.status.platformStatus.aws.region}")
$ export OIDC_ENDPOINT=$(oc get authentication.config.openshift.io cluster -o
jsonpath='{.spec.serviceAccountIssuer}' | sed  's|^https://||')
$ export AWS_ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
$ export SCRATCH="/tmp/${CLUSTER}/alb-waf"
$ mkdir -p ${SCRATCH}
$ echo "Cluster: $(echo ${CLUSTER} | sed 's/-[a-z0-9]\{5\}$//'), Region: ${REGION}, OIDC
Endpoint: ${OIDC_ENDPOINT}, AWS Account ID: ${AWS_ACCOUNT_ID}"
```

6.1.2. AWS VPC とサブネット

注記

このセクションは、既存の VPC にデプロイされたクラスターにのみ適用されます。クラスターを既存の VPC にデプロイしなかった場合は、このセクションをスキップして、その後のインストールセクションに進んでください。

1. 以下の変数を、実際の Red Hat OpenShift Service on AWS デプロイメントに合わせて適切な値に設定します。

```
$ export VPC_ID=<vpc-id> 1
$ export PUBLIC_SUBNET_IDS=(<space-separated-list-of-ids>) 2
$ export PRIVATE_SUBNET_IDS=(<space-separated-list-of-ids>) 3
```

1. クラスターの VPC ID に置き換えます (例: **export VPC_ID=vpc-04c429b7dbc4680ba**)。
2. クラスターのプライベートサブネット ID のスペース区切りリストに置き換えます。**()** は必ず保持してください。例: **export PUBLIC_SUBNET_IDS=(subnet-056fd6861ad332ba2 subnet-08ce3b4ec753fe74c subnet-071aa28228664972f)**。
3. クラスターのプライベートサブネット ID のスペース区切りリストに置き換えます。**()** は必ず保持してください。たとえば、**export PRIVATE_SUBNET_IDS=(subnet-0b933d72a8d72c36a subnet-0817eb72070f1d3c2 subnet-0806e64159b66665a)** です。

2. クラスター識別子を使用して、クラスターの VPC にタグを追加します。

```
$ aws ec2 create-tags --resources ${VPC_ID} \
  --tags Key=kubernetes.io/cluster/${CLUSTER},Value=shared --region ${REGION}
```

3. パブリックサブネットにタグを追加します。

```
$ aws ec2 create-tags \
  --resources ${PUBLIC_SUBNET_IDS} \
  --tags Key=kubernetes.io/role/elb,Value='1' \
        Key=kubernetes.io/cluster/${CLUSTER},Value=shared \
  --region ${REGION}
```

4. プライベートサブネットにタグを追加します。

```
$ aws ec2 create-tags \
  --resources ${PRIVATE_SUBNET_IDS} \
  --tags Key=kubernetes.io/role/internal-elb,Value='1' \
        Key=kubernetes.io/cluster/${CLUSTER},Value=shared \
  --region ${REGION}
```

6.2. AWS LOAD BALANCER OPERATOR のデプロイ

AWS Load Balancer Operator は、Red Hat OpenShift Service on AWS クラスター内の **aws-load-balancer-controller** のインスタンスをインストール、管理、設定するために使用します。Red Hat OpenShift Service on AWS に ALB をデプロイするには、まず AWS Load Balancer Operator をデプロイする必要があります。

1. 次のコマンドを実行して、AWS Load Balancer Operator をデプロイする新しいプロジェクトを作成します。

```
$ oc new-project aws-load-balancer-operator
```

2. 次のコマンドを実行して、AWS Load Balancer Controller の AWS IAM ポリシーを作成します (まだ存在しない場合)。

注記

ポリシーは アップストリームの AWS Load Balancer Controller ポリシー から取得されます。これは Operator が機能するために必要です。

```
$ POLICY_ARN=$(aws iam list-policies --query \
    "Policies[?PolicyName=='aws-load-balancer-operator-policy'].{ARN:Arn}" \
    --output text)
```

```
$ if [[ -z "${POLICY_ARN}" ]]; then
   wget -O "${SCRATCH}/load-balancer-operator-policy.json" \
     https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/main/docs/install/iam_policy.json
    POLICY_ARN=$(aws --region "$REGION" --query Policy.Arn \
    --output text iam create-policy \
    --policy-name aws-load-balancer-operator-policy \
    --policy-document "file://${SCRATCH}/load-balancer-operator-policy.json")
fi
```

3. AWS Load Balancer Operator の AWS IAM 信頼ポリシーを作成します。

```
$ cat <<EOF > "${SCRATCH}/trust-policy.json"
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Condition": {
   "StringEquals" : {
    "${OIDC_ENDPOINT}:sub": ["system:serviceaccount:aws-load-balancer-operator:aws-
load-balancer-operator-controller-manager", "system:serviceaccount:aws-load-balancer-
operator:aws-load-balancer-controller-cluster"]
   }
 },
 "Principal": {
   "Federated": "arn:aws:iam::$AWS_ACCOUNT_ID:oidc-provider/${OIDC_ENDPOINT}"
 },
 "Action": "sts:AssumeRoleWithWebIdentity"
 }
 ]
}
EOF
```

4. AWS Load Balancer Operator の AWS IAM ロールを作成します。

```
$ ROLE_ARN=$(aws iam create-role --role-name "${CLUSTER}-alb-operator" \
```

```
    --assume-role-policy-document "file://${SCRATCH}/trust-policy.json" \
    --query Role.Arn --output text)
```

5. 次のコマンドを実行して、以前に作成した IAM ロールに AWS Load Balancer Operator ポリシーを割り当てます。

```
$ aws iam attach-role-policy --role-name "${CLUSTER}-alb-operator" \
     --policy-arn ${POLICY_ARN}
```

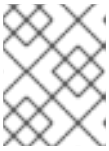
6. 新しく作成した AWS IAM ロールを引き受けるための AWS Load Balancer Operator 用のシークレットを作成します。

```
$ cat << EOF | oc apply -f -
apiVersion: v1
kind: Secret
metadata:
  name: aws-load-balancer-operator
  namespace: aws-load-balancer-operator
stringData:
  credentials: |
    [default]
    role_arn = ${ROLE_ARN}
    web_identity_token_file = /var/run/secrets/openshift/serviceaccount/token
EOF
```

7. AWS Load Balancer Operator をインストールします。

```
$ cat << EOF | oc apply -f -
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: aws-load-balancer-operator
  namespace: aws-load-balancer-operator
spec:
  upgradeStrategy: Default
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: aws-load-balancer-operator
  namespace: aws-load-balancer-operator
spec:
  channel: stable-v1.0
  installPlanApproval: Automatic
  name: aws-load-balancer-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  startingCSV: aws-load-balancer-operator.v1.0.0
EOF
```

8. 次の Operator を使用して、AWS Load Balancer Controller のインスタンスをデプロイします。



注記

ここでエラーが発生した場合は、少し待ってから再試行してください。エラーが発生するのは、Operator がまだインストールを完了していないためです。

```
$ cat << EOF | oc apply -f -
apiVersion: networking.olm.openshift.io/v1
kind: AWSLoadBalancerController
metadata:
  name: cluster
spec:
  credentials:
    name: aws-load-balancer-operator
  enabledAddons:
    - AWSWAFv2
EOF
```

9. Operator Pod とコントローラー Pod の両方が実行されていることを確認します。

```
$ oc -n aws-load-balancer-operator get pods
```

次のようなメッセージが表示されます。表示されない場合は、少し待ってから再試行してください。

```
NAME                                                READY   STATUS    RESTARTS   AGE
aws-load-balancer-controller-cluster-6ddf658785-pdp5d           1/1     Running   0          99s
aws-load-balancer-operator-controller-manager-577d9ffcb9-w6zqn   2/2     Running   0
2m4s
```

6.3. サンプルアプリケーションのデプロイ

1. サンプルアプリケーション用に新しいプロジェクトを作成します。

```
$ oc new-project hello-world
```

2. Hello World アプリケーションをデプロイします。

```
$ oc new-app -n hello-world --image=docker.io/openshift/hello-openshift
```

3. 事前に作成済みのサービスリソースを NodePort サービスタイプに変換します。

```
$ oc -n hello-world patch service hello-openshift -p '{"spec":{"type":"NodePort"}}'
```

4. AWS Load Balancer Operator を使用して AWS ALB をデプロイします。

```
$ cat << EOF | oc apply -f -
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: hello-openshift-alb
  namespace: hello-world
  annotations:
```

```
    alb.ingress.kubernetes.io/scheme: internet-facing
spec:
  ingressClassName: alb
  rules:
    - http:
        paths:
          - path: /
            pathType: Exact
            backend:
              service:
                name: hello-openshift
                port:
                  number: 8080
EOF
```

5. AWS ALB Ingress エンドポイントを curl して、Hello World アプリケーションにアクセスできることを確認します。

注記

AWS ALB のプロビジョニングには数分かかります。**curl: (6) Could not resolve host** というエラーが表示された場合は、待機してから再試行してください。

```
$ INGRESS=$(oc -n hello-world get ingress hello-openshift-alb -o
jsonpath='{.status.loadBalancer.ingress[0].hostname}')
$ curl "http://${INGRESS}"
```

出力例

```
Hello OpenShift!
```

6.3.1. AWS WAF の設定

AWS WAF サービスは Web アプリケーションファイアウォールです。Red Hat OpenShift Service on AWS などの保護対象の Web アプリケーションリソースに転送される HTTP および HTTPS 要求を監視、保護、制御できます。

1. Web ACL に適用する AWS WAF ルールファイルを作成します。

```
$ cat << EOF > ${SCRATCH}/waf-rules.json
[
  {
    "Name": "AWS-AWSManagedRulesCommonRuleSet",
    "Priority": 0,
    "Statement": {
      "ManagedRuleGroupStatement": {
        "VendorName": "AWS",
        "Name": "AWSManagedRulesCommonRuleSet"
      }
    },
    "OverrideAction": {
      "None": {}
    },
```

```
      "VisibilityConfig": {
        "SampledRequestsEnabled": true,
        "CloudWatchMetricsEnabled": true,
        "MetricName": "AWS-AWSManagedRulesCommonRuleSet"
      }
    },
    {
      "Name": "AWS-AWSManagedRulesSQLiRuleSet",
      "Priority": 1,
      "Statement": {
        "ManagedRuleGroupStatement": {
          "VendorName": "AWS",
          "Name": "AWSManagedRulesSQLiRuleSet"
        }
      },
      "OverrideAction": {
        "None": {}
      },
      "VisibilityConfig": {
        "SampledRequestsEnabled": true,
        "CloudWatchMetricsEnabled": true,
        "MetricName": "AWS-AWSManagedRulesSQLiRuleSet"
      }
    }
]
EOF
```

これにより、コア (共通) および SQL AWS マネージドルールセットが有効になります。

2. 上記で指定したルールを使用して、AWS WAF の Web ACL を作成します。

```
$ WAF_ARN=$(aws wafv2 create-web-acl \
  --name ${CLUSTER}-waf \
  --region ${REGION} \
  --default-action Allow={} \
  --scope REGIONAL \
  --visibility-config
SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=${CLUSTER}-
waf-metrics \
  --rules file://${SCRATCH}/waf-rules.json \
  --query 'Summary.ARN' \
  --output text)
```

3. Ingress リソースに AWS WAF の Web ACL ARN のアノテーションを付けます。

```
$ oc annotate -n hello-world ingress.networking.k8s.io/hello-openshift-alb \
  alb.ingress.kubernetes.io/wafv2-acl-arn=${WAF_ARN}
```

4. ルールが反映されるまで 10 秒待ち、アプリケーションがまだ動作するかテストします。

```
$ curl "http://${INGRESS}"
```

出力例

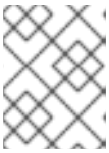
```
Hello OpenShift!
```

5. WAF が不正な要求を拒否することをテストします。

```
$ curl -X POST "http://${INGRESS}" \
  -F "user='<script><alert>Hello></alert></script>'"
```

出力例

```
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
</body>
</html
```



注記

AWS WAF 統合の有効化には数分かかる場合があります。**403 Forbidden** エラーが表示されない場合は、数秒待ってからもう一度お試しください。

予期される結果は **403 Forbidden** エラーです。このエラーが返されれば、アプリケーションは AWS WAF によって保護されています。

6.4. 関連情報

- YouTube の Adding Extra Security with AWS WAF, CloudFront and ROSA | Amazon Web Services

第7章 チュートリアル: RED HAT OPENSHIFT SERVICE ON AWS クラスターへの OPENSHIFT API FOR DATA PROTECTION のデプロイ

重要

このコンテンツは Red Hat のエキスパートが作成したものですが、サポート対象のすべての設定でまだテストされていません。

前提条件

- Red Hat OpenShift Service on AWS クラスター

環境

- 環境変数を準備します。

注記

実際の Red Hat OpenShift Service on AWS クラスターに合わせてクラスター名を変更し、管理者としてクラスターにログインしていることを確認してください。次に進む前に、すべてのフィールドが正しく出力されていることを確認してください。

```
$ export CLUSTER_NAME=$(oc get infrastructure cluster -o=jsonpath="
{.status.infrastructureName}"  | sed 's/-[a-z0-9]\{5\}$//')
$ export ROSA_CLUSTER_ID=$(rosa describe cluster -c ${CLUSTER_NAME} --output json
| jq -r .id)
$ export REGION=$(rosa describe cluster -c ${CLUSTER_NAME} --output json | jq -r
.region.id)
$ export OIDC_ENDPOINT=$(oc get authentication.config.openshift.io cluster -o
jsonpath='{.spec.serviceAccountIssuer}' | sed  's|^https://||')
$ export AWS_ACCOUNT_ID=`aws sts get-caller-identity --query Account --output text`
$ export CLUSTER_VERSION=`rosa describe cluster -c ${CLUSTER_NAME} -o json | jq -r
.version.raw_id | cut -f -2 -d '.'`
$ export ROLE_NAME="${CLUSTER_NAME}-openshift-oadp-aws-cloud-credentials"
$ export AWS_PAGER=""
$ export SCRATCH="/tmp/${CLUSTER_NAME}/oadp"
$ mkdir -p ${SCRATCH}
$ echo "Cluster ID: ${ROSA_CLUSTER_ID}, Region: ${REGION}, OIDC Endpoint:
${OIDC_ENDPOINT}, AWS Account ID: ${AWS_ACCOUNT_ID}"
```

7.1. AWS アカウントの準備

1. S3 アクセスを許可する IAM ポリシーを作成します。

