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: 2024-03-26

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

法律上の通知

概要

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

目次

第1章 チュートリアルの概要

マネージド OpenShift クラスターを最大限に活用するのに役立つ、Red Hat の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です。

クラウドエキスパートによるこのチュートリアルコンテンツの一部は、迅速な提供のために、サポート対象のすべての構成でテストされていない場合があります。

第2章 チュートリアル: ROSA WITH HCP のアクティベーション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、Hosted Control Plane (HCP) を使用して Red Hat OpenShift Service on AWS (ROSA) をアクティブ化し、AWS アカウントにリンクするプロセスについて説明します。

重要

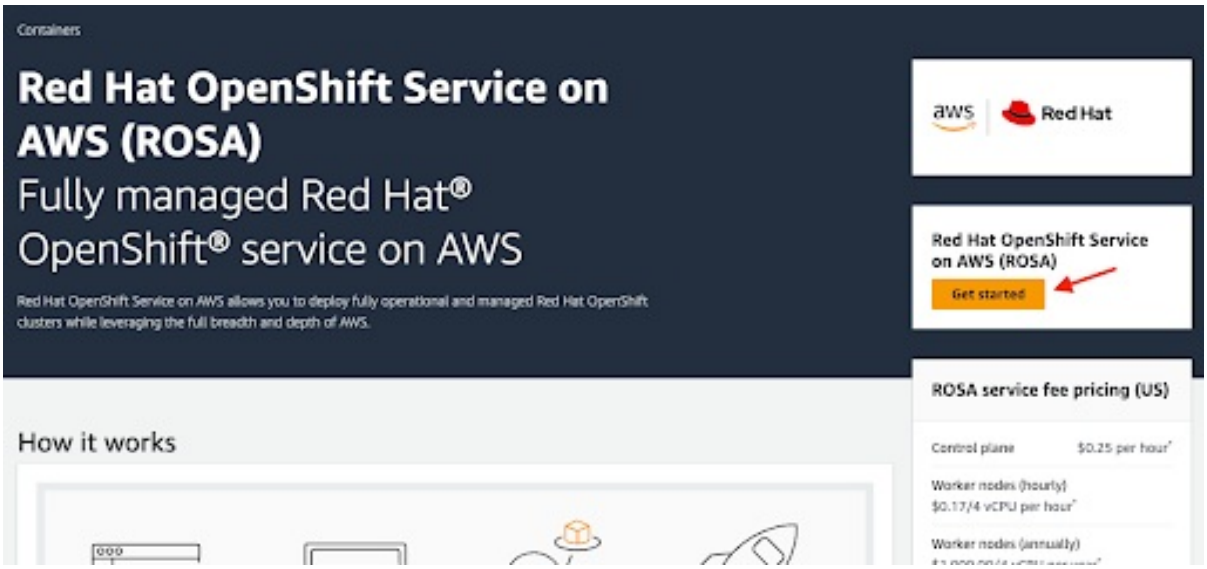
製品のプライベートオファーを受け取っている場合、このチュートリアルを始める前に、プライベートオファーに記載されている手順に従ってください。プライベートオファーは、製品がすでにアクティブ化されている場合 (この場合はアクティブなサブスクリプションを置き換えます)、また初めてアクティブ化する場合に利用できるように設計されています。

2.1. 前提条件

- 前の手順で ROSA with HCP をアクティブ化した AWS アカウントに関連付ける予定の Red Hat アカウントにログインしてください。
- Red Hat アカウントに関連付けることができるのは、サービスの請求に使用される AWS アカウント 1 つのみです。通常、個々の AWS エンドユーザーアカウントではなく、開発者アカウントなどの他の AWS アカウントがリンクされている組織の AWS アカウントが請求の対象となります。
- 同じ Red Hat 組織に属する Red Hat アカウントは、同じ AWS アカウントにリンクされます。したがって、ROSA with HCP クラスターの作成にアクセスできるユーザーを Red Hat 組織アカウントレベルで管理できます。

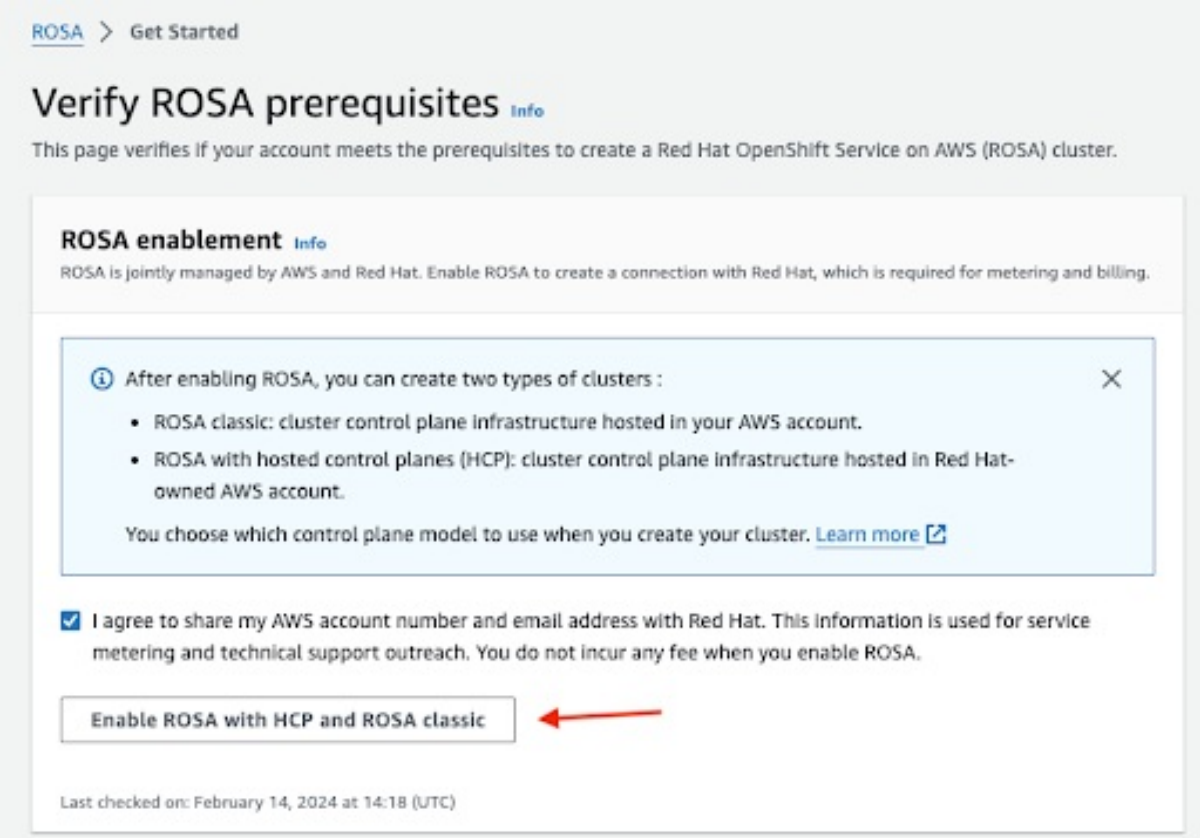
2.2. サブスクリプションの有効化と AWS アカウントのセットアップ

1. AWS コンソールページで **Get started** ボタンをクリックして、ROSA with HCP 製品をアクティブ化します。



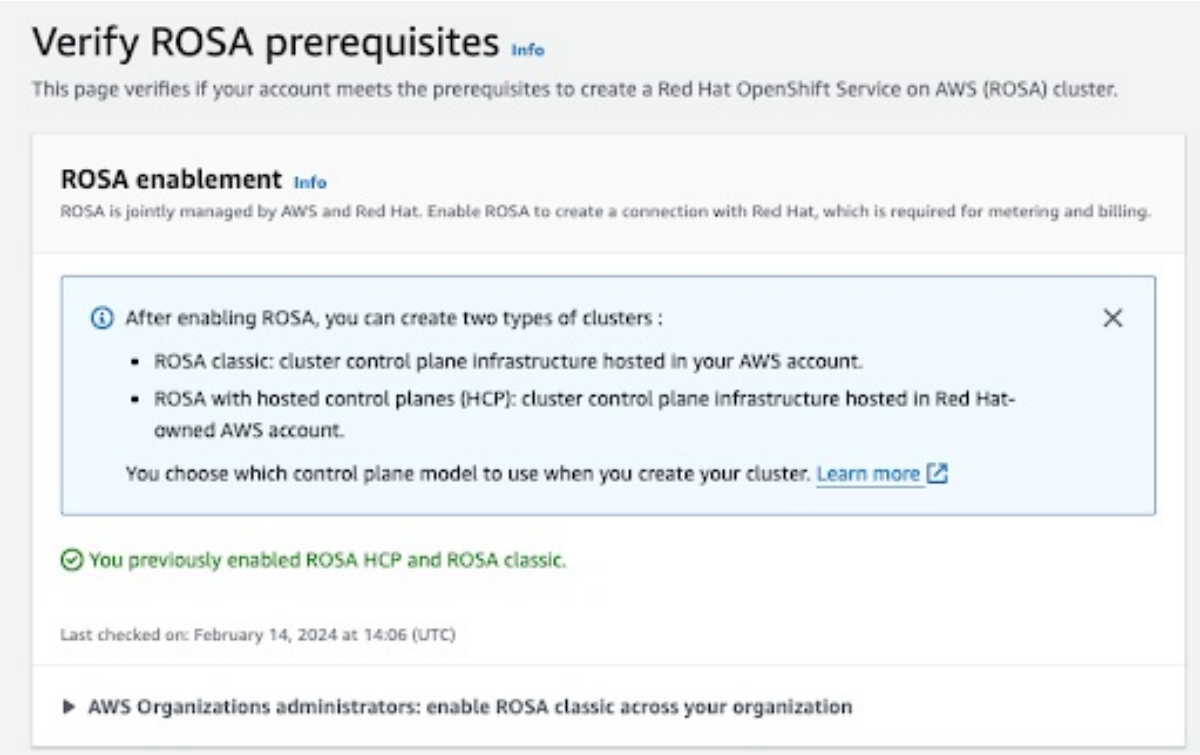
   以前に ROSA をアクティブ化したが、プロセスを完了していない場合は、ボタンをクリックして、次の手順で説明するようにアカウントのリンクを完了できます。