```
$ POLICY_ARN=$(aws iam list-policies --query "Policies[?PolicyName=='RosaOadpVer1'].
{ARN:Arn}" --output text)
if [[ -z "${POLICY_ARN}" ]]; then
$ cat << EOF > ${SCRATCH}/policy.json
{
"Version": "2012-10-17",
```

```
"Statement": [
 {
   "Effect": "Allow",
   "Action": [
     "s3:CreateBucket",
     "s3:DeleteBucket",
     "s3:PutBucketTagging",
     "s3:GetBucketTagging",
     "s3:PutEncryptionConfiguration",
     "s3:GetEncryptionConfiguration",
     "s3:PutLifecycleConfiguration",
     "s3:GetLifecycleConfiguration",
     "s3:GetBucketLocation",
     "s3:ListBucket",
     "s3:GetObject",
     "s3:PutObject",
     "s3:DeleteObject",
     "s3:ListBucketMultipartUploads",
     "s3:AbortMultipartUpload",
     "s3:ListMultipartUploadParts",
     "ec2:DescribeSnapshots",
     "ec2:DescribeVolumes",
     "ec2:DescribeVolumeAttribute",
     "ec2:DescribeVolumesModifications",
     "ec2:DescribeVolumeStatus",
     "ec2:CreateTags",
     "ec2:CreateVolume",
     "ec2:CreateSnapshot",
     "ec2:DeleteSnapshot"
   ],
   "Resource": "*"
 }
]}
EOF
$ POLICY_ARN=$(aws iam create-policy --policy-name "RosaOadpVer1" \
--policy-document file:///${SCRATCH}/policy.json --query Policy.Arn \
--tags Key=rosa_openshift_version,Value=${CLUSTER_VERSION}
Key=rosa_role_prefix,Value=ManagedOpenShift
Key=operator_namespace,Value=openshift-oadp Key=operator_name,Value=openshift-oadp
\
--output text)
fi
$ echo ${POLICY_ARN}
```

2. クラスターの IAM ロール信頼ポリシーを作成します。

```
$ cat <<EOF > ${SCRATCH}/trust-policy.json
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "Federated": "arn:aws:iam::${AWS_ACCOUNT_ID}:oidc-provider/${OIDC_ENDPOINT}"
    },
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {
```

```
      "StringEquals": {
        "${OIDC_ENDPOINT}:sub": [
          "system:serviceaccount:openshift-adp:openshift-adp-controller-manager",
          "system:serviceaccount:openshift-adp:velero"]
      }
    }
  }]
}
EOF
$ ROLE_ARN=$(aws iam create-role --role-name \
 "${ROLE_NAME}" \
  --assume-role-policy-document file://${SCRATCH}/trust-policy.json \
  --tags Key=rosa_cluster_id,Value=${ROSA_CLUSTER_ID}
Key=rosa_openshift_version,Value=${CLUSTER_VERSION}
Key=rosa_role_prefix,Value=ManagedOpenShift
Key=operator_namespace,Value=openshift-adp Key=operator_name,Value=openshift-oadp \
  --query Role.Arn --output text)

$ echo ${ROLE_ARN}
```

3. IAM ポリシーを IAM ロールに割り当てます。

```
$ aws iam attach-role-policy --role-name "${ROLE_NAME}" \
 --policy-arn ${POLICY_ARN}
```

7.2. クラスターへの OADP のデプロイ

1. OADP の namespace を作成します。

```
$ oc create namespace openshift-adp
```

2. 認証情報のシークレットを作成します。

```
$ cat <<EOF > ${SCRATCH}/credentials
[default]
role_arn = ${ROLE_ARN}
web_identity_token_file = /var/run/secrets/openshift/serviceaccount/token
region=<aws_region> (1)
EOF
$ oc -n openshift-adp create secret generic cloud-credentials \
 --from-file=${SCRATCH}/credentials
```

(1) **<aws_region>** は、Security Token Service (STS) エンドポイントに使用する AWS リージョンに置き換えます。

3. OADP Operator をデプロイします。

注記

現在、Operator のバージョン 1.1 では、バックアップが **PartiallyFailed** ステータスになるという問題があります。これはバックアップと復元のプロセスには影響しないと思われますが、それに関連する問題があるため注意が必要です。

```
$ cat << EOF | oc create -f -
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
 generateName: openshift-adp-
 namespace: openshift-adp
 name: oadp
spec:
 targetNamespaces:
 - openshift-adp
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
 name: redhat-oadp-operator
 namespace: openshift-adp
spec:
 channel: stable-1.2
 installPlanApproval: Automatic
 name: redhat-oadp-operator
 source: redhat-operators
 sourceNamespace: openshift-marketplace
EOF
```

4. Operator の準備が整うまで待ちます。

```
$ watch oc -n openshift-adp get pods
```

出力例

```
NAME                                                READY   STATUS    RESTARTS   AGE
openshift-adp-controller-manager-546684844f-qqjhn   1/1     Running   0          22s
```

5. クラウドストレージを作成します。

```
$ cat << EOF | oc create -f -
apiVersion: oadp.openshift.io/v1alpha1
kind: CloudStorage
metadata:
 name: ${CLUSTER_NAME}-oadp
 namespace: openshift-adp
spec:
 creationSecret:
   key: credentials
   name: cloud-credentials
 enableSharedConfig: true
 name: ${CLUSTER_NAME}-oadp
 provider: aws
 region: $REGION
EOF
```

6. アプリケーションのストレージのデフォルトのストレージクラスを確認します。

```
$ oc get pvc -n <namespace> 1
```

```
-
```

1 アプリケーションの namespace を入力します。

出力例

```
NAME     STATUS   VOLUME                                     CAPACITY   ACCESS MODES
STORAGECLASS   AGE
applog   Bound    pvc-351791ae-b6ab-4e8b-88a4-30f73caf5ef8   1Gi        RWO
gp3-csi        4d19h
mysql    Bound    pvc-16b8e009-a20a-4379-accc-bc81fedd0621   1Gi        RWO
gp3-csi        4d19h
```

```
$ oc get storageclass
```

出力例

```
NAME              PROVISIONER             RECLAIMPOLICY   VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION   AGE
gp2               kubernetes.io/aws-ebs   Delete          WaitForFirstConsumer   true
4d21h
gp2-csi           ebs.csi.aws.com         Delete          WaitForFirstConsumer   true
4d21h
gp3               ebs.csi.aws.com         Delete          WaitForFirstConsumer   true
4d21h
gp3-csi (default)   ebs.csi.aws.com         Delete          WaitForFirstConsumer   true
4d21h
```

gp3-csi、gp2-csi、gp3、または gp2 のいずれかを使用すると機能します。バックアップ対象のアプリケーションがすべて CSI を使用した PV を使用している場合は、OADP の DPA 設定に CSI プラグインを含めます。

7. CSI のみ: Data Protection Application をデプロイします。

```
$ cat << EOF | oc create -f -
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
 name: ${CLUSTER_NAME}-dpa
 namespace: openshift-adp
spec:
 backupImages: true
 features:
   dataMover:
     enable: false
 backupLocations:
 - bucket:
     cloudStorageRef:
       name: ${CLUSTER_NAME}-oadp
     credential:
       key: credentials
       name: cloud-credentials
     prefix: velero
     default: true
     config:
```

```
      region: ${REGION}
 configuration:
   velero:
     defaultPlugins:
     - openshift
     - aws
     - csi
   restic:
     enable: false
EOF
```

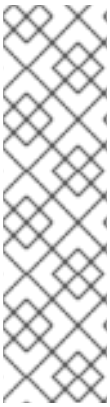
注記

CSI ボリュームに対してこのコマンドを実行する場合は、次のステップをスキップできます。

8. 非 CSI ボリューム: Data Protection Application をデプロイします。

```
$ cat << EOF | oc create -f -
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
 name: ${CLUSTER_NAME}-dpa
 namespace: openshift-adp
spec:
 backupImages: true
 features:
   dataMover:
     enable: false
 backupLocations:
 - bucket:
     cloudStorageRef:
       name: ${CLUSTER_NAME}-oadp
     credential:
       key: credentials
       name: cloud-credentials
     prefix: velero
     default: true
     config:
       region: ${REGION}
 configuration:
   velero:
     defaultPlugins:
     - openshift
     - aws
   restic:
     enable: false
 snapshotLocations:
   - velero:
       config:
         credentialsFile: /tmp/credentials/openshift-adp/cloud-credentials-credentials
         enableSharedConfig: 'true'
         profile: default
```

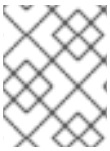
```
        region: ${REGION}
      provider: aws
EOF
```



注記

- OADP 1.1.x の Red Hat OpenShift Service on AWS STS 環境では、コンテナーイメージのバックアップと復元 (**spec.backupImages**) の値はサポートされていないため、**false** に設定する必要があります。
- Restic 機能 (**restic.enable=false**) は、Red Hat OpenShift Service on AWS STS 環境では無効になっており、サポートされていません。
- DataMover 機能 (**dataMover.enable=false**) は、Red Hat OpenShift Service on AWS STS 環境では無効になっており、サポートされていません。

7.3. バックアップの実行



注記

次のサンプル hello-world アプリケーションには、永続ボリュームが接続されていません。どちらの DPA 設定も機能します。

1. バックアップするワークロードを作成します。

```
$ oc create namespace hello-world
$ oc new-app -n hello-world --image=docker.io/openshift/hello-openshift
```

2. ルートを公開します。

```
$ oc expose service/hello-openshift -n hello-world
```

3. アプリケーションが動作していることを確認します。

```
$ curl `oc get route/hello-openshift -n hello-world -o jsonpath='{.spec.host}'`
```

出力例

```
Hello OpenShift!
```

4. ワークロードをバックアップします。

```
$ cat << EOF | oc create -f -
apiVersion: velero.io/v1
kind: Backup
metadata:
 name: hello-world
 namespace: openshift-adp
spec:
 includedNamespaces:
 - hello-world
```

```
 storageLocation: ${CLUSTER_NAME}-dpa-1
 ttl: 720h0m0s
EOF
```

5. バックアップが完了するまで待ちます。

```
$ watch "oc -n openshift-adp get backup hello-world -o json | jq .status"
```

出力例

```
{
 "completionTimestamp": "2022-09-07T22:20:44Z",
 "expiration": "2022-10-07T22:20:22Z",
 "formatVersion": "1.1.0",
 "phase": "Completed",
 "progress": {
   "itemsBackedUp": 58,
   "totalItems": 58
 },
 "startTimestamp": "2022-09-07T22:20:22Z",
 "version": 1
}
```

6. デモワークロードを削除します。

```
$ oc delete ns hello-world
```

7. バックアップから復元します。

```
$ cat << EOF | oc create -f -
apiVersion: velero.io/v1
kind: Restore
metadata:
 name: hello-world
 namespace: openshift-adp
spec:
 backupName: hello-world
EOF
```

8. 復元が完了するまで待ちます。

```
$ watch "oc -n openshift-adp get restore hello-world -o json | jq .status"
```

出力例

```
{
 "completionTimestamp": "2022-09-07T22:25:47Z",
 "phase": "Completed",
 "progress": {
   "itemsRestored": 38,
   "totalItems": 38
 },
```

```
 "startTimestamp": "2022-09-07T22:25:28Z",
 "warnings": 9
}
```


9. ワークロードが復元されていることを確認します。

```
$ oc -n hello-world get pods
```

出力例

```
NAME                          READY   STATUS    RESTARTS   AGE
hello-openshift-9f885f7c6-kdjpj   1/1     Running   0          90s
```

```
$ curl `oc get route/hello-openshift -n hello-world -o jsonpath='{.spec.host}'`
```

出力例

```
Hello OpenShift!
```

10. トラブルシューティングのヒントは、OADP チームの トラブルシューティングドキュメント を参照してください。

11. 追加のサンプルアプリケーションは、OADP チームの サンプルアプリケーションディレクトリー にあります。

7.4. クリーンアップ

1. ワークロードを削除します。

```
$ oc delete ns hello-world
```

2. バックアップおよび復元リソースが不要になった場合は、クラスターからリソースを削除します。

```
$ oc delete backups.velero.io hello-world
$ oc delete restores.velero.io hello-world
```

3. s3 のバックアップ/復元オブジェクトとリモートオブジェクトを削除するには、以下を実行します。

```
$ velero backup delete hello-world
$ velero restore delete hello-world
```

4. Data Protection Application を削除します。

```
$ oc -n openshift-adp delete dpa ${CLUSTER_NAME}-dpa
```

5. クラウドストレージを削除します。

```
$ oc -n openshift-adp delete cloudstorage ${CLUSTER_NAME}-oadp
```

警告

このコマンドがハングした場合は、ファイナライザーの削除が必要になる場合があります。

```
$ oc -n openshift-adp patch cloudstorage ${CLUSTER_NAME}-oadp -p
'{"metadata":{"finalizers":null}}' --type=merge
```

6. Operator が不要になった場合は、Operator を削除します。

```
$ oc -n openshift-adp delete subscription oadp-operator
```

7. Operator の namespace を削除します。

```
$ oc delete ns redhat-openshift-adp
```

8. カスタムリソース定義が不要になった場合は、クラスターからカスタムリソース定義を削除します。

```
$ for CRD in `oc get crds | grep velero | awk '{print $1}'`; do oc delete crd $CRD; done
$ for CRD in `oc get crds | grep -i oadp | awk '{print $1}'`; do oc delete crd $CRD; done
```

9. AWS S3 バケットを削除します。

```
$ aws s3 rm s3://${CLUSTER_NAME}-oadp --recursive
$ aws s3api delete-bucket --bucket ${CLUSTER_NAME}-oadp
```

10. ロールからポリシーの割り当てを解除します。

```
$ aws iam detach-role-policy --role-name "${ROLE_NAME}" \
 --policy-arn "${POLICY_ARN}"
```

11. ロールを削除します。

```
$ aws iam delete-role --role-name "${ROLE_NAME}"
```

第8章 チュートリアル: RED HAT OPENSHIFT SERVICE ON AWS 上の AWS LOAD BALANCER OPERATOR

重要

このコンテンツは Red Hat のエキスパートが作成したものですが、サポート対象のすべての設定でまだテストされていません。

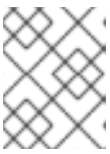
ヒント

AWS Load Balancer Operator によって作成されたロードバランサーは、OpenShift ルート には使用できません。OpenShift ルートのレイヤー 7 機能をすべて必要としない個々のサービスまたは Ingress リソースにのみ使用する必要があります。

AWS Load Balancer Controller は、Red Hat OpenShift Service on AWS クラスターの AWS Elastic Load Balancer を管理します。このコントローラーは、Kubernetes Ingress リソースを作成するときに AWS Application Load Balancer (ALB) をプロビジョニングし、LoadBalancer タイプを使用して Kubernetes Service リソースを実装するときに AWS Network Load Balancer (NLB) をプロビジョニングします。

デフォルトの AWS インツリーロードバランサープロバイダーと比較して、このコントローラーは ALB と NLB 用の詳細なアノテーションを使用して開発されています。高度な使用例としては以下が挙げられます。

- ネイティブ Kubernetes Ingress オブジェクトと ALB を使用する
- AWS ウェブアプリケーションファイアウォール (WAF) サービスと ALB を統合する



注記

WAFv1 (WAF クラシック) はサポートされなくなりました。WAFv2 を使用してください。

- カスタムの NLB ソース IP 範囲を指定する
- カスタムの NLB 内部 IP アドレスを指定する

AWS Load Balancer Operator は、Red Hat OpenShift Service on AWS クラスター内の **aws-load-balancer-controller** のインスタンスをインストール、管理、設定するために使用します。

8.1. 前提条件

注記

AWS ALB には、マルチ AZ クラスターと、クラスターと同じ VPC 内の 3 つの AZ に分散して配置された 3 つのパブリックサブネットが必要です。このため、ALB は多くの PrivateLink クラスターには適していません。AWS NLB にはこの制限はありません。

- マルチ AZ の Red Hat OpenShift Service on AWS クラスター
- BYO VPC クラスター

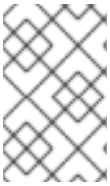
- AWS CLI
- OC CLI

8.1.1. 環境

- 環境変数を準備します。

```
$ export AWS_PAGER=""
$ export ROSA_CLUSTER_NAME=$(oc get infrastructure cluster -o=jsonpath="
{.status.infrastructureName}"  | sed 's/-[a-z0-9]\{5\}$//')
$ export REGION=$(oc get infrastructure cluster -o=jsonpath="
{.status.platformStatus.aws.region}")
$ export OIDC_ENDPOINT=$(oc get authentication.config.openshift.io cluster -o
jsonpath='{.spec.serviceAccountIssuer}' | sed  's|^https://||')
$ export AWS_ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
$ export SCRATCH="/tmp/${ROSA_CLUSTER_NAME}/alb-operator"
$ mkdir -p ${SCRATCH}
$ echo "Cluster: ${ROSA_CLUSTER_NAME}, Region: ${REGION}, OIDC Endpoint:
${OIDC_ENDPOINT}, AWS Account ID: ${AWS_ACCOUNT_ID}"
```

8.1.2. AWS VPC とサブネット



注記

このセクションは、既存の VPC にデプロイされたクラスターにのみ適用されます。クラスターを既存の VPC にデプロイしなかった場合は、このセクションをスキップして、その後のインストールセクションに進んでください。

1. 以下の変数を、実際のクラスターのデプロイメントに合わせて適切な値に設定します。

```
$ export VPC_ID=<vpc-id>
$ export PUBLIC_SUBNET_IDS=<public-subnets>
$ export PRIVATE_SUBNET_IDS=<private-subnets>
$ export CLUSTER_NAME=$(oc get infrastructure cluster -o=jsonpath="
{.status.infrastructureName}")
```

2. クラスター名を使用してクラスターの VPC にタグを追加します。

```
$ aws ec2 create-tags --resources ${VPC_ID} --tags
Key=kubernetes.io/cluster/${CLUSTER_NAME},Value=owned --region ${REGION}
```

3. パブリックサブネットにタグを追加します。

```
$ aws ec2 create-tags \
    --resources ${PUBLIC_SUBNET_IDS} \
    --tags Key=kubernetes.io/role/elb,Value='' \
    --region ${REGION}
```

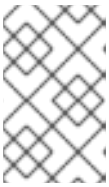
4. プライベートサブネットにタグを追加します。

```
$ aws ec2 create-tags \
    --resources "${PRIVATE_SUBNET_IDS}" \
```

```
    --tags Key=kubernetes.io/role/internal-elb,Value='' \
    --region ${REGION}
```

8.2. インストール

1. AWS Load Balancer Controller の AWS IAM ポリシーを作成します。



注記

このポリシーは、アップストリームの AWS Load Balancer Controller ポリシーに加えて、サブネット上にタグを作成する権限から取得されます。これは Operator が機能するために必要です。

```
$ oc new-project aws-load-balancer-operator
$ POLICY_ARN=$(aws iam list-policies --query \
    "Policies[?PolicyName=='aws-load-balancer-operator-policy'].{ARN:Arn}" \
    --output text)
$ if [[ -z "${POLICY_ARN}" ]]; then
   wget -O "${SCRATCH}/load-balancer-operator-policy.json" \
     https://raw.githubusercontent.com/rh-mobb/documentation/main/content/rosa/aws-load-balancer-operator/load-balancer-operator-policy.json
    POLICY_ARN=$(aws --region "$REGION" --query Policy.Arn \
    --output text iam create-policy \
    --policy-name aws-load-balancer-operator-policy \
    --policy-document "file://${SCRATCH}/load-balancer-operator-policy.json")
fi
$ echo $POLICY_ARN
```

2. AWS Load Balancer Operator の AWS IAM 信頼ポリシーを作成します。

```
$ cat <<EOF > "${SCRATCH}/trust-policy.json"
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Condition": {
   "StringEquals" : {
     "${OIDC_ENDPOINT}:sub": ["system:serviceaccount:aws-load-balancer-operator:aws-load-balancer-operator-controller-manager", "system:serviceaccount:aws-load-balancer-operator:aws-load-balancer-controller-cluster"]
   }
 },
 "Principal": {
   "Federated": "arn:aws:iam::$AWS_ACCOUNT_ID:oidc-provider/${OIDC_ENDPOINT}"
 },
 "Action": "sts:AssumeRoleWithWebIdentity"
 }
 ]
}
EOF
```

3. AWS Load Balancer Operator の AWS IAM ロールを作成します。

```
$ ROLE_ARN=$(aws iam create-role --role-name "${ROSA_CLUSTER_NAME}-alb-operator"
\
   --assume-role-policy-document "file://${SCRATCH}/trust-policy.json" \
   --query Role.Arn --output text)
$ echo $ROLE_ARN

$ aws iam attach-role-policy --role-name "${ROSA_CLUSTER_NAME}-alb-operator" \
    --policy-arn $POLICY_ARN
```

4. 新しく作成した AWS IAM ロールを引き受けるための AWS Load Balancer Operator 用のシークレットを作成します。

```
$ cat << EOF | oc apply -f -
apiVersion: v1
kind: Secret
metadata:
  name: aws-load-balancer-operator
  namespace: aws-load-balancer-operator
stringData:
  credentials: |
    [default]
    role_arn = $ROLE_ARN
    web_identity_token_file = /var/run/secrets/openshift/serviceaccount/token
EOF
```

5. AWS Load Balancer Operator をインストールします。

```
$ cat << EOF | oc apply -f -
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: aws-load-balancer-operator
  namespace: aws-load-balancer-operator
spec:
  upgradeStrategy: Default
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: aws-load-balancer-operator
  namespace: aws-load-balancer-operator
spec:
  channel: stable-v1.0
  installPlanApproval: Automatic
  name: aws-load-balancer-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  startingCSV: aws-load-balancer-operator.v1.0.0
EOF
```

6. 次の Operator を使用して、AWS Load Balancer Controller のインスタンスをデプロイします。

注記

ここでエラーが発生した場合は、少し待ってから再試行してください。エラーが発生するのは、Operator がまだインストールを完了していないためです。

```
$ cat << EOF | oc apply -f -
apiVersion: networking.olm.openshift.io/v1
kind: AWSLoadBalancerController
metadata:
  name: cluster
spec:
  credentials:
    name: aws-load-balancer-operator
EOF
```

7. Operator Pod とコントローラー Pod の両方が実行されていることを確認します。

```
$ oc -n aws-load-balancer-operator get pods
```

次のようなメッセージが表示されます。表示されない場合は、少し待ってから再試行してください。

```
NAME                                                READY   STATUS    RESTARTS   AGE
aws-load-balancer-controller-cluster-6ddf658785-pdp5d          1/1     Running   0          99s
aws-load-balancer-operator-controller-manager-577d9ffcb9-w6zqn   2/2     Running   0
2m4s
```

8.3. デプロイメントの検証

1. プロジェクトを新規作成します。

```
$ oc new-project hello-world
```

2. Hello World アプリケーションをデプロイします。

```
$ oc new-app -n hello-world --image=docker.io/openshift/hello-openshift
```

3. AWS ALB が接続する NodePort サービスを設定します。

```
$ cat << EOF | oc apply -f -
apiVersion: v1
kind: Service
metadata:
  name: hello-openshift-nodeport
  namespace: hello-world
spec:
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
  type: NodePort
```

```
  selector:
    deployment: hello-openshift
EOF
```

4. AWS Load Balancer Operator を使用して AWS ALB をデプロイします。

```
$ cat << EOF | oc apply -f -
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: hello-openshift-alb
  namespace: hello-world
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
spec:
  ingressClassName: alb
  rules:
    - http:
        paths:
          - path: /
            pathType: Exact
            backend:
              service:
                name: hello-openshift-nodeport
                port:
                  number: 80
EOF
```

5. AWS ALB Ingress エンドポイントを curl して、Hello World アプリケーションにアクセスできることを確認します。

注記

AWS ALB のプロビジョニングには数分かかります。**curl: (6) Could not resolve host** というエラーが表示された場合は、待機してから再試行してください。

```
$ INGRESS=$(oc -n hello-world get ingress hello-openshift-alb \
    -o jsonpath='{.status.loadBalancer.ingress[0].hostname}')
$ curl "http://${INGRESS}"
```

出力例

```
Hello OpenShift!
```

6. Hello World アプリケーション用に AWS NLB をデプロイします。

```
$ cat << EOF | oc apply -f -
apiVersion: v1
kind: Service
metadata:
  name: hello-openshift-nlb
  namespace: hello-world
  annotations:
```

```
    service.beta.kubernetes.io/aws-load-balancer-type: external
    service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: instance
    service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
spec:
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
  type: LoadBalancer
  selector:
    deployment: hello-openshift
EOF
```

7. AWS NLB エンドポイントをテストします。

注記

NLB のプロビジョニングには数分かかります。**curl: (6) Could not resolve host** というエラーが表示された場合は、待機してから再試行してください。

```
$ NLB=$(oc -n hello-world get service hello-openshift-nlb \
  -o jsonpath='{.status.loadBalancer.ingress[0].hostname}')
$ curl "http://${NLB}"
```

出力例

```
Hello OpenShift!
```

8.4. クリーンアップ

1. hello world アプリケーションの namespace (および namespace 内のすべてのリソース) を削除します。

```
$ oc delete project hello-world
```

2. AWS Load Balancer Operator と AWS IAM ロールを削除します。

```
$ oc delete subscription aws-load-balancer-operator -n aws-load-balancer-operator
$ aws iam detach-role-policy \
  --role-name "${ROSA_CLUSTER_NAME}-alb-operator" \
  --policy-arn $POLICY_ARN
$ aws iam delete-role \
  --role-name "${ROSA_CLUSTER_NAME}-alb-operator"
```

3. AWS IAM ポリシーを削除します。

```
$ aws iam delete-policy --policy-arn $POLICY_ARN
```

第9章 チュートリアル: MICROSOFT ENTRA ID (旧称 AZURE ACTIVE DIRECTORY) をアイデンティティープロバイダーとして設定する

Microsoft Entra ID (旧称 Azure Active Directory) を Red Hat OpenShift Service on AWS のクラスターアイデンティティープロバイダーとして設定できます。

このチュートリアルでは、次のタスクを完了する手順を示します。

1. 認証のために Entra ID に新しいアプリケーションを登録します。
2. Entra ID でのアプリケーション登録を設定して、トークンに任意のクレームとグループクレームを含めます。
3. Entra ID をアイデンティティープロバイダーとして使用するように Red Hat OpenShift Service on AWS クラスターを設定します。
4. 個々のグループに追加の権限を付与します。

9.1. 前提条件

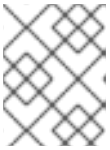
- Microsoft のドキュメント に従って、一連のセキュリティーグループを作成し、ユーザーを割り当てている。

9.2. 認証のために ENTRA ID に新規アプリケーションを登録する

Entra ID にアプリケーションを登録するには、まず OAuth コールバック URL を作成し、次にアプリケーションを登録します。

手順

1. 指定の変数を変更し、次のコマンドを実行して、クラスターの OAuth コールバック URL を作成します。



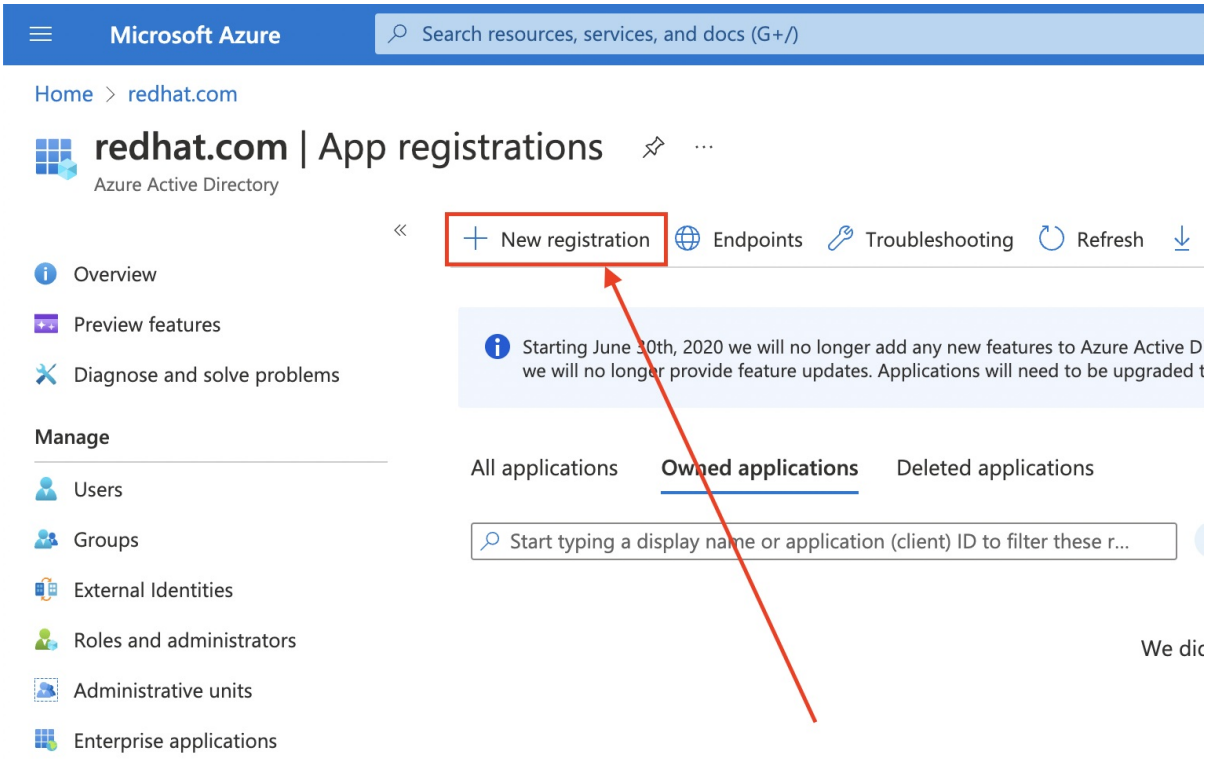
注記

このコールバック URL を忘れずに保存してください。後のプロセスで必要になります。