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。

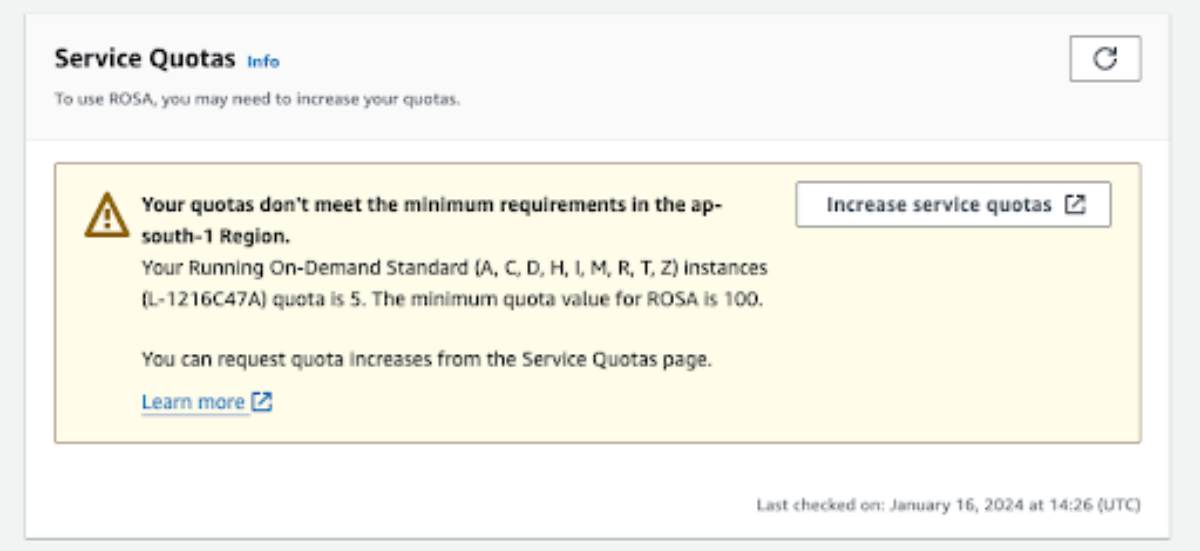


- このステップでサービスを有効にしても料金は発生しません。課金と計測が連携されるのは、最初のクラスターをデプロイした後のみです。これには数分かかる場合があります。

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。

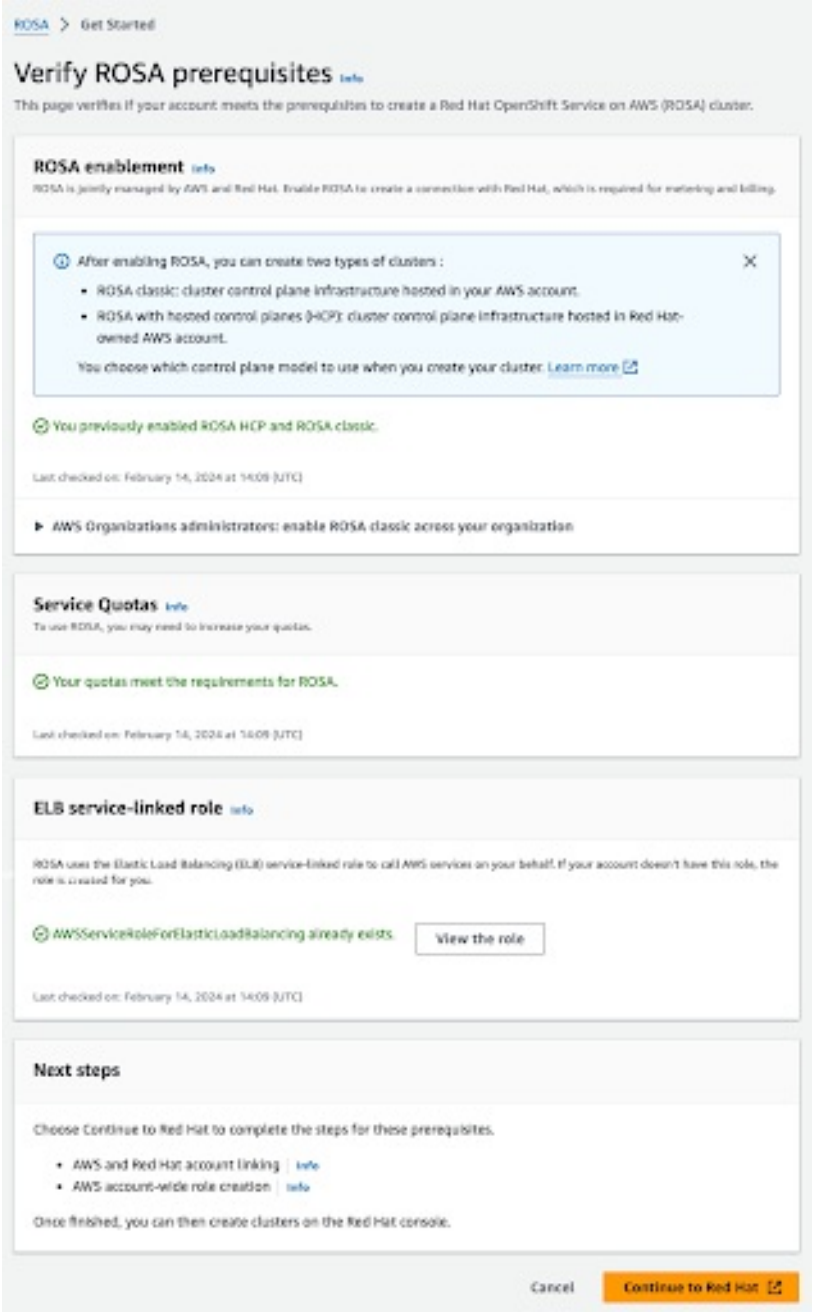


4. この検証ページの他のセクションには、追加の前提条件 (が満たされているかどうか) のステータスが表示されます。これらの前提条件のいずれかが満たされていない場合は、それぞれのメッセージが表示されます。選択したリージョンで割り当てが不十分な例を次に示します。