```
$ domain=$(rosa describe cluster -c <cluster_name> | grep "DNS" | grep -oE
'\S+.openshiftapps.com')
echo "OAuth callback URL: https://oauth.${domain}/oauth2callback/AAD"
```

OAuth コールバック URL の末尾にある "AAD" ディレクトリーは、このプロセスで後で設定する OAuth アイデンティティープロバイダー名と同じである必要があります。

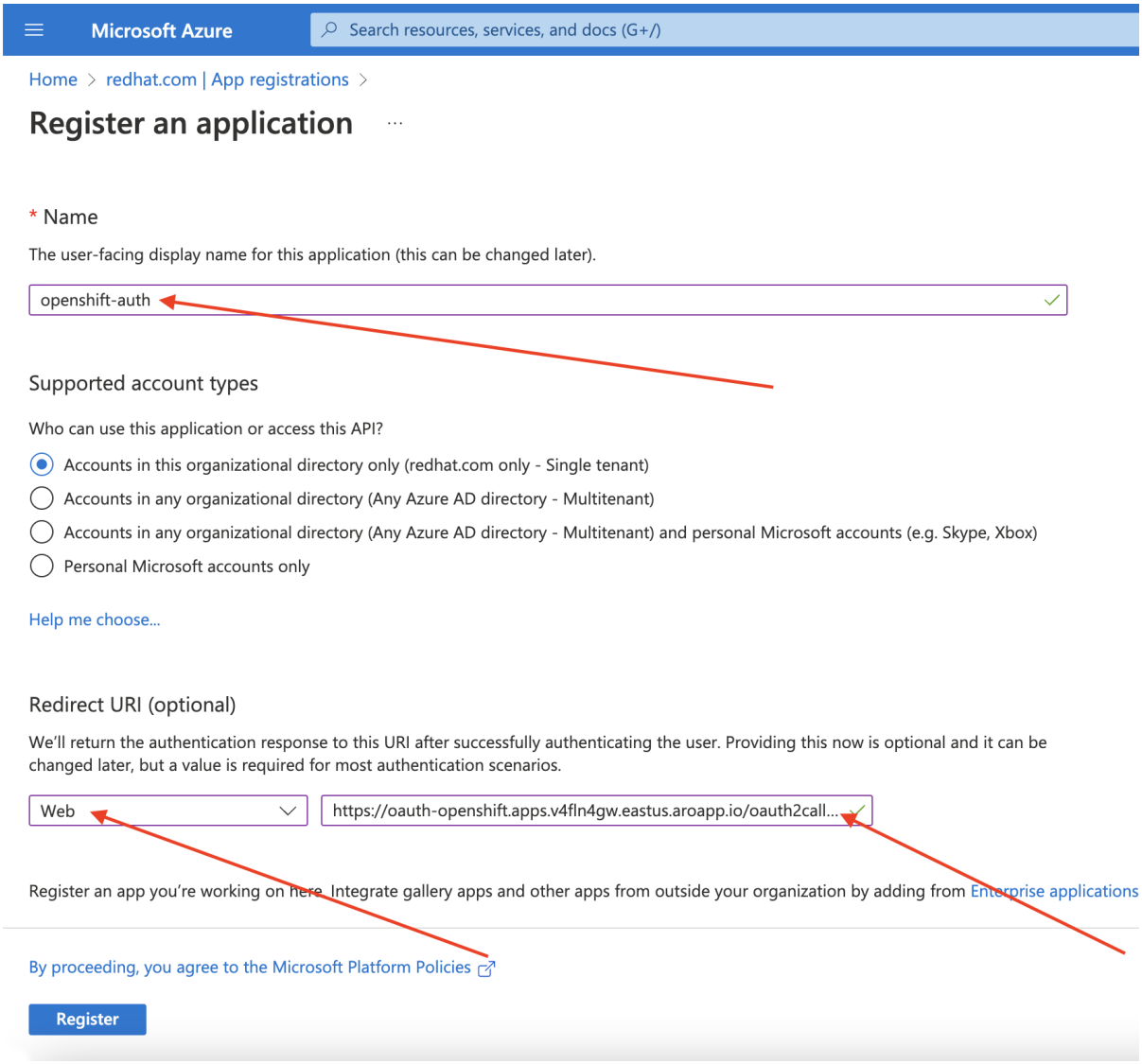
2. Azure portal にログインして Entra ID アプリケーションを作成し、App registrations ブレードを選択します。次に、**New registration** を選択して新しいアプリケーションを作成します。



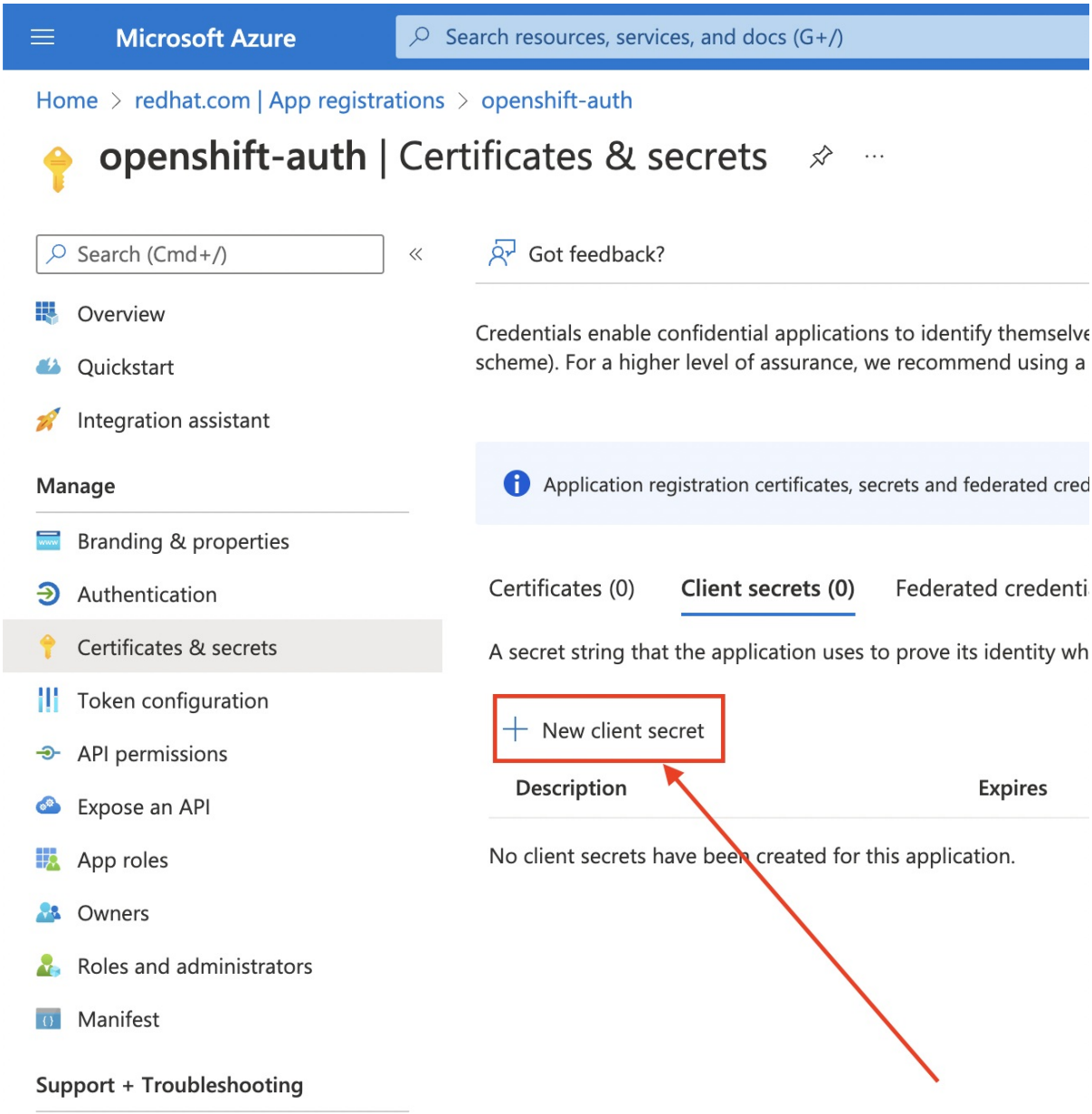
3. アプリケーションに名前を付けます (例: **openshift-auth**)。

4. **Redirect URI** ドロップダウンから **Web** を選択し、前のステップで取得した OAuth コールバック URL の値を入力します。

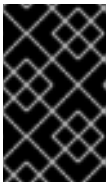
5. 必要な情報を入力したら、**Register** をクリックしてアプリケーションを作成します。



6. **Certificates & secrets** サブブレードを選択し、**New client secret** を選択します。

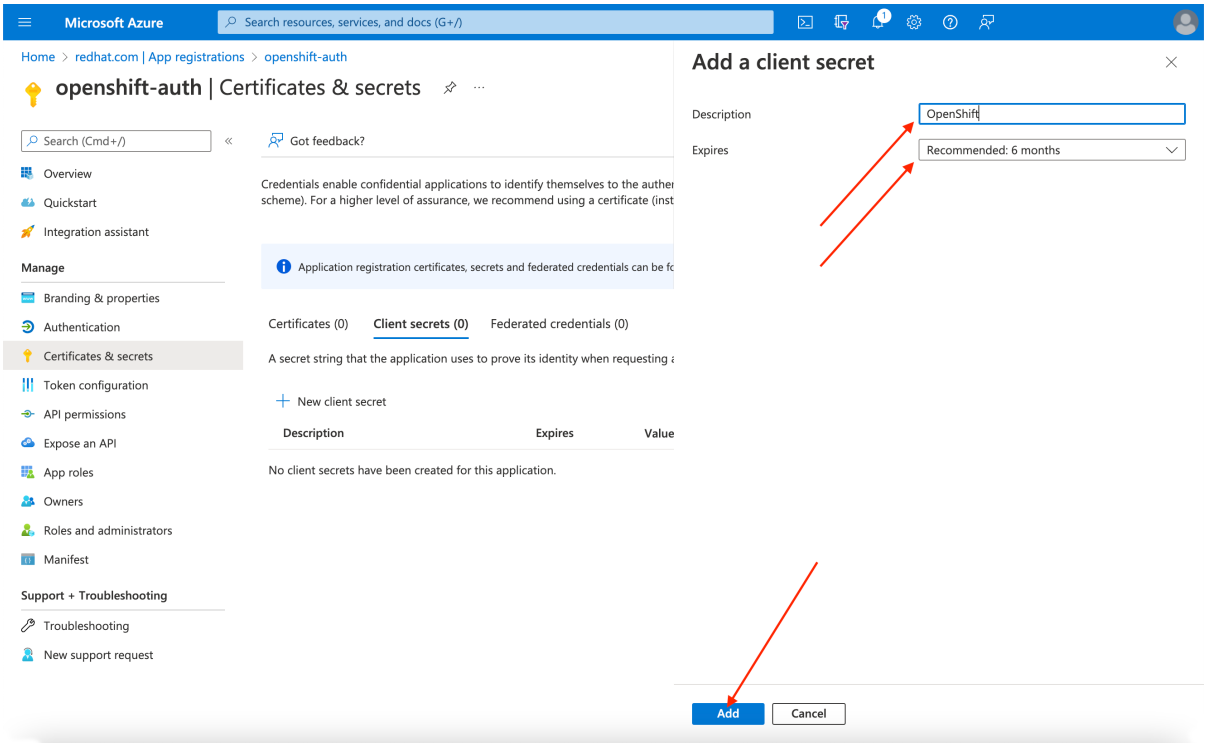


7. 要求された詳細を入力し、生成されたクライアントシークレット値を保存します。このシークレットは、このプロセスで後で必要になります。

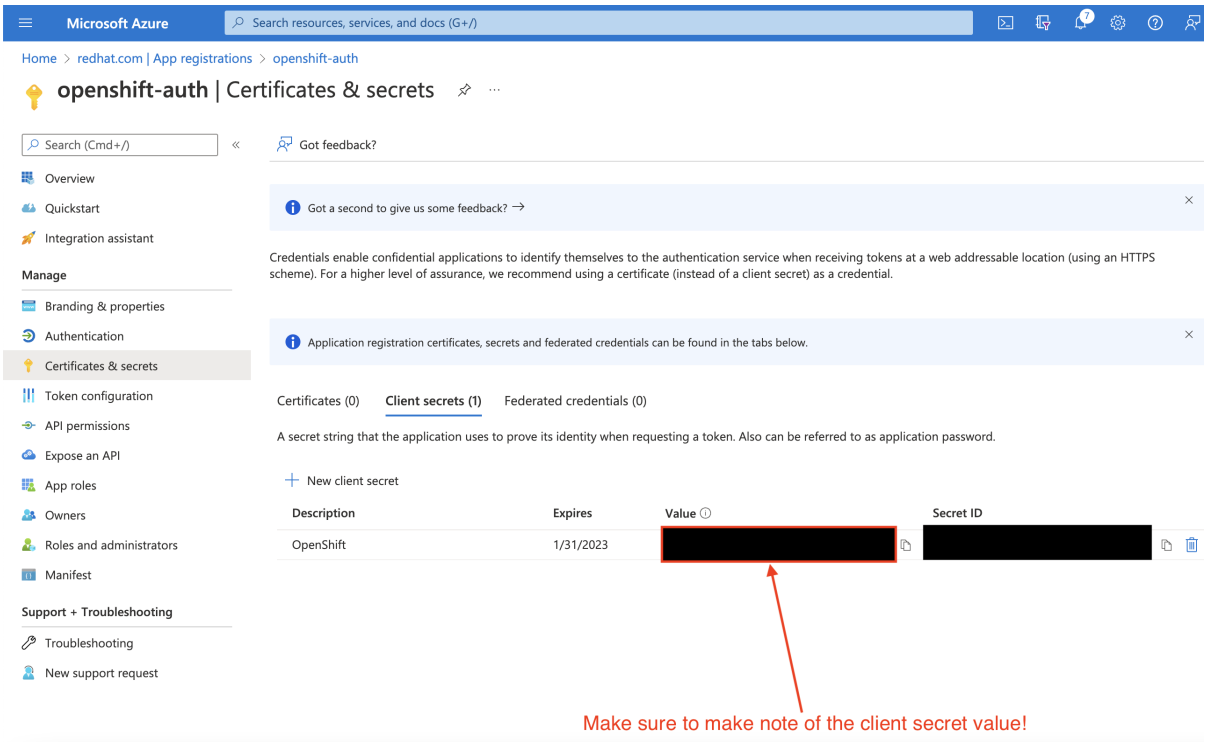


重要

初期セットアップ後は、クライアントシークレットを確認できません。クライアントシークレットを保存しなかった場合は、新しいクライアントシークレットを生成する必要があります。



8. **Overview** サブブレードを選択し、**Application (client) ID** と **Directory (tenant) ID** をメモします。これらの値は後のステップで必要になります。



9.3. 任意のクレームとグループクレームを含めるように ENTRA ID でのアプリケーション登録を設定する

Red Hat OpenShift Service on AWS がユーザーのアカウントを作成するのに十分な情報を取得できるように、Entra ID を設定して 2 つの任意のクレーム (**email** と **preferred_username**) を指定する必要があります。Entra ID の任意のクレームに関する詳細は、Microsoft のドキュメント を参照してください。

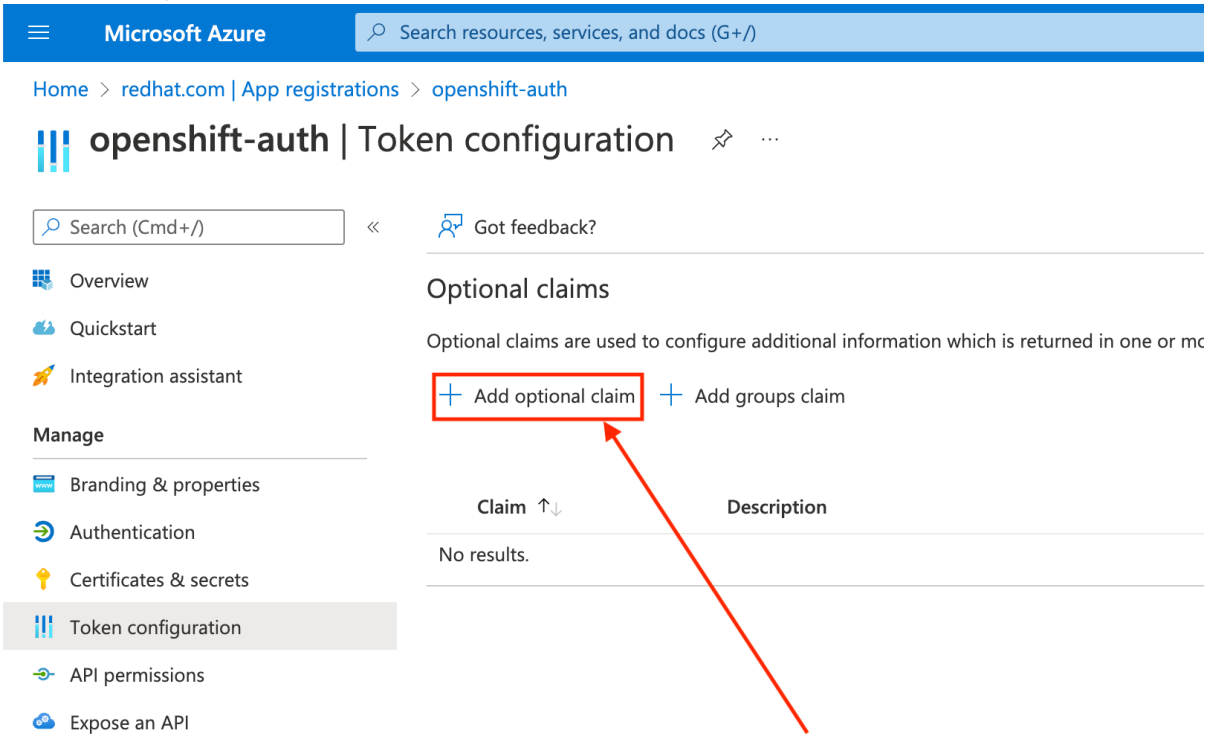