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。

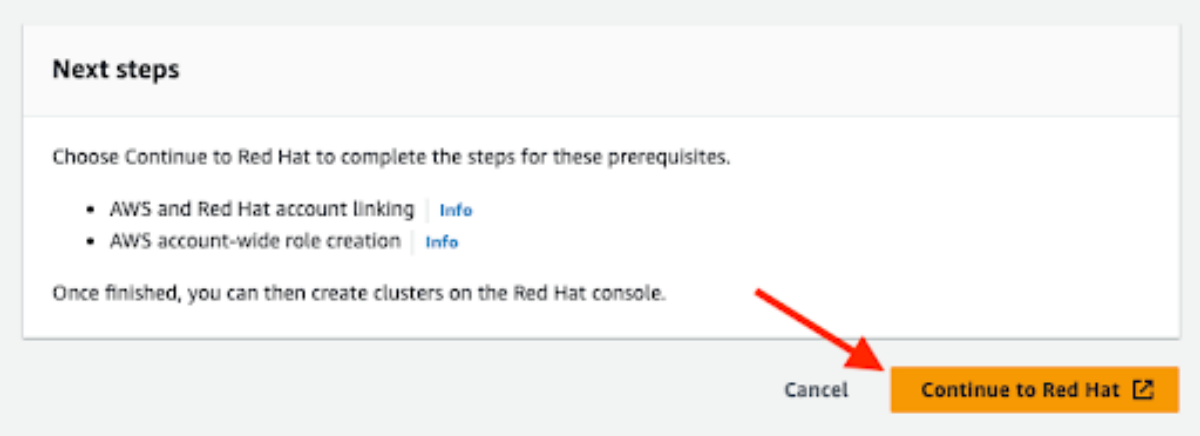
5. すべての前提条件が満たされている場合、ページは次のようになります。



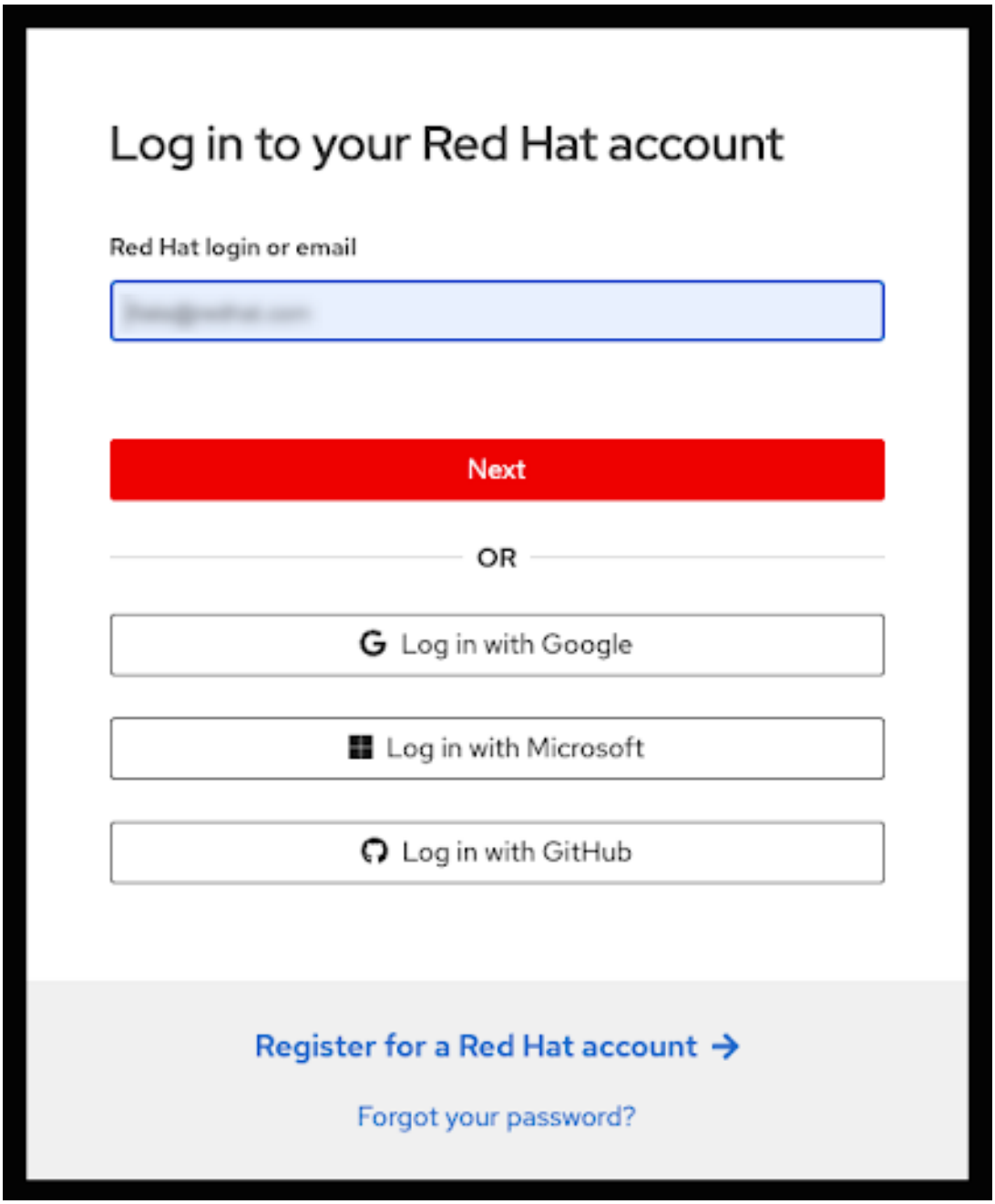
ELB サービスにリンクされたロールは自動的に作成されます。青色の小さな **Info** リンクをクリックすると、状況に応じたヘルプやリソースが表示されます。

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。



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。



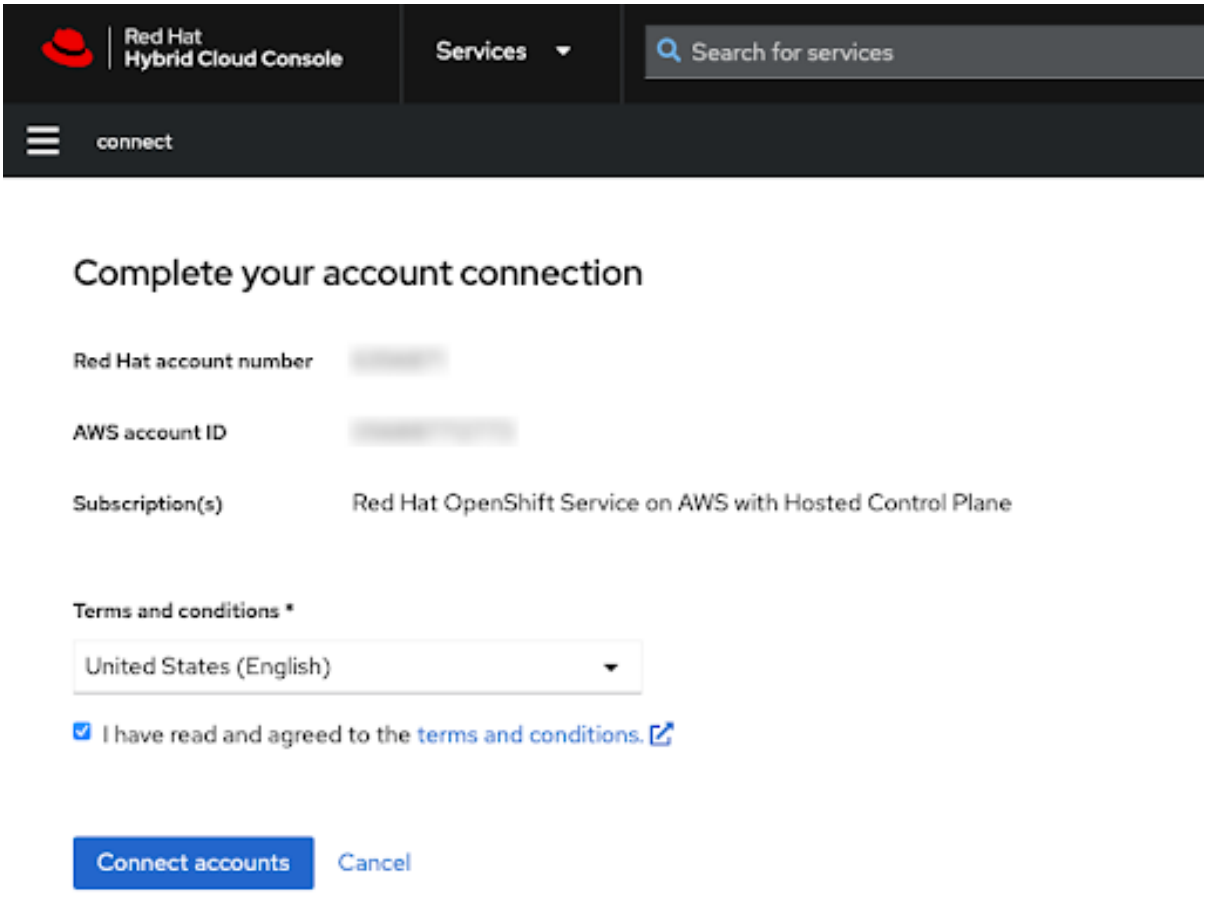
- このページでは、新しい Red Hat アカウントに登録したり、パスワードをリセットしたりすることもできます。

- 前の手順で ROSA with HCP をアクティブ化した AWS アカウントに関連付ける予定の Red Hat アカウントにログインしてください。
- Red Hat アカウントに関連付けることができるのは、サービスの請求に使用される AWS アカウント 1 つのみです。通常、個々の AWS エンドユーザーアカウントではなく、開発者アカウントなどの他の AWS アカウントがリンクされている組織の AWS アカウントが請求の対象となります。
- 同じ Red Hat 組織に属する Red Hat アカウントは、同じ AWS アカウントにリンクされます。したがって、ROSA with HCP クラスターの作成にアクセスできるユーザーを Red Hat 組織アカウントレベルで管理できます。

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。

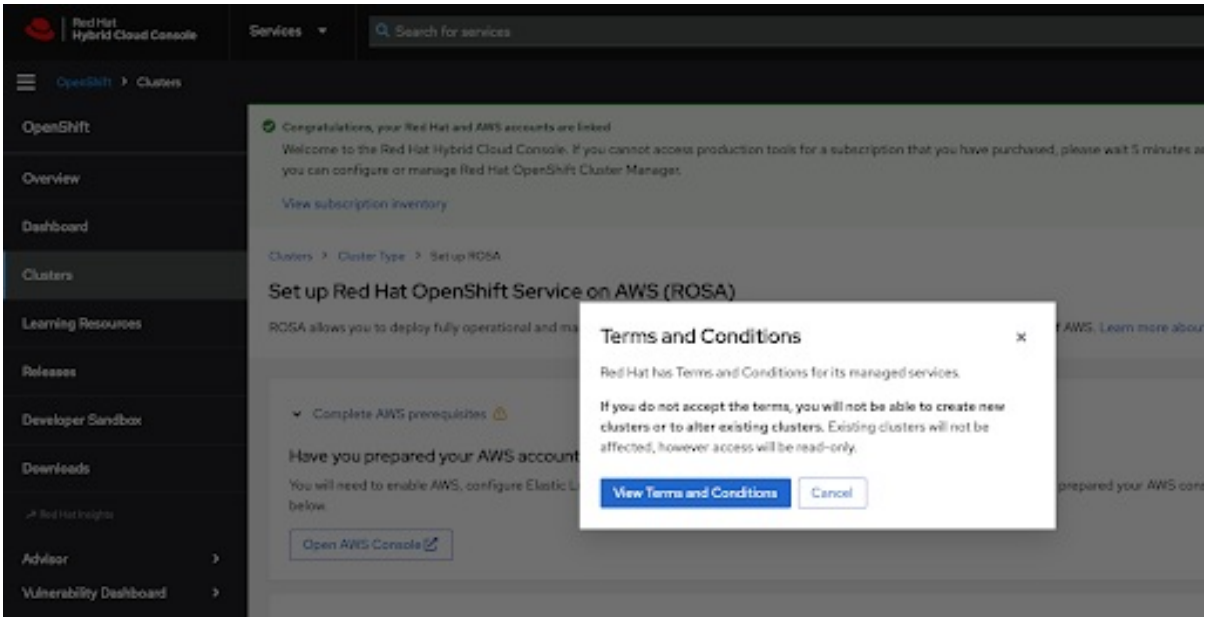
注記

このステップは、ログインしている Red Hat アカウント、または Red Hat アカウントを管理する Red Hat 組織が以前に AWS アカウントにリンクされていない場合にのみ使用できます。

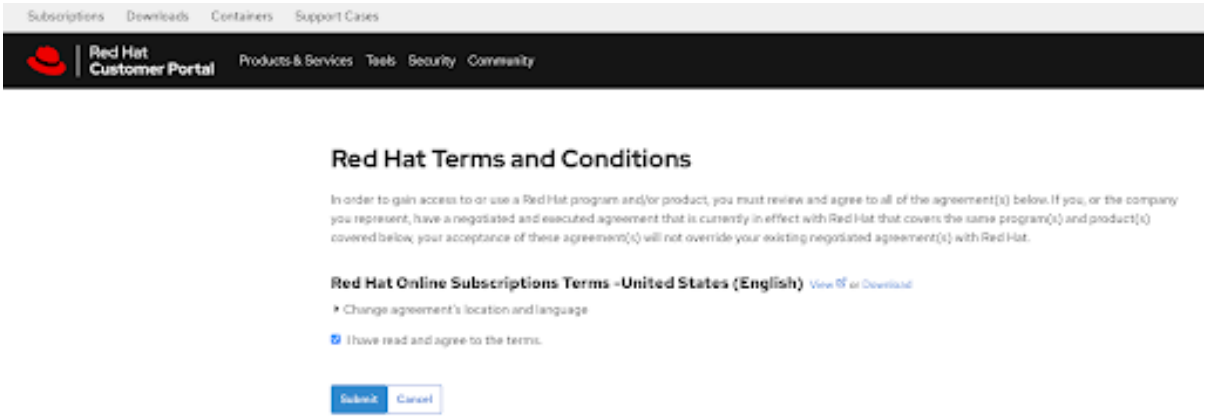


Red Hat と AWS の両方のアカウント番号がこの画面に表示されます。

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 ROSA クラスターを作成する前に、一般的なマネージドサービスの利用規約に同意するよう求められます。

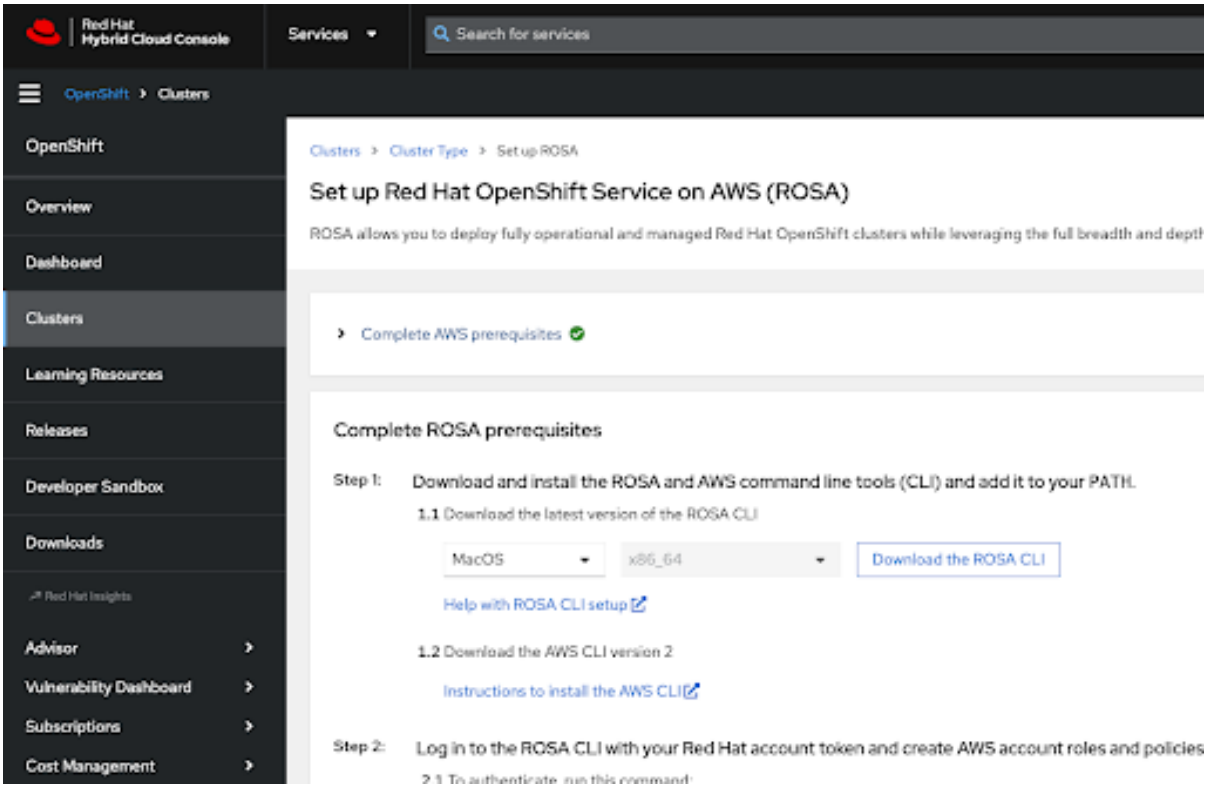


View Terms and Conditions ボタンをクリックすると、確認して同意する必要がある追加の規約が表示されます。

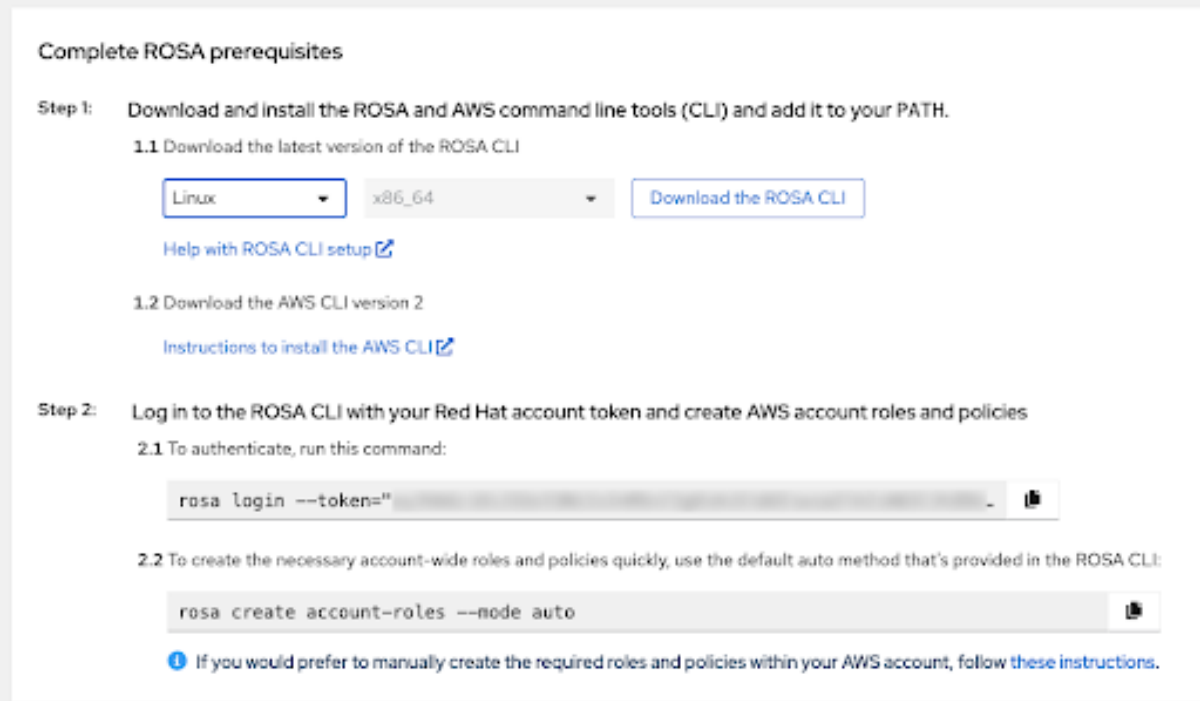


この時点で追加の条件を確認し、プロンプトが表示されたら同意を確定します。

5. Hybrid Cloud Console は、AWS の前提条件に対応済みであることを認識し、クラスターデプロイメントに必要な最初の手順がリストされます。

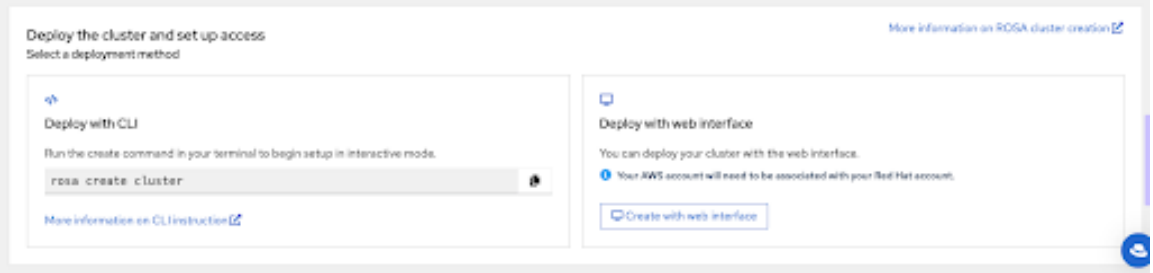


6. 次の手順は、クラスターの技術的なデプロイメントに関連するものです。



- これらの手順は、サービスの有効化とアカウントのリンクが完了したマシンとは別のマシンで実行される可能性があります。

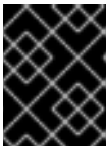
- 前述したように、Red Hat アカウントが、ROSA サービスをアクティブ化した AWS アカウントにリンクされた Red Hat 組織に属する場合は、クラスターの作成権限があり、以前に Red Hat 組織にリンクされていた請求 AWS アカウントを選択できます。
このページの最後のセクションでは、**rosa** CLI または Web コンソールを使用したクラスターデプロイメントオプションを示します。



- 次の手順では、**rosa** CLI を使用したクラスターデプロイメントについて説明します。
- Web コンソールのみを使用したデプロイメントを行う場合は、**ROSA with HCP cluster deployment using the web console** セクションまでスキップできます。ただし、アカウントロールの作成など、特定のタスクには **rosa** CLI が必要であることに注意してください。ROSA を初めてデプロイする場合は、**rosa whoami** コマンドを実行するまで、この CLI の手順に従い、その後はスキップして Web コンソールのデプロイメント手順を実行してください。

2.4. CLI を使用した ROSA WITH HCP クラスターのデプロイメント

1. **Download the ROSA CLI**ボタンをクリックして、オペレーティングシステム用の ROSA コマンドラインインターフェイス (CLI) をダウンロードし、Help with ROSA CLI setup の説明に従ってセットアップします。