個々のユーザー認証に加えて、Red Hat OpenShift Service on AWS はグループクレーム機能を提供します。この機能により、Entra ID などの OpenID Connect (OIDC) アイデンティティープロバイダーが、Red Hat OpenShift Service on AWS 内で使用するユーザーのグループメンバーシップを提供できるよう

になります。

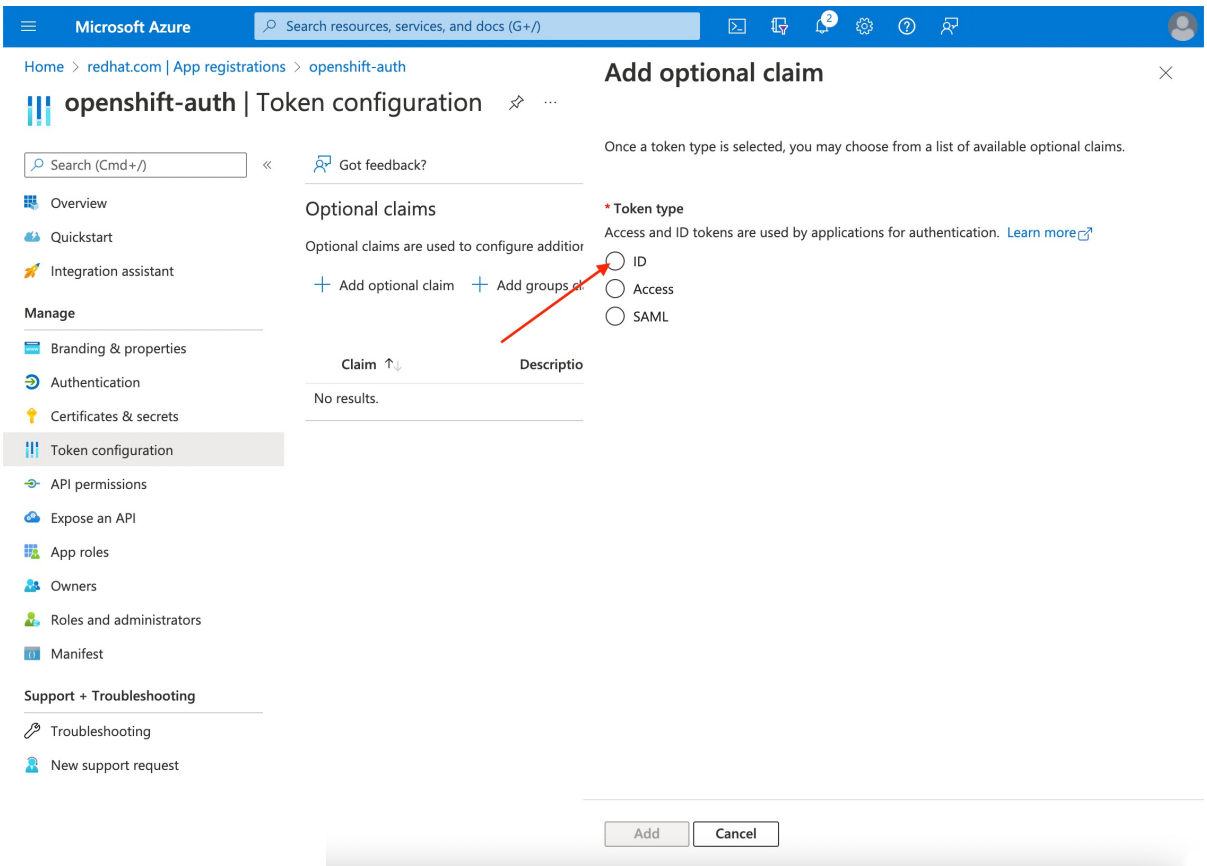
9.3.1. 任意のクレームの設定

Entra ID の任意のクレームを設定できます。

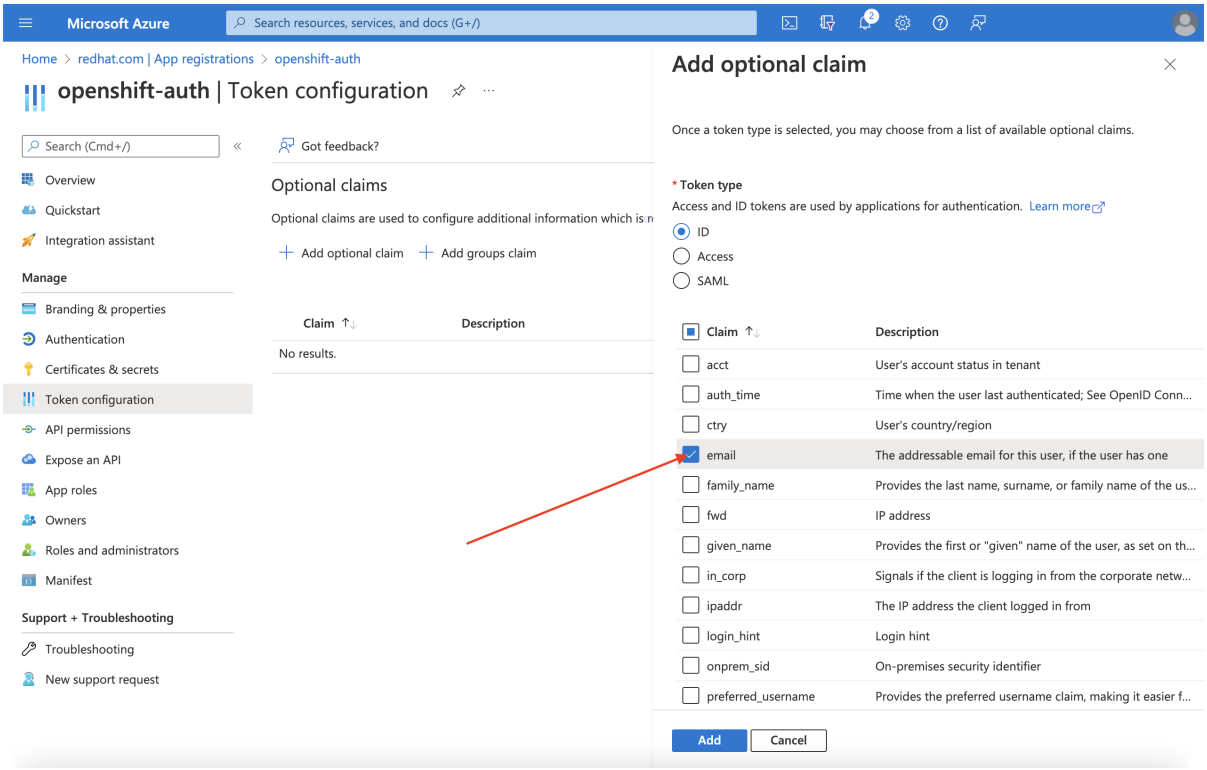
1. **Token configuration** サブブレードをクリックし、**Add optional claim** ボタンを選択します。



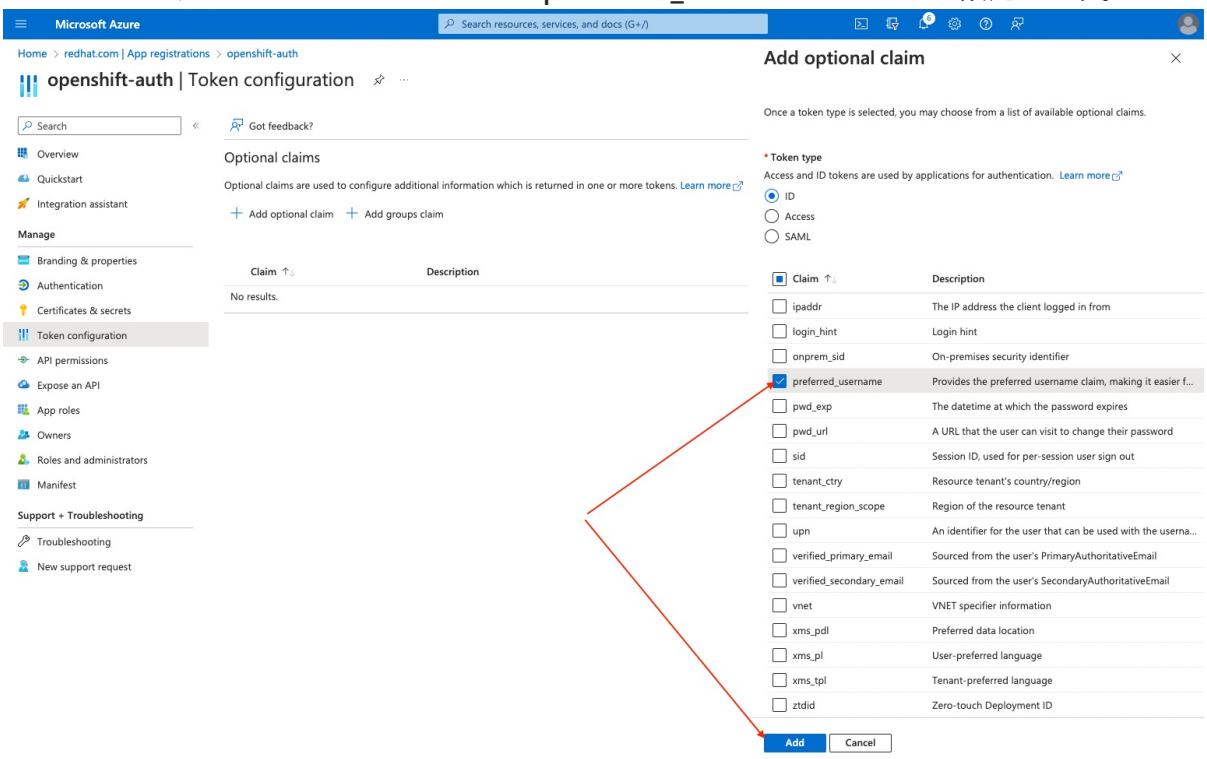
2. **ID** ラジオボタンを選択します。



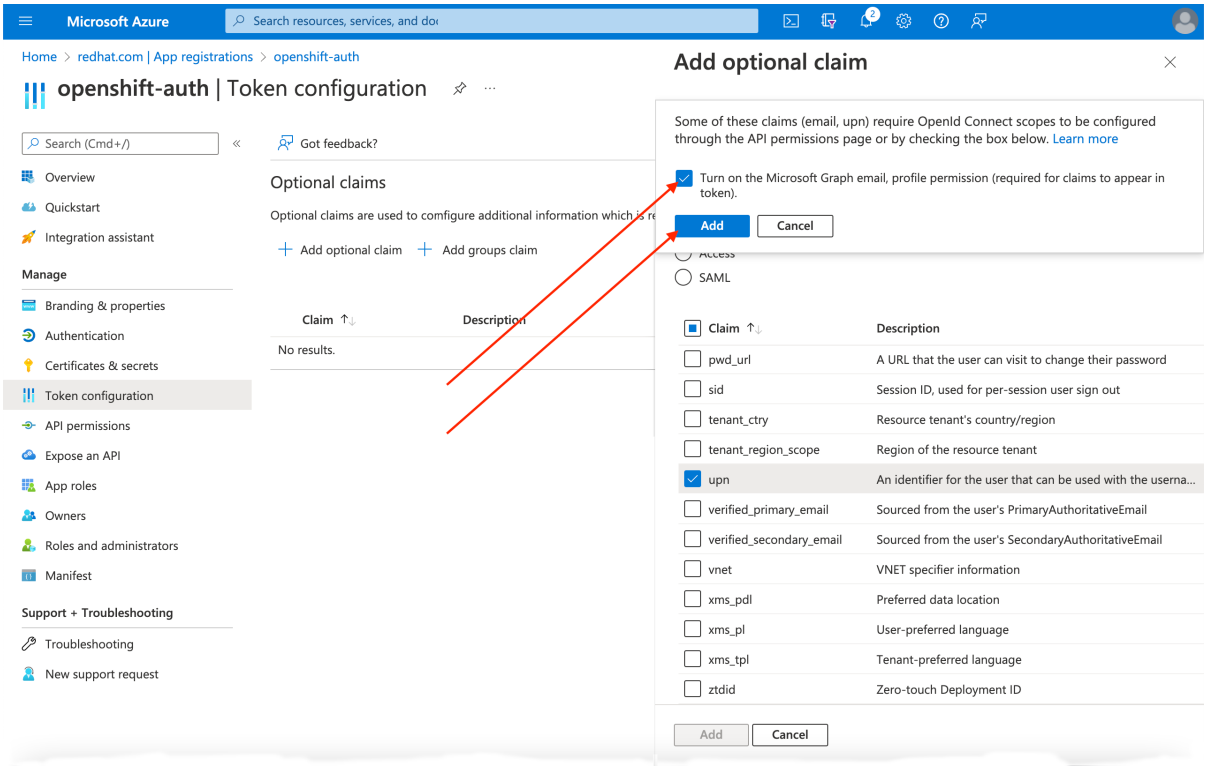
3. **email** クレームのチェックボックスを選択します。



4. **preferred_username** クレームのチェックボックスを選択します。次に、**Add** をクリックし、Entra ID アプリケーションの **email** および **preferred_username** クレームを設定します。



5. ページの上部にダイアログボックスが表示されます。プロンプトに従って、必要な Microsoft Graph 権限を有効にします。

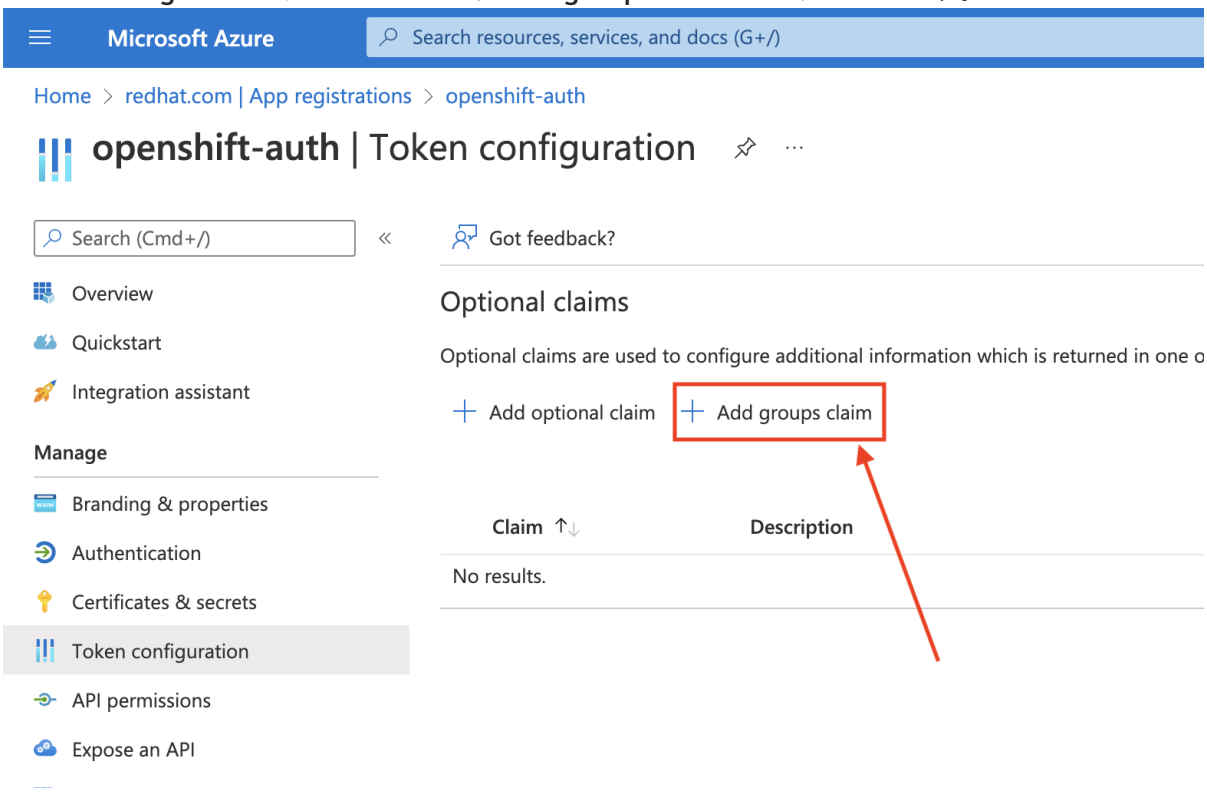


9.3.2. グループクレームの設定 (オプション)

グループクレームを提供するように Entra ID を設定します。

手順

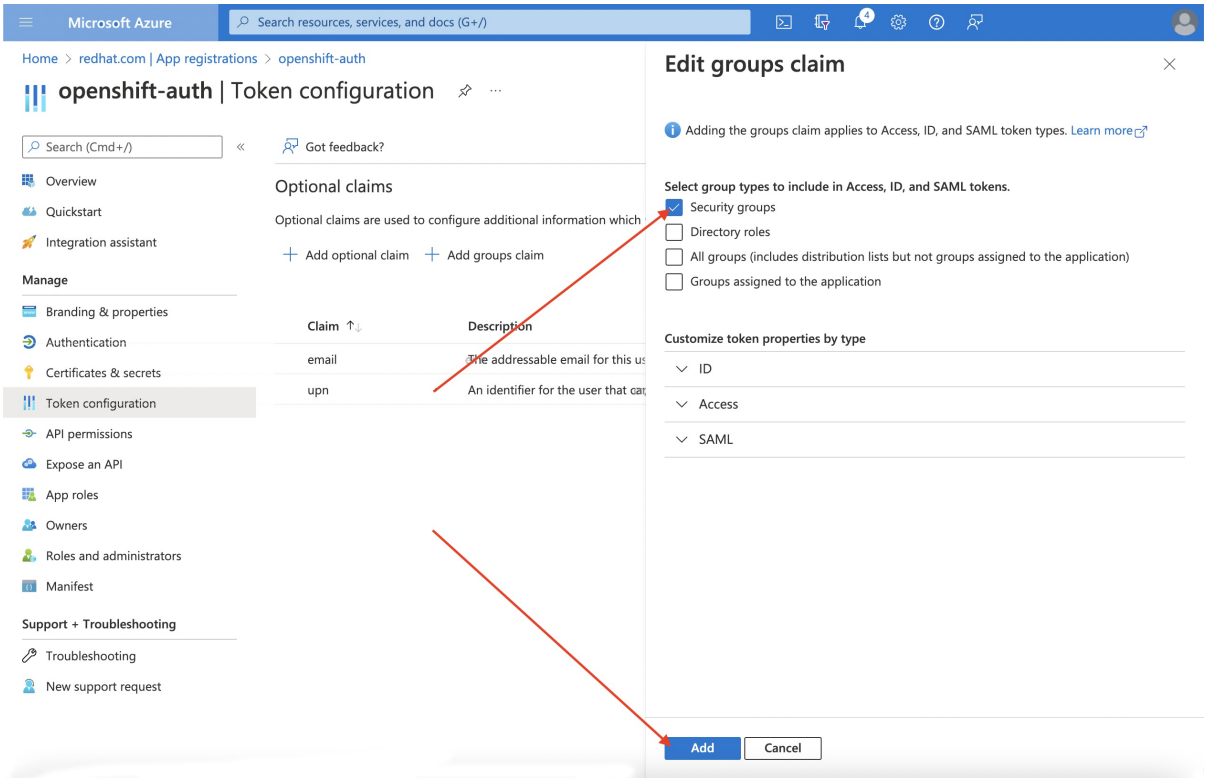
1. **Token configuration** サブブレードで、**Add groups claim** をクリックします。



2. Entra ID アプリケーションのグループクレームを設定するには、**Security groups** を選択し、**Add** をクリックします。

注記

この例では、グループクレームに、ユーザーがメンバーとなっているすべてのセキュリティーグループを含めます。実際の実稼働環境では、グループクレームに、Red Hat OpenShift Service on AWS に適用するグループのみを含めてください。



9.4. ENTRA ID をアイデンティティープロバイダーとして使用するように RED HAT OPENSHIFT SERVICE ON AWS クラスターを設定する

Entra ID をアイデンティティープロバイダーとして使用するように Red Hat OpenShift Service on AWS を設定する必要があります。

Red Hat OpenShift Service on AWS は OpenShift Cluster Manager を使用してアイデンティティープロバイダーを設定する機能を提供しますが、ここでは ROSA CLI を使用して、Entra ID をアイデンティティープロバイダーとして使用するようにクラスターの OAuth プロバイダーを設定します。アイデンティティープロバイダーを設定する前に、アイデンティティープロバイダー設定に必要な変数を設定します。

手順

1. 次のコマンドを実行して変数を作成します。