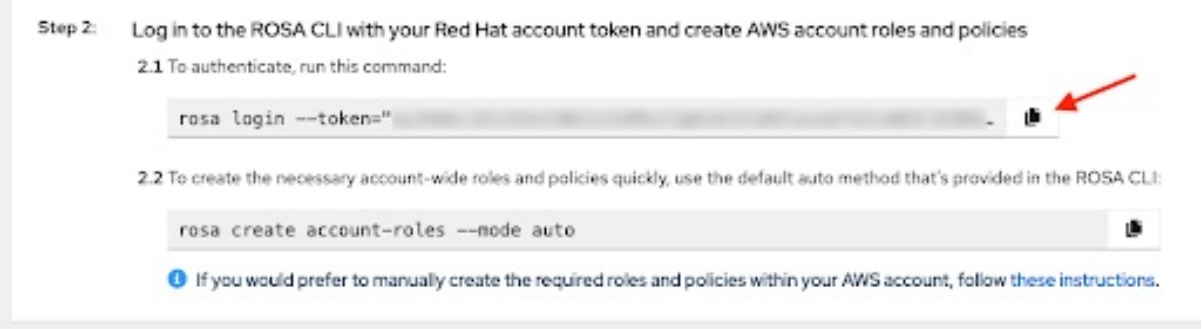


重要

最新の AWS CLI がインストールされていることを確認してください。詳細は AWS CLI をインストールする手順 を参照してください。

2. 前の手順が完了したら、**rosa version** を実行して、両方の CLI が使用できることを確認できます。このコマンドは、ターミナルで古いバージョンと **aws –version** コマンドを使用している場合に更新通知を表示します。

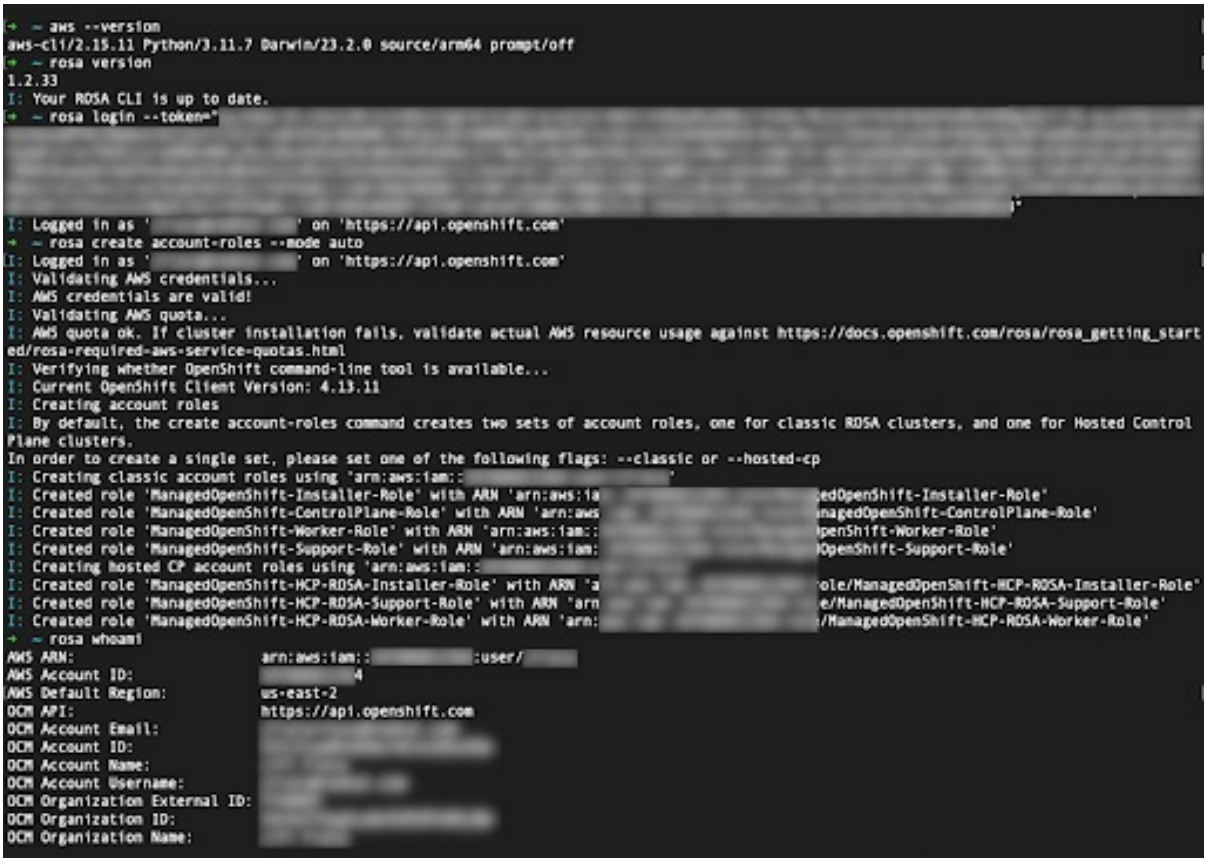
3. ROSA with HCP クラスターを作成するための前提条件は、Web コンソールで **rosa** cli を使用して、一意のトークンで独自のコマンドを実行し、ログインしてください。この説明は、**2.1To authenticate, run this command** を参照してください。**copy** ボタンを使用すると、完全なトークンを含むコマンドを簡単にコピーしてターミナルに貼り付けることができます。



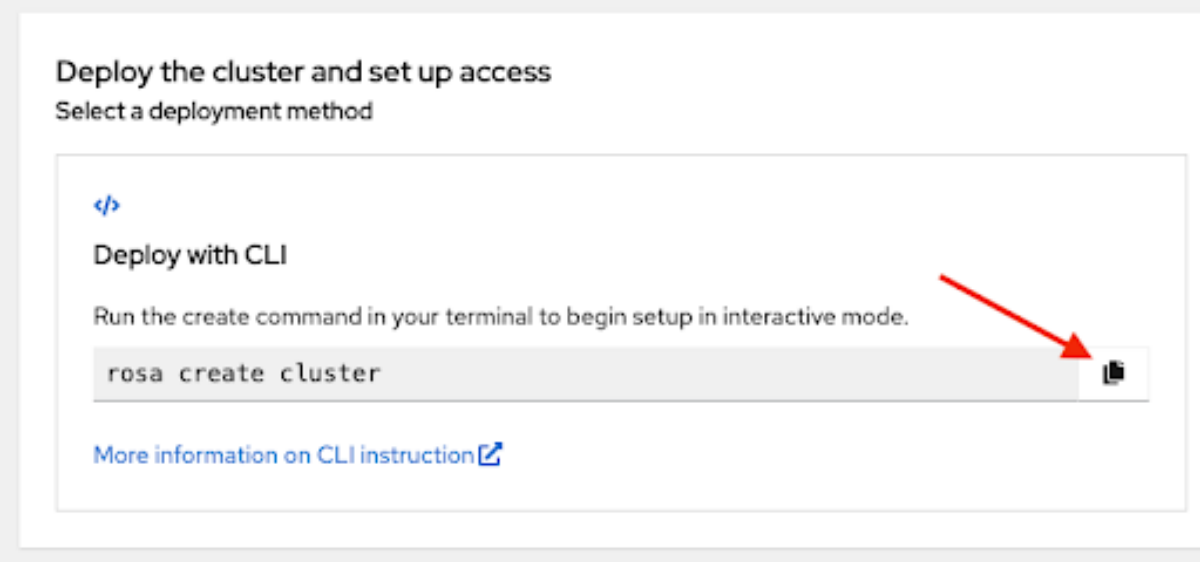
独自のトークンを共有しないでください。

4. 初めてクラスターをデプロイするための最後の前提条件として、必要とされるアカウント全体のロールおよびポリシーを作成します。**rosa** CLI で、該当するコマンドを使用できます。方法は、**2.2必要とされるアカウント全体のロールおよびポリシーを素早く作成する方法** を参照してください。これに代わる方法は、これらのロールとポリシーを手動で作成することです。

5. ログインし、アカウントロールを作成して、**rosa whoami** コマンドを使用し、アイデンティティーを確認します。この操作のターミナルの内容は、次のようになります。



6. 表示されたコマンドを使用してクラスターのデプロイメントを開始します。もう一度 **copy** ボタンをクリックして、ターミナルにコマンドを貼り付けます。



7. 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。

8. クラスター名を入力すると、Hosted Control Plane を使用するかどうかを尋ねられます。**YES** を選択します。

```
➜  ~ rosa create cluster
I: Enabling interactive mode
? Cluster name: test-cluster-07
? Deploy cluster with Hosted Control Plane: [? for help] (y/N) y
```

9. ROSA with HCP クラスターをデプロイする場合、請求先の AWS アカウントを指定する必要があります。

```
➜  ~ rosa create cluster
I: Enabling interactive mode
? Cluster name: test-cluster-07
? Deploy cluster with Hosted Control Plane: Yes
? Billing Account:  [Use arrows to move, type to filter, ? for more help]
> [redacted] [Contract enabled]
  [redacted]
```

- 現在使用されている Red Hat 組織にリンクされている AWS アカウントのみが表示されます。
- インフラストラクチャー AWS アカウントが同じ AWS 組織内でそのアカウントにリンクされているかどうかに関係なく、指定された AWS アカウントは ROSA サービスの使用に対して課金されます。
- 特定の AWS 請求先アカウントに対して ROSA の契約が有効になっているかどうかを示すインジケーターが表示されます。
- 契約が有効になっていない AWS アカウントを選択するには、このチュートリアルの最初のいくつかの手順を参照して契約を有効にし、クラスターデプロイメントを正常に実行するために必要なサービス課金を許可します。

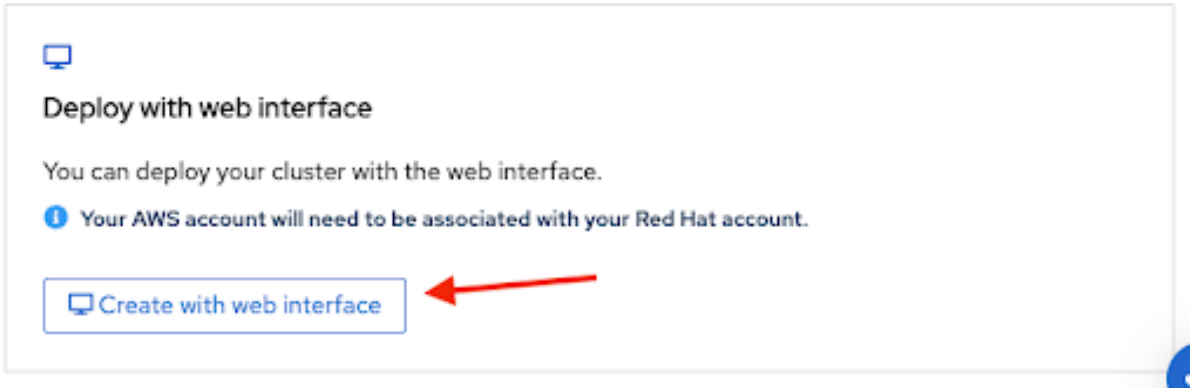
10. 次の手順では、デプロイするクラスターの技術的な詳細を指定します。

```
➜  ~ rosa create cluster
I: Enabling interactive mode
? Cluster name: test-cluster-07
? Deploy cluster with Hosted Control Plane: Yes
? Billing Account: [redacted] [Contract enabled]
I: Using '[redacted]' as billing account.
I:
   +--------------------+----------------+
   | Start Date         |Dec 08, 2023    |
   | End Date           |Aug 17, 2024    |
   | Number of vCPUs:   |[redacted]      |
   | Number of clusters:|[redacted]      |
   +--------------------+----------------+

? OpenShift version (default = '4.14.8'):  [Use arrows to move, type to filter, ? for more help]
> 4.14.8
  4.14.7
  4.14.6
  4.14.5
```

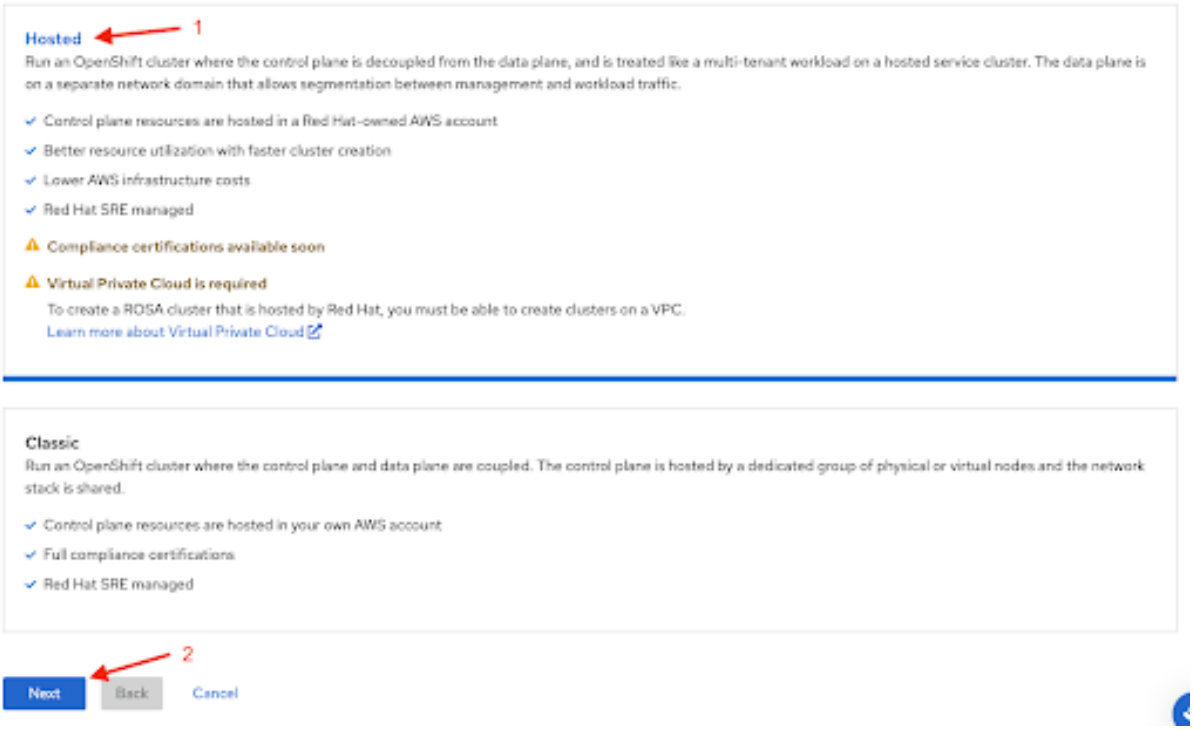
- これらの手順は、このチュートリアルの範囲外です。CLI を使用して ROSA with HCP クラスターのデプロイメントを完了する方法の詳細は デフォルトオプションを使用した ROSA with HCP クラスターの作成 を参照してください。

2.5. WEB コンソールを使用した ROSA WITH HCP クラスターのデプロイメント

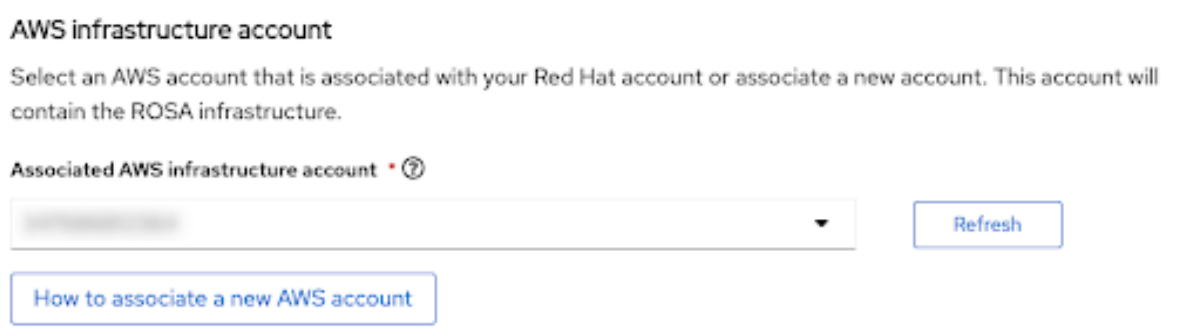
1. Web コンソールを使用してクラスターを作成するには、**ROSA のセットアップ** ページの下部セクションにある 2 番目のオプションを選択します。



2. Web コンソールを使用して ROSA 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。



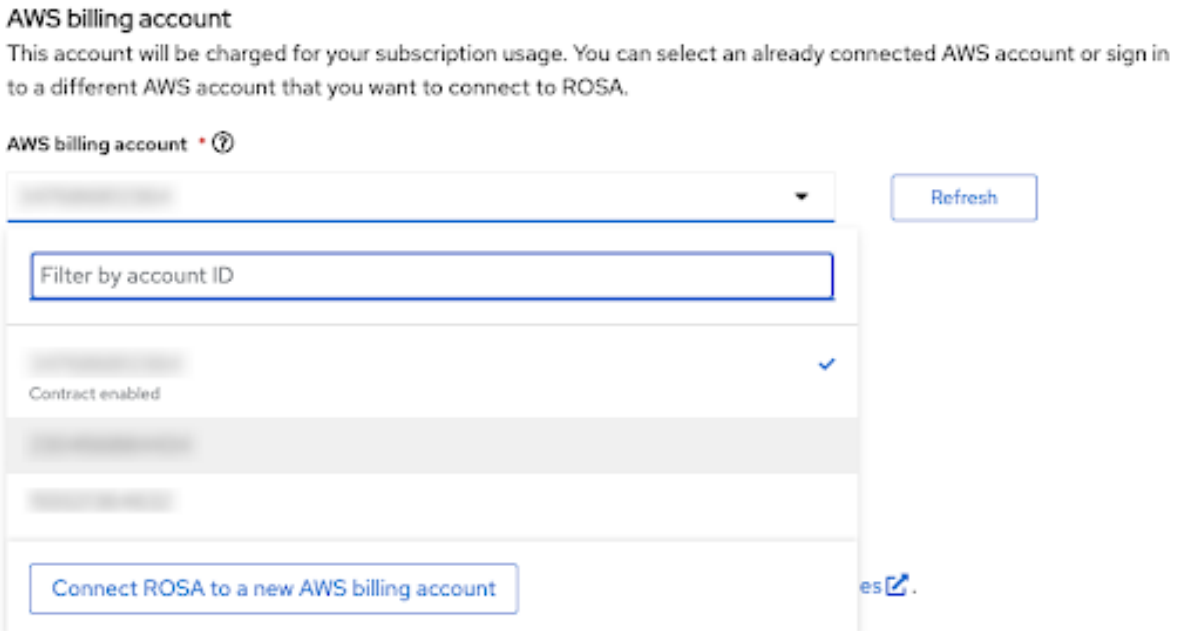
3. 次のステップ **アカウントとロール** では、ROSA クラスターがデプロイされ、リソースを消費および管理するインフラストラクチャー AWS アカウントを指定できます。



- ROSA 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。



- 現在使用されている Red Hat 組織にリンクされている AWS アカウントのみが表示されます。
- インフラストラクチャー AWS アカウントが同じ AWS 組織内でそのアカウントにリンクされているかどうかに関係なく、指定された AWS アカウントは ROSA サービスの使用に対して課金されます。
- 特定の AWS 請求先アカウントに対して ROSA 契約が有効になっているかどうかを示すインジケーターを確認できます。
- 契約がまだ有効になっていない AWS アカウントを使用したい場合は、**Connect ROSA to a new AWS billing account** を使用して ROSA AWS コンソールページにアクセスし、ログイン後に有効化できます。先ほど、このチュートリアルで説明した手順に従って、それぞれの AWS アカウントを作成するか、AWS アカウントの管理者に依頼してください。

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 ROSA with HCP 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 ROSA クラスターのデプロイメントを完了する方法の詳細は、この学習パス を参照してください。