```
$ CLUSTER_NAME=example-cluster (1)
$ IDP_NAME=AAD (2)
$ APP_ID=yyyyyyyy-yyyy-yyyy-yyyy-yyyyyyyyyyyy (3)
$ CLIENT_SECRET=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx (4)
$ TENANT_ID=zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzzzzzz (5)
```

(1) これはクラスターの名前に置き換えます。

2 この値は、このプロセスで前に生成した OAuth コールバック URL で使用した名前に置き換えます。

3 これはアプリケーション (クライアント) ID に置き換えます。

4 これはクライアントシークレットに置き換えます。

5 これはディレクトリー (テナント) ID に置き換えます。

2. 次のコマンドを実行して、クラスターの OAuth プロバイダーを設定します。グループクレームを有効にした場合は、必ず **--group-claims groups** 引数を使用してください。

- グループクレームを有効にした場合は、次のコマンドを実行します。

```
$ rosa create idp \
--cluster ${CLUSTER_NAME} \
--type openid \
--name ${IDP_NAME} \
--client-id ${APP_ID} \
--client-secret ${CLIENT_SECRET} \
--issuer-url https://login.microsoftonline.com/${TENANT_ID}/v2.0 \
--email-claims email \
--name-claims name \
--username-claims preferred_username \
--extra-scopes email,profile \
--groups-claims groups
```

- グループクレームを有効にしなかった場合は、次のコマンドを実行します。

```
$ rosa create idp \
--cluster ${CLUSTER_NAME} \
--type openid \
--name ${IDP_NAME} \
--client-id ${APP_ID} \
--client-secret ${CLIENT_SECRET} \
--issuer-url https://login.microsoftonline.com/${TENANT_ID}/v2.0 \
--email-claims email \
--name-claims name \
--username-claims preferred_username \
--extra-scopes email,profile
```

数分後、クラスター認証 Operator が変更を調整します。すると、Entra ID を使用してクラスターにログインできるようになります。

9.5. 個々のユーザーおよびグループへの追加の権限の付与

初めてログインすると、権限が非常に制限されていることがわかります。デフォルトでは、Red Hat OpenShift Service on AWS はクラスター内に新しいプロジェクトまたは namespace を作成する権限のみを付与します。他のプロジェクトの表示は制限されています。

このような追加の権限を個々のユーザーおよびグループに付与する必要があります。

9.5.1. 個々のユーザーに追加の権限を付与する

Red Hat OpenShift Service on AWS には、クラスターに対する完全なアクセスと制御を付与する **cluster-admin** ロールなど、事前設定済みの多数のロールが組み込まれています。

手順

- 次のコマンドを実行して、ユーザーに **cluster-admin** ロールへのアクセス権を付与します。

```
$ rosa grant user cluster-admin \
  --user=<USERNAME> 1
  --cluster=${CLUSTER_NAME}
```

1 cluster-admin 権限を付与する Entra ID ユーザー名を指定します。

9.5.2. 個々のグループに追加の権限を付与する

グループクレームを有効にすることを選択した場合、クラスター OAuth プロバイダーによって、ユーザーのグループメンバーシップがグループ ID を使用して自動的に作成または更新されます。クラスター OAuth プロバイダーは、作成されたグループの **RoleBindings** と **ClusterRoleBindings** を自動的に作成しません。これらのバインディングは、ユーザーが独自のプロセスを使用して作成する必要があります。

自動生成されたグループに **cluster-admin** ロールへのアクセス権を付与するには、グループ ID への **ClusterRoleBinding** を作成する必要があります。

手順

- 次のコマンドを実行して、**ClusterRoleBinding** を作成します。

```
$ oc create clusterrolebinding cluster-admin-group \
--clusterrole=cluster-admin \
--group=<GROUP_ID> 1
```

1 cluster-admin 権限を付与する Entra ID グループ ID を指定します。

これで、指定したグループ内のすべてのユーザーに **cluster-admin** アクセス権が自動的に付与されます。

9.6. 関連情報

RBAC を使用して Red Hat OpenShift Service on AWS の権限を定義および適用する方法の詳細は、Red Hat OpenShift Service on AWS のドキュメント を参照してください。

第10章 チュートリアル: STS を使用する RED HAT OPENSHIFT SERVICE ON AWS での AWS SECRETS MANAGER CSI の使用

AWS Secrets and Configuration Provider (ASCP) は、AWS シークレットを Kubernetes ストレージボリュームとして公開する方法を提供します。ASCP を使用すると、Secrets Manager のシークレットを保存および管理し、Red Hat OpenShift Service on AWS で実行されているワークロードを通じてシークレットを取得できます。

10.1. 前提条件

このプロセスを開始する前に、次のリソースとツールがあることを確認してください。

- STS を使用してデプロイした Red Hat OpenShift Service on AWS クラスター
- Helm 3
- **aws** CLI
- **oc** CLI
- **jq** CLI

10.1.1. その他の環境要件

1. 次のコマンドを実行して、Red Hat OpenShift Service on AWS クラスターにログインします。

```
$ oc login --token=<your-token> --server=<your-server-url>
```

ログイントークンを見つけるには、Red Hat OpenShift Cluster Manager からプルシークレットでクラスターにアクセスします。

2. 次のコマンドを実行して、クラスターに STS があることを検証します。

```
$ oc get authentication.config.openshift.io cluster -o json \
  | jq .spec.serviceAccountIssuer
```

出力例

```
"https://xxxxx.cloudfront.net/xxxxx"
```

出力が異なる場合は、続行しないでください。このプロセスを続行する前に、STS クラスターの作成に関する Red Hat ドキュメント を参照してください。

3. 次のコマンドを実行して、CSI ドライバーの実行を許可するように **SecurityContextConstraints** 権限を設定します。

```
$ oc new-project csi-secrets-store
$ oc adm policy add-scc-to-user privileged \
    system:serviceaccount:csi-secrets-store:secrets-store-csi-driver
$ oc adm policy add-scc-to-user privileged \
    system:serviceaccount:csi-secrets-store:csi-secrets-store-provider-aws
```

4. 次のコマンドを実行して、このプロセスで後で使用する環境変数を作成します。

```
$ export REGION=$(oc get infrastructure cluster -o=jsonpath="
{.status.platformStatus.aws.region}")
$ export OIDC_ENDPOINT=$(oc get authentication.config.openshift.io cluster \
  -o jsonpath='{.spec.serviceAccountIssuer}' | sed  's|^https://||')
$ export AWS_ACCOUNT_ID=`aws sts get-caller-identity --query Account --output text`
$ export AWS_PAGER=""
```

10.2. AWS シークレットと設定プロバイダーのデプロイ

1. 次のコマンドを実行し、Helm を使用してシークレットストア CSI ドライバーを登録します。

```
$ helm repo add secrets-store-csi-driver \
    https://kubernetes-sigs.github.io/secrets-store-csi-driver/charts
```

2. 次のコマンドを実行して、Helm リポジトリーを更新します。

```
$ helm repo update
```

3. 次のコマンドを実行して、シークレットストア CSI ドライバーをインストールします。

```
$ helm upgrade --install -n csi-secrets-store \
    csi-secrets-store-driver secrets-store-csi-driver/secrets-store-csi-driver
```

4. 次のコマンドを実行して、AWS プロバイダーをデプロイします。

```
$ oc -n csi-secrets-store apply -f \
    https://raw.githubusercontent.com/rh-mobb/documentation/main/content/misc/secrets-
store-csi/aws-provider-installer.yaml
```

5. 次のコマンドを実行して、両方の Daemonset が実行されていることを確認します。

```
$ oc -n csi-secrets-store get ds \
    csi-secrets-store-provider-aws \
    csi-secrets-store-driver-secrets-store-csi-driver
```

6. 次のコマンドを実行して、シークレットストア CSI ドライバーにラベルを付けて、制限付き Pod セキュリティープロファイルとともに使用することを許可します。

```
$ oc label csidriver.storage.k8s.io/secrets-store.csi.k8s.io security.openshift.io/csi-ephemeral-
volume-profile=restricted
```

10.3. シークレットと IAM アクセスポリシーの作成

1. 次のコマンドを実行して、Secrets Manager のシークレットを作成します。

```
$ SECRET_ARN=$(aws --region "$REGION" secretsmanager create-secret \
    --name MySecret --secret-string \
    '{"username":"shadowman", "password":"hunter2"}' \
    --query ARN --output text); echo $SECRET_ARN
```

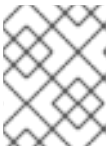
2. 次のコマンドを実行して、IAM アクセスポリシードキュメントを作成します。

```
$ cat << EOF > policy.json
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret"
      ],
      "Resource": ["$SECRET_ARN"]
      }]
}
EOF
```

3. 次のコマンドを実行して、IAM アクセスポリシーを作成します。

```
$ POLICY_ARN=$(aws --region "$REGION" --query Policy.Arn \
--output text iam create-policy \
--policy-name openshift-access-to-mysecret-policy \
--policy-document file://policy.json); echo $POLICY_ARN
```

4. 次のコマンドを実行して、IAM ロール信頼ポリシードキュメントを作成します。



注記

信頼ポリシーは、このプロセスで後で作成する namespace のデフォルトのサービスアカウントにロックされます。

```
$ cat <<EOF > trust-policy.json
{
   "Version": "2012-10-17",
   "Statement": [
   {
   "Effect": "Allow",
   "Condition": {
     "StringEquals" : {
       "${OIDC_ENDPOINT}:sub": ["system:serviceaccount:my-application:default"]
     }
    },
    "Principal": {
       "Federated": "arn:aws:iam::$AWS_ACCOUNT_ID:oidc-provider/${OIDC_ENDPOINT}"
    },
    "Action": "sts:AssumeRoleWithWebIdentity"
    }
    ]
}
EOF
```

5. 次のコマンドを実行して、IAM ロールを作成します。

```
$ ROLE_ARN=$(aws iam create-role --role-name openshift-access-to-mysecret \
--assume-role-policy-document file://trust-policy.json \
--query Role.Arn --output text); echo $ROLE_ARN
```

6. 次のコマンドを実行して、ロールをポリシーに割り当てます。

```
$ aws iam attach-role-policy --role-name openshift-access-to-mysecret \
    --policy-arn $POLICY_ARN
```

10.4. このシークレットを使用するアプリケーションの作成

1. 次のコマンドを実行して、OpenShift プロジェクトを作成します。

```
$ oc new-project my-application
```

2. 次のコマンドを実行して、STS ロールを使用するようにデフォルトのサービスアカウントにアノテーションを付けます。

```
$ oc annotate -n my-application serviceaccount default \
    eks.amazonaws.com/role-arn=$ROLE_ARN
```

3. 次のコマンドを実行して、シークレットにアクセスするためのシークレットプロバイダークラスを作成します。

```
$ cat << EOF | oc apply -f -
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: my-application-aws-secrets
spec:
  provider: aws
  parameters:
    objects: |
      - objectName: "MySecret"
        objectType: "secretsmanager"
EOF
```

4. 次のコマンドでシークレットを使用してデプロイメントを作成します。

```
$ cat << EOF | oc apply -f -
apiVersion: v1
kind: Pod
metadata:
  name: my-application
  labels:
    app: my-application
spec:
  volumes:
  - name: secrets-store-inline
    csi:
      driver: secrets-store.csi.k8s.io
      readOnly: true
      volumeAttributes:
        secretProviderClass: "my-application-aws-secrets"
  containers:
  - name: my-application-deployment
    image: k8s.gcr.io/e2e-test-images/busybox:1.29
    command:
```

```
      - "/bin/sleep"
      - "10000"
    volumeMounts:
    - name: secrets-store-inline
      mountPath: "/mnt/secrets-store"
      readOnly: true
EOF
```

5. 次のコマンドを実行して、Pod にシークレットがマウントされていることを確認します。

```
$ oc exec -it my-application -- cat /mnt/secrets-store/MySecret
```

10.5. クリーンアップ

1. 次のコマンドを実行してアプリケーションを削除します。

```
$ oc delete project my-application
```

2. 次のコマンドを実行して、シークレットストア CSI ドライバーを削除します。

```
$ helm delete -n csi-secrets-store csi-secrets-store-driver
```

3. 次のコマンドを実行して、Security Context Constraints を削除します。

```
$ oc adm policy remove-scc-from-user privileged \
    system:serviceaccount:csi-secrets-store:secrets-store-csi-driver; oc adm policy remove-scc-from-user privileged \
    system:serviceaccount:csi-secrets-store:csi-secrets-store-provider-aws
```

4. 次のコマンドを実行して、AWS プロバイダーを削除します。

```
$ oc -n csi-secrets-store delete -f \
https://raw.githubusercontent.com/rh-mobb/documentation/main/content/misc/secrets-store-csi/aws-provider-installer.yaml
```

5. 次のコマンドを実行して、AWS のロールとポリシーを削除します。

```
$ aws iam detach-role-policy --role-name openshift-access-to-mysecret \
    --policy-arn $POLICY_ARN; aws iam delete-role --role-name openshift-access-to-mysecret; aws iam delete-policy --policy-arn $POLICY_ARN
```

6. 次のコマンドを実行して、Secrets Manager のシークレットを削除します。

```
$ aws secretsmanager --region $REGION delete-secret --secret-id $SECRET_ARN
```

第11章 チュートリアル: RED HAT OPENSHIFT SERVICE ON AWS での AWS CONTROLLERS FOR KUBERNETES の使用

AWS Controllers for Kubernetes (ACK) を使用すると、AWS サービスリソースを Red Hat OpenShift Service on AWS から直接定義して使用できます。ACK を使用すると、クラスター外のリソースを定義したり、クラスター内のデータベースやメッセージキューなどのサポート機能を提供するサービスを実行したりすることなく、アプリケーションで AWS マネージドサービスを利用できます。

さまざまな ACK Operator をソフトウェアカタログから直接インストールできます。これにより、アプリケーションで Operator を簡単に使い始めることができます。このコントローラーは AWS Controller for Kubernetes プロジェクトのコンポーネントであり、現在開発者プレビュー段階にあります。

このチュートリアルを使用して、ACK S3 Operator をデプロイしてください。クラスターのソフトウェアカタログ内の他の ACK Operator に合わせて調整することもできます。

11.1. 前提条件

- Red Hat OpenShift Service on AWS クラスター
- **cluster-admin** 権限を持つユーザーアカウント
- OpenShift CLI (**oc**)
- Amazon Web Services (AWS) CLI (**aws**)

11.2. 環境の設定

1. 次の環境変数を設定し、お使いのクラスターに合わせてクラスター名を変更します。

```
$ export CLUSTER_NAME=$(oc get infrastructure cluster -o=jsonpath="
{.status.infrastructureName}"  | sed 's/-[a-z0-9]\{5\}$//')
$ export REGION=$(rosa describe cluster -c ${ROSA_CLUSTER_NAME} --output json | jq -r
.region.id)
$ export OIDC_ENDPOINT=$(oc get authentication.config.openshift.io cluster -o json | jq -r
.spec.serviceAccountIssuer | sed  's|^https://||')
$ export AWS_ACCOUNT_ID=`aws sts get-caller-identity --query Account --output text`
$ export ACK_SERVICE=s3
$ export ACK_SERVICE_ACCOUNT=ack-${ACK_SERVICE}-controller
$ export POLICY_ARN=arn:aws:iam::aws:policy/AmazonS3FullAccess
$ export AWS_PAGER=""
$ export SCRATCH="/tmp/${ROSA_CLUSTER_NAME}/ack"
$ mkdir -p ${SCRATCH}
```

2. 次のセクションに進む前に、すべてのフィールドが正しく出力されていることを確認してください。

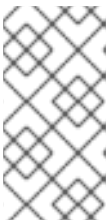
```
$ echo "Cluster: ${ROSA_CLUSTER_NAME}, Region: ${REGION}, OIDC Endpoint:
${OIDC_ENDPOINT}, AWS Account ID: ${AWS_ACCOUNT_ID}"
```

11.3. AWS アカウントの準備

1. ACK Operator の AWS Identity Access Management (IAM) 信頼ポリシーを作成します。

```
$ cat <<EOF > "${SCRATCH}/trust-policy.json"
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Condition": {
   "StringEquals" : {
     "${OIDC_ENDPOINT}:sub": "system:serviceaccount:ack-
system:${ACK_SERVICE_ACCOUNT}"
   }
 },
 "Principal": {
   "Federated": "arn:aws:iam::$AWS_ACCOUNT_ID:oidc-provider/${OIDC_ENDPOINT}"
 },
 "Action": "sts:AssumeRoleWithWebIdentity"
 }
 ]
}
EOF
```

2. **AmazonS3FullAccess** ポリシーを割り当てた、ACK Operator が引き受ける AWS IAM ロールを作成します。



注記

推奨されるポリシーは、各プロジェクトの GitHub リポジトリー (例: https://github.com/aws-controllers-k8s/s3-controller/blob/main/config/iam/recommended-policy-arn) で見つけることができます。

```
$ ROLE_ARN=$(aws iam create-role --role-name "ack-${ACK_SERVICE}-controller" \
   --assume-role-policy-document "file://${SCRATCH}/trust-policy.json" \
   --query Role.Arn --output text)
$ echo $ROLE_ARN

$ aws iam attach-role-policy --role-name "ack-${ACK_SERVICE}-controller" \
    --policy-arn ${POLICY_ARN}
```

11.4. ACK S3 コントローラーのインストール

1. ACK S3 Operator をインストールするプロジェクトを作成します。

```
$ oc new-project ack-system
```

2. ACK S3 Operator の設定を含むファイルを作成します。

注記

ACK_WATCH_NAMESPACE は、コントローラーがクラスター内のすべての namespace を適切に監視できるように、意図的に空白のままにします。

```
$ cat << EOF  "${SCRATCH}/config.txt"
ACK_ENABLE_DEVELOPMENT_LOGGING=true
ACK_LOG_LEVEL=debug
ACK_WATCH_NAMESPACE=
AWS_REGION=${REGION}
AWS_ENDPOINT_URL=
ACK_RESOURCE_TAGS=${CLUSTER_NAME}
ENABLE_LEADER_ELECTION=true
LEADER_ELECTION_NAMESPACE=
RECONCILE_DEFAULT_MAX_CONCURRENT_SYNCS=1
FEATURE_FLAGS=
FEATURE_GATES=
EOF
```

3. 前のステップのファイルを使用して ConfigMap を作成します。

```
$ oc -n ack-system create configmap \
  --from-env-file=${SCRATCH}/config.txt ack-${ACK_SERVICE}-user-config
```

4. ソフトウェアカタログから ACK S3 Operator をインストールします。