第3章 チュートリアル: ROSA STS デプロイメントの権限の検証

ROSA クラスターのデプロイに進むには、アカウントによって必要なロールと権限がサポートされている必要があります。AWS Service Control Policies (SCP) は、インストーラーまたは Operator ロールによって行われる API 呼び出しをブロックできません。

ROSA の STS 対応インストールに必要な IAM リソースの詳細は、STS を使用する ROSA クラスターの IAM リソースについて を参照してください。

このガイドは ROSA v4.11.X で検証されています。

3.1. 前提条件

- AWS CLI
- ROSA CLI v1.2.6
- jq CLI
- 必要な権限を持つ AWS ロール

3.2. ROSA 権限の検証

ROSA に必要な権限を検証するには、AWS リソースを作成していない状態で、次のセクションに示すスクリプトを実行します。

このスクリプトは、**rosa**、**aws**、および **jq** CLI コマンドを使用して、現在の AWS 設定に接続されているアカウントの権限の検証に使用されるファイルを作業ディレクトリーに作成します。

AWS Policy Simulator を使用して、**jq** によって抽出された API 呼び出しに対して各ロールポリシーの権限を検証します。結果は、**.results** が追加されたテキストファイルに保存されます。

このスクリプトは、現在のアカウントとリージョンの権限を確認するように設計されています。

3.3. 使用手順

1. スクリプトを使用するには、**bash** ターミナルで次のコマンドを実行します (-p オプションはロールの接頭辞を定義します)。

```
$ mkdir scratch
$ cd scratch
$ cat << 'EOF' > verify-permissions.sh
#!/bin/bash
while getopts 'p:' OPTION; do
  case "$OPTION" in
    p)
      PREFIX="$OPTARG"
      ;;
    ?)
      echo "script usage: $(basename \$0) [-p PREFIX]" >&2
      exit 1
      ;;
  esac
done
```

```
shift "$(($OPTIND -1))"
rosa create account-roles --mode manual --prefix $PREFIX
INSTALLER_POLICY=$(cat sts_installer_permission_policy.json | jq )
CONTROL_PLANE_POLICY=$(cat sts_instance_controlplane_permission_policy.json | jq)
WORKER_POLICY=$(cat sts_instance_worker_permission_policy.json | jq)
SUPPORT_POLICY=$(cat sts_support_permission_policy.json | jq)
simulatePolicy () {
    outputFile="${2}.results"
    echo $2
    aws iam simulate-custom-policy --policy-input-list "$1" --action-names $(jq '.Statement |
map(select(.Effect == "Allow"))[].Action | if type == "string" then . else .[] end' "$2" -r) --output
text > $outputFile
}
simulatePolicy "$INSTALLER_POLICY" "sts_installer_permission_policy.json"
simulatePolicy "$CONTROL_PLANE_POLICY"
"sts_instance_controlplane_permission_policy.json"
simulatePolicy "$WORKER_POLICY" "sts_instance_worker_permission_policy.json"
simulatePolicy "$SUPPORT_POLICY" "sts_support_permission_policy.json"
EOF
$ chmod +x verify-permissions.sh
$ ./verify-permissions.sh -p SimPolTest
```

2. スクリプトが完了したら、各結果ファイルを確認して、必要な API 呼び出しがブロックされていないことを確認します。

```
$ for file in $(ls *.results); do echo $file; cat $file; done
```

出力は以下のようになります。

```
sts_installer_permission_policy.json.results
EVALUATIONRESULTS       autoscaling:DescribeAutoScalingGroups   allowed *
MATCHEDSTATEMENTS       PolicyInputList.1       IAM Policy
ENDPOSITION     6       195
STARTPOSITION   17      3
EVALUATIONRESULTS       ec2:AllocateAddress     allowed *
MATCHEDSTATEMENTS       PolicyInputList.1       IAM Policy
ENDPOSITION     6       195
STARTPOSITION   17      3
EVALUATIONRESULTS       ec2:AssociateAddress    allowed *
MATCHEDSTATEMENTS       PolicyInputList.1       IAM Policy
...
```

注記

アクションがブロックされている場合は、AWS が提供するエラーを確認し、管理者に相談して、SCP によって必要な API 呼び出しがブロックされているかどうかを判断してください。

第4章 CLOUDWATCH ログと STS のログ転送の設定

このチュートリアルでは、Red Hat OpenShift Logging Operator をデプロイし、Security Token Services (STS) 認証を使用してログを CloudWatch に転送するように設定します。

前提条件

- Red Hat OpenShift Service on AWS (ROSA) Classic クラスター
- **jq** コマンドラインインターフェイス (CLI)
- Amazon Web Services (AWS) CLI (**aws**)

4.1. 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。

注記

管理者としてログインしている必要があります。

```
$ export ROSA_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 AWS_PAGER=""
$ export SCRATCH="/tmp/${ROSA_CLUSTER_NAME}/clf-cloudwatch-sts"
$ 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}"
```

4.2. AWS アカウントの準備

1. ロギング用の Identity Access Management (IAM) ポリシーを作成します。

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

```
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
            "logs:DescribeLogGroups",
            "logs:DescribeLogStreams",
            "logs:PutLogEvents",
            "logs:PutRetentionPolicy"
          ],
          "Resource": "arn:aws:logs:*:*:*"
      }
]
}
EOF
POLICY_ARN=$(aws iam create-policy --policy-name "RosaCloudWatch" \
--policy-document file:///${SCRATCH}/policy.json --query Policy.Arn --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-logging:logcollector"
      }
    }
  }]
}
EOF
$ ROLE_ARN=$(aws iam create-role --role-name "${ROSA_CLUSTER_NAME}-
RosaCloudWatch" \
      --assume-role-policy-document file://${SCRATCH}/trust-policy.json \
      --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 "${ROSA_CLUSTER_NAME}-RosaCloudWatch" \
  --policy-arn ${POLICY_ARN}
```

4.3. OPERATOR のデプロイ

1. Red Hat OpenShift Logging Operator をデプロイします。

```
$  cat << EOF | oc apply -f -
  apiVersion: operators.coreos.com/v1alpha1
  kind: Subscription
```

```
  metadata:
    labels:
     operators.coreos.com/cluster-logging.openshift-logging: ""
    name: cluster-logging
    namespace: openshift-logging
  spec:
    channel: stable
    installPlanApproval: Automatic
    name: cluster-logging
    source: redhat-operators
    sourceNamespace: openshift-marketplace
EOF
```

2. シークレットを作成します。

```
$  cat << EOF | oc apply -f -
  apiVersion: v1
  kind: Secret
  metadata:
    name: cloudwatch-credentials
    namespace: openshift-logging
  stringData:
    role_arn: $ROLE_ARN
EOF
```

4.4. クラスターロギングの設定

1. **ClusterLogForwarder** カスタムリソース (CR) を作成します。

```
$ cat << EOF | oc apply -f -
  apiVersion: "logging.openshift.io/v1"
  kind: ClusterLogForwarder
  metadata:
    name: instance
    namespace: openshift-logging
  spec:
    outputs:
      - name: cw
        type: cloudwatch
        cloudwatch:
          groupBy: namespaceName
          groupPrefix: rosa-${ROSA_CLUSTER_NAME}
          region: ${REGION}
        secret:
          name: cloudwatch-credentials
    pipelines:
      - name: to-cloudwatch
        inputRefs:
          - infrastructure
          - audit
          - application
        outputRefs:
          - cw
EOF
```

2. **ClusterLogging** CR を作成します。

```
$ cat << EOF | oc apply -f -
  apiVersion: logging.openshift.io/v1
  kind: ClusterLogging
  metadata:
    name: instance
    namespace: openshift-logging
  spec:
    collection:
      logs:
        type: vector
    managementState: Managed
EOF
```

4.5. CLOUDWATCH のログの確認

- AWS コンソールまたは AWS CLI を使用して、クラスターからのログストリームがあることを検証します。
  - AWS CLI でログを検証するには、次のコマンドを実行します。

```
$ aws logs describe-log-groups --log-group-name-prefix rosa-
${ROSA_CLUSTER_NAME}
```

出力例

```
{
  "logGroups": [
      {
        "logGroupName": "rosa-xxxx.audit",
        "creationTime": 1661286368369,
        "metricFilterCount": 0,
        "arn": "arn:aws:logs:us-east-2:xxxx:log-group:rosa-xxxx.audit:*",
        "storedBytes": 0
      },
      {
        "logGroupName": "rosa-xxxx.infrastructure",
        "creationTime": 1661286369821,
        "metricFilterCount": 0,
        "arn": "arn:aws:logs:us-east-2:xxxx:log-group:rosa-xxxx.infrastructure:*",
        "storedBytes": 0
      }
  ]
}
```

注記

これが新しいクラスターの場合は、アプリケーションがまだ実行されていないため、**application** ログのロググループが表示されない可能性があります。

4.6. リソースのクリーンアップ

1. **ClusterLogForwarder** CR を削除します。

```
$ oc delete -n openshift-logging clusterlogforwarder instance
```

2. **ClusterLogging** CR を削除します。

```
$ oc delete -n openshift-logging clusterlogging instance
```

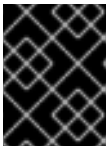
3. IAM ロールから IAM ポリシーの割り当てを解除します。

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

4. IAM ロールを削除します。

```
$ aws iam delete-role --role-name "${ROSA_CLUSTER_NAME}-RosaCloudWatch"
```

5. IAM ポリシーを削除します。



重要

IAM ポリシーは、そのポリシーを使用している他のリソースがない場合にのみ削除してください。

```
$ aws iam delete-policy --policy-arn "${POLICY_ARN}"
```

6. CloudWatch ロググループを削除します。

```
$ aws logs delete-log-group --log-group-name "rosa-${ROSA_CLUSTER_NAME}.audit"
$ aws logs delete-log-group --log-group-name "rosa-
${ROSA_CLUSTER_NAME}.infrastructure"
```

第5章 チュートリアル: AWS WAF と AMAZON CLOUDFRONT を使用した ROSA 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 (ROSA) 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 ROSA リソースを保護できます。