```
$ cat << EOF | oc apply -f -
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: ack-${ACK_SERVICE}-controller
  namespace: ack-system
spec:
  upgradeStrategy: Default
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ack-${ACK_SERVICE}-controller
  namespace: ack-system
spec:
  channel: alpha
  installPlanApproval: Automatic
  name: ack-${ACK_SERVICE}-controller
  source: community-operators
  sourceNamespace: openshift-marketplace
EOF
```

5. ACK S3 Operator サービスアカウントに、割り当てる AWS IAM ロールのアノテーションを付けて、デプロイメントを再起動します。

```
$ oc -n ack-system annotate serviceaccount ${ACK_SERVICE_ACCOUNT} \
  eks.amazonaws.com/role-arn=${ROLE_ARN} && \
  oc -n ack-system rollout restart deployment ack-${ACK_SERVICE}-controller
```

6. ACK S3 Operator が実行されていることを確認します。

```
$ oc -n ack-system get pods
```

出力例

```
NAME                                READY   STATUS    RESTARTS   AGE
ack-s3-controller-585f6775db-s4lfz  1/1     Running   0          51s
```

11.5. デプロイメントの検証

1. S3 バケットリソースをデプロイします。

```
$ cat << EOF | oc apply -f -
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: ${CLUSTER-NAME}-bucket
  namespace: ack-system
spec:
  name: ${CLUSTER-NAME}-bucket
EOF
```

2. S3 バケットが AWS で作成されたことを確認します。

```
$ aws s3 ls | grep ${CLUSTER_NAME}-bucket
```

出力例

```
2023-10-04 14:51:45 mrmc-test-maz-bucket
```

11.6. クリーンアップ

1. S3 バケットリソースを削除します。

```
$ oc -n ack-system delete bucket.s3.services.k8s.aws/${CLUSTER-NAME}-bucket
```

2. ACK S3 Operator と AWS IAM ロールを削除します。

```
$ oc -n ack-system delete subscription ack-${ACK_SERVICE}-controller
$ aws iam detach-role-policy \
  --role-name "ack-${ACK_SERVICE}-controller" \
  --policy-arn ${POLICY_ARN}
$ aws iam delete-role \
  --role-name "ack-${ACK_SERVICE}-controller"
```

3. **ack-system** プロジェクトを削除します。

```
$ oc delete project ack-system
```

第12章 チュートリアル: 外部トラフィックへの一貫した EGRESS IP の割り当て

セキュリティー基準を満たすために IP ベースの設定を必要とするセキュリティーグループなど、クラスターから出るトラフィックに一貫した IP アドレスを割り当てることができます。

デフォルトでは、Red Hat OpenShift Service on AWS は、OVN-Kubernetes の Container Network Interface (CNI) を使用して、プールからランダムな IP アドレスを割り当てます。これにより、セキュリティーロックダウンの設定が予測不可能になったり、オープンになったりする可能性があります。

詳細は、Egress IP アドレスの設定 を参照してください。

目的

- Egress クラスタートラフィック用に予測可能な IP アドレスのセットを設定する方法を学習します。

前提条件

- OVN-Kubernetes を使用してデプロイした Red Hat OpenShift Service on AWS クラスター
- OpenShift CLI (**oc**)
- ROSA CLI (**rosa**)
- **jq**

12.1. 環境変数の設定

- 次のコマンドを実行して、環境変数を設定します。

注記

別のマシンプールをターゲットにするには、**ROSA_MACHINE_POOL_NAME** 変数の値を置き換えます。

```
$ export ROSA_CLUSTER_NAME=$(oc get infrastructure cluster -o=jsonpath="
{.status.infrastructureName}"  | sed 's/-[a-z0-9]\{5\}$//')
$ export ROSA_MACHINE_POOL_NAME=worker
```

12.2. 容量の確保

各ノードに割り当てられる IP アドレスの数は、パブリッククラウドプロバイダーごとに制限されます。

- 次のコマンドを実行して、十分な容量を確認します。

```
$ oc get node -o json | \
    jq '.items[] |
        {
            "name": .metadata.name,
            "ips": (.status.addresses | map(select(.type == "InternalIP") | .address)),
```

```
            "capacity": (.metadata.annotations."cloud.network.openshift.io/egress-ipconfig" |
fromjson[] | .capacity.ipv4)
        }'
```

出力例

```
---
{
  "name": "ip-10-10-145-88.ec2.internal",
  "ips": [
    "10.10.145.88"
  ],
  "capacity": 14
}
{
  "name": "ip-10-10-154-175.ec2.internal",
  "ips": [
    "10.10.154.175"
  ],
  "capacity": 14
}
---
```

12.3. EGRESS IP ルールの作成

1. Egress IP ルールを作成する前に、使用する Egress IP を特定します。

注記

選択する Egress IP は、ワーカーノードがプロビジョニングされているサブネットの一部として存在する必要があります。

2. **オプション**: AWS Virtual Private Cloud (VPC) Dynamic Host Configuration Protocol (DHCP) サービスとの競合を回避するために、要求した Egress IP を予約します。
CIDR 予約ページの AWS ドキュメント で明示的な IP 予約をリクエストします。

12.4. NAMESPACE への EGRESS IP の割り当て

1. 次のコマンドを実行して新しいプロジェクトを作成します。

```
$ oc new-project demo-egress-ns
```

2. 次のコマンドを実行して、namespace 内のすべての Pod に Egress ルールを作成します。

```
$ cat <<EOF | oc apply -f -
apiVersion: k8s.ovn.org/v1
kind: EgressIP
metadata:
  name: demo-egress-ns
spec:
  # NOTE: these egress IPs are within the subnet range(s) in which my worker nodes
  #       are deployed.
  egressIPs:
```

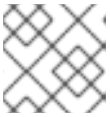
```
    - 10.10.100.253
    - 10.10.150.253
    - 10.10.200.253
  namespaceSelector:
    matchLabels:
      kubernetes.io/metadata.name: demo-egress-ns
EOF
```

12.5. POD への EGRESS IP の割り当て

1. 次のコマンドを実行して新しいプロジェクトを作成します。

```
$ oc new-project demo-egress-pod
```

2. 次のコマンドを実行して、Pod の Egress ルールを作成します。



注記

spec.namespaceSelector は必須フィールドです。

```
$ cat <<EOF | oc apply -f -
apiVersion: k8s.ovn.org/v1
kind: EgressIP
metadata:
  name: demo-egress-pod
spec:
  # NOTE: these egress IPs are within the subnet range(s) in which my worker nodes
  #       are deployed.
  egressIPs:
    - 10.10.100.254
    - 10.10.150.254
    - 10.10.200.254
  namespaceSelector:
    matchLabels:
      kubernetes.io/metadata.name: demo-egress-pod
  podSelector:
    matchLabels:
      run: demo-egress-pod
EOF
```

12.5.1. ノードのラベル付け

1. 次のコマンドを実行して、保留中の Egress IP 割り当てを取得します。

```
$ oc get egressips
```

出力例

```
NAME              EGRESSIPS       ASSIGNED NODE   ASSIGNED EGRESSIPS
demo-egress-ns    10.10.100.253
demo-egress-pod   10.10.100.254
```

作成した Egress IP ルールは、**k8s.ovn.org/egress-assignable** ラベルを持つノードにのみ適用されます。ラベルが特定のマシンプールにのみ存在することを確認します。

2. 次のコマンドを使用して、マシンプールにラベルを割り当てます。

警告

マシンプールのノードラベルに依存している場合は、このコマンドによってそれらのラベルが置き換えられます。ノードラベルを保持するには、必要なラベルを **--labels** フィールドに入力してください。

```
$ rosa update machinepool ${ROSA_MACHINE_POOL_NAME} \
  --cluster="${ROSA_CLUSTER_NAME}" \
  --labels "k8s.ovn.org/egress-assignable="
```

12.5.2. Egress IP の確認

- 次のコマンドを実行して、Egress IP の割り当てを確認します。

```
$ oc get egressips
```

出力例

```
NAME              EGRESSIPS       ASSIGNED NODE                 ASSIGNED EGRESSIPS
demo-egress-ns    10.10.100.253   ip-10-10-156-122.ec2.internal   10.10.150.253
demo-egress-pod   10.10.100.254   ip-10-10-156-122.ec2.internal   10.10.150.254
```

12.6. 検証

12.6.1. サンプルアプリケーションのデプロイ

Egress IP ルールをテストするには、指定した Egress IP アドレスに制限されるサービスを作成します。これは、IP アドレスの小さなサブセットを期待する外部サービスをシミュレートします。

1. リクエストを複製するには、**echoserver** コマンドを実行します。

```
$ oc -n default run demo-service --image=gcr.io/google_containers/echoserver:1.4
```

2. 次のコマンドを実行して、Pod をサービスとして公開し、指定した Egress IP アドレスへの Ingress を制限します。

```
$ cat <<EOF | oc apply -f -
apiVersion: v1
kind: Service
metadata:
  name: demo-service
  namespace: default
```

```
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-scheme: "internal"
    service.beta.kubernetes.io/aws-load-balancer-internal: "true"
spec:
  selector:
    run: demo-service
  ports:
    - port: 80
      targetPort: 8080
  type: LoadBalancer
  externalTrafficPolicy: Local
  # NOTE: this limits the source IPs that are allowed to connect to our service.  It
  #       is being used as part of this demo, restricting connectivity to our egress
  #       IP addresses only.
  # NOTE: these egress IPs are within the subnet range(s) in which my worker nodes
  #       are deployed.
  loadBalancerSourceRanges:
    - 10.10.100.254/32
    - 10.10.150.254/32
    - 10.10.200.254/32
    - 10.10.100.253/32
    - 10.10.150.253/32
    - 10.10.200.253/32
EOF
```

3. 次のコマンドを実行して、ロードバランサーのホスト名を取得し、環境変数として保存します。

```
$ export LOAD_BALANCER_HOSTNAME=$(oc get svc -n default demo-service -o json | jq -
r '.status.loadBalancer.ingress[].hostname')
```

12.6.2. namespace の Egress のテスト

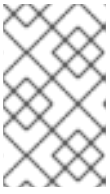
1. 対話型シェルを起動して、namespace の Egress ルールをテストします。

```
$ oc run \
  demo-egress-ns \
  -it \
  --namespace=demo-egress-ns \
  --env=LOAD_BALANCER_HOSTNAME=$LOAD_BALANCER_HOSTNAME \
  --image=registry.access.redhat.com/ubi9/ubi -- \
  bash
```

2. ロードバランサーにリクエストを送信し、正常に接続できることを確認します。

```
$ curl -s http://$LOAD_BALANCER_HOSTNAME
```

3. 接続が成功したかどうかの出力を確認します。



注記

client_address は、Egress IP ではなく、ロードバランサーの内部 IP アドレスです。**.spec.loadBalancerSourceRanges** に制限されたサービスに接続することで、クライアントアドレスが正しく設定されていることを確認できます。

出力例

```
CLIENT VALUES:
client_address=10.10.207.247
command=GET
real path=/
query=nil
request_version=1.1
request_uri=http://internal-a3e61de18bfca4a53a94a208752b7263-148284314.us-east-
1.elb.amazonaws.com:8080/

SERVER VALUES:
server_version=nginx: 1.10.0 - lua: 10001

HEADERS RECEIVED:
accept=*/*
host=internal-a3e61de18bfca4a53a94a208752b7263-148284314.us-east-
1.elb.amazonaws.com
user-agent=curl/7.76.1
BODY:
-no body in request-
```

4. 以下のコマンドを実行して Pod を終了します。

```
$ exit
```

12.6.3. Pod の Egress のテスト

1. 対話型シェルを起動して、Pod の Egress ルールをテストします。

```
$ oc run \
  demo-egress-pod \
  -it \
  --namespace=demo-egress-pod \
  --env=LOAD_BALANCER_HOSTNAME=$LOAD_BALANCER_HOSTNAME \
  --image=registry.access.redhat.com/ubi9/ubi -- \
  bash
```

2. 次のコマンドを実行して、ロードバランサーにリクエストを送信します。

```
$ curl -s http://$LOAD_BALANCER_HOSTNAME
```

3. 接続が成功したかどうかの出力を確認します。

注記

client_address は、Egress IP ではなく、ロードバランサーの内部 IP アドレスです。**.spec.loadBalancerSourceRanges** に制限されたサービスに接続することで、クライアントアドレスが正しく設定されていることを確認できます。

出力例

```
CLIENT VALUES:
```

```
client_address=10.10.207.247
command=GET
real path=/
query=nil
request_version=1.1
request_uri=http://internal-a3e61de18bfca4a53a94a208752b7263-148284314.us-east-
1.elb.amazonaws.com:8080/

SERVER VALUES:
server_version=nginx: 1.10.0 - lua: 10001

HEADERS RECEIVED:
accept=*/*
host=internal-a3e61de18bfca4a53a94a208752b7263-148284314.us-east-
1.elb.amazonaws.com
user-agent=curl/7.76.1
BODY:
-no body in request-
```

4. 以下のコマンドを実行して Pod を終了します。

```
$ exit
```

12.6.4. オプション: ブロックされた Egress のテスト

1. **オプション:** 次のコマンドを実行して、Egress ルールが適用されない場合にトラフィックが正常にブロックされることをテストします。

```
$ oc run \
  demo-egress-pod-fail \
  -it \
  --namespace=demo-egress-pod \
  --env=LOAD_BALANCER_HOSTNAME=$LOAD_BALANCER_HOSTNAME \
  --image=registry.access.redhat.com/ubi9/ubi -- \
  bash
```

2. 次のコマンドを実行して、ロードバランサーにリクエストを送信します。

```
$ curl -s http://$LOAD_BALANCER_HOSTNAME
```

3. コマンドが失敗すると、Egress が正常にブロックされます。

4. 以下のコマンドを実行して Pod を終了します。

```
$ exit
```

12.7. クラスターのクリーンアップ

1. 次のコマンドを実行してクラスターをクリーンアップします。

```
$ oc delete svc demo-service -n default; \
$ oc delete pod demo-service -n default; \
$ oc delete project demo-egress-ns; \
```

```
$ oc delete project demo-egress-pod; \
$ oc delete egressip demo-egress-ns; \
$ oc delete egressip demo-egress-pod
```

2. 次のコマンドを実行して、割り当てられたノードラベルをクリーンアップします。

警告

マシンプールのノードラベルに依存している場合は、このコマンドによってそれらのラベルが置き換えられます。ノードラベルを保持するには、必要なラベルを **--labels** フィールドに入力します。

```
$ rosa update machinepool ${ROSA_MACHINE_POOL_NAME} \
  --cluster="${ROSA_CLUSTER_NAME}" \
  --labels ""
```