5.1. 前提条件

- ROSA Classic 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 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_NAME}/cloudfront-waf"
$ mkdir -p ${SCRATCH}
$ echo "Cluster: ${CLUSTER_NAME}, Region: ${REGION}, AWS Account ID:
${AWS_ACCOUNT_ID}"
```

5.2. カスタムドメインの設定

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。ROSA では、Custom Domain Operator を使用してこれを行います。

前提条件

- 一意のドメイン (***.apps.<company_name>.io** など)
- カスタム SAN またはワイルドカード証明書 (**CN=*.apps.<company_name>.io** など)

手順

1. 新しいプロジェクトを作成する

```
$ oc new-project waf-demo
```

2. 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 waf-demo create secret tls waf-tls --cert=fullchain.pem --key=privkey.pem
   ```

3. 新規の **CustomDomain** カスタムリソース (CR) を作成します。

   waf-custom-domain.yaml の例

   ```
   apiVersion: managed.openshift.io/v1alpha1
   kind: CustomDomain
   metadata:
     name: cloudfront-waf
   spec:
     domain: apps.<company_name>.io (1)
     scope: External
     loadBalancerType: NLB
     certificate:
       name: waf-tls
       namespace: waf-demo
     routeSelector: (2)
       matchLabels:
        route: waf
   ```

   (1) カスタムドメイン。

   (2) CustomDomain ingress によって提供されるルートのセットをフィルターします。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。

4. CR を適用します。

   例

   ```
   $ oc apply -f waf-custom-domain.yaml
   ```

5. カスタムドメイン Ingress Controller がデプロイされ、**Ready** ステータスになっていることを確認します。

   ```
   $ oc get customdomains
   ```

   出力例

   ```
   NAME            ENDPOINT                                        DOMAIN                  STATUS
   cloudfront-waf  xxrywp.<company_name>.cluster-01.opln.s1.openshiftapps.com  *.apps.
   <company_name>.io  Ready
   ```

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。ROSA 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
```

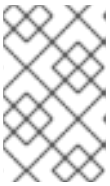
```
SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=${CLUSTER_NAME}-waf-metrics \
  --rules file://${SCRATCH}/waf-rules.json \
  --query 'Summary.Name' \
  --output text)
```

5.3. AMAZON CLOUDFRONT の設定

1. 新しく作成したカスタムドメイン 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}')
$ echo "Origin domain: ${NLB}"
```

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

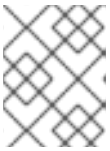
この証明書は、クラスターがデプロイされているリージョンに関係なく **us-east-1** にインポートする必要があります。Amazon CloudFront はグローバル AWS サービスであるためです。

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	上記のコマンドの出力 [1]
Name	rosa-waf-ingress [2]
Viewer protocol policy	Redirect HTTP to HTTPS
Allowed HTTP methods	GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE

オプション	値
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.<company_name>.io [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、カスタムドメイン 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.<company_name>.io CNAME d1b2c3d4e5f6g7.cloudfront.net
```

5.4. サンプルアプリケーションのデプロイ

1. Hello World アプリケーションをデプロイします。

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

2. 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 waf-demo create route edge --service=hello-openshift hello-openshift-tls \
--hostname hello-openshift.apps.<company_name>.io
```

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

```
$ oc -n waf-demo 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.apps.<company_name>.io"
```

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.apps.<company_name>.io" \
  -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
```

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。

5.6. 関連情報

- Red Hat ドキュメントの アプリケーションのカスタムドメイン
- 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 ROSA 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、AWS Application Load Balancer (ALB) を Red Hat OpenShift Service on AWS (ROSA) 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 ROSA リソースを保護できます。

注記

ALB ベースのソリューションを使用する必要がある場合を除き、CloudFront 方式 を使用することを推奨します。

6.1. 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 に分割された 3 つのパブリックサブネットが必要です。

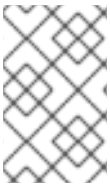
- マルチ AZ ROSA Classic クラスター。
- OpenShift CLI (**oc**) にアクセスできる。
- AWS CLI (**aws**) にアクセスできる。

6.1.1. 環境設定

- 環境変数を準備します。

```
$ 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 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_NAME}/alb-waf"
$ mkdir -p ${SCRATCH}
$ echo "Cluster: ${CLUSTER_NAME}, 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、ROSA 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>
```

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}
```

6.2. AWS LOAD BALANCER OPERATOR のデプロイ

AWS Load Balancer Operator は、ROSA 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。ROSA 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. 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/docs/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=$(aws iam create-role --role-name "${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 "${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. Red Hat 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
  enabledAddons:
    - AWSWAFv2
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
```

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
```

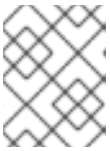
3. 事前に作成済みのサービスリソースを NodePort サービスタイプに変換します。

```
$ 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。ROSA 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_NAME}-waf \
  --region ${REGION} \
  --default-action Allow={} \
```

```
  --scope REGIONAL \
  --visibility-config
SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=${CLUSTER_NAME}-waf-metrics \
  --rules file://${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
```

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

- Red Hat ドキュメントの アプリケーションのカスタムドメイン
- YouTube の Adding Extra Security with AWS WAF, CloudFront and ROSA | Amazon Web Services

第7章 チュートリアル: ROSA 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

- ROSA Classic クラスター

環境

- 環境変数を準備します。

注記

ROSA クラスターに一致するように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

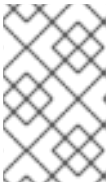
1. OADP の namespace を作成します。

```
$ 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
EOF
$ oc -n openshift-adp create secret generic cloud-credentials \
 --from-file=${SCRATCH}/credentials
```

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
```

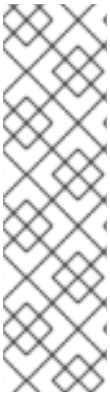
```
    - aws
    - csi
  restic:
    enable: false
EOF
```

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 ROSA 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、ROSA 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、ROSA 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については、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 backup hello-world
$ oc delete restore 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: ROSA 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 (ROSA) 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
- 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、ROSA 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 に分割された 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 ROSA Classic 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、ROSA 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。

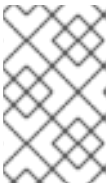
```
$ aws ec2 create-tags \
    --resources ${PUBLIC_SUBNET_IDS} \
    --tags Key=kubernetes.io/role/elb,Value='' \
    --region ${REGION}
```

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/docs/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
```

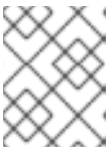
4. 新しく作成した AWS IAM ロールを引き受けるための AWS Load Balancer Operator 用のシークレットを作成します。

```
$ 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. Red Hat 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. 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. 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: INGRESS CONTROLLER でカスタム TLS 暗号を使用するように ROSA/OSD を設定する

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。

ここでは、クラスターの Ingress Controller と、Custom Domain Operator によって作成された Ingress Controller に適切にパッチを適用する方法を説明します。この機能を使用すると、クラスター Ingress Controller の **tlsSecurityProfile** 値を変更できます。ここでは、カスタムの **tlsSecurityProfile**、関連するロールとロールバインディングを持つスコープ指定されたサービスアカウント、および Ingress Controller が再作成または変更された場合に暗号の変更を 60 分で再適用する CronJob を適用する方法を説明します。

前提条件

- **tlsSecurityProfile** のオプションについて説明している OpenShift ドキュメント を確認している。デフォルトでは、Ingress Controller は、Intermediate Mozilla プロファイル に相当する **Intermediate** プロファイルを使用するように設定されています。

手順

1. CronJob が使用するサービスアカウントを作成します。
   サービスアカウントを使用すると、CronJob は通常のユーザーの認証情報を使用せずにクラスター API に直接アクセスできます。サービスアカウントを作成するには、次のコマンドを実行します。

   ```
   $ oc create sa cron-ingress-patch-sa -n openshift-ingress-operator
   ```

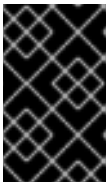
2. Ingress Controller にパッチを適用するための制限付きアクセスを許可するロールとロールバインディングを作成します。
   クラスター内のセキュリティーを確保するには、ロールベースのアクセス制御 (RBAC) が重要です。ロールを作成すると、クラスター内で必要な API リソースだけにスコープが限定されたアクセス権を提供できるようになります。ロールを作成するには、次のコマンドを実行します。

   ```
   $ oc create role cron-ingress-patch-role --verb=get,patch,update --
   resource=ingresscontroller.operator.openshift.io -n openshift-ingress-operator
   ```

   ロールを作成したら、ロールバインディングを使用してサービスアカウントにロールをバインドする必要があります。ロールバインディングを作成するには、次のコマンドを実行します。

   ```
   $ oc create rolebinding cron-ingress-patch-rolebinding --role=cron-ingress-patch-role --
   serviceaccount=openshift-ingress-operator:cron-ingress-patch-sa -n openshift-ingress-
   operator
   ```

3. Ingress Controller にパッチを適用します。



重要

以下に示す例では、Ingress Controller の **tlsSecurityProfile** に別の暗号を追加して、Windows Server 2008 R2 からの IE 11 アクセスを許可します。このコマンドはお客様固有のビジネス要件に合わせて変更してください。

CronJob を作成する前に、**tlsSecurityProfile** 設定を適用して変更を検証します。このプロセスは、Custom Domain Operator を使用するかどうかによって異なります。

a. Custom Domain Operator を使用しないクラスターの場合:
デフォルトの Ingress Controller のみを使用し、Custom Domain Operator を使用しない場合は、次のコマンドを実行して Ingress Controller にパッチを適用します。

```
$ oc patch ingresscontroller/default -n openshift-ingress-operator --type=merge -p
'{"spec":{"tlsSecurityProfile":{"type":"Custom","custom":{"ciphers":
["TLS_AES_128_GCM_SHA256","TLS_AES_256_GCM_SHA384","ECDHE-ECDSA-
AES128-GCM-SHA256","ECDHE-RSA-AES128-GCM-SHA256","ECDHE-ECDSA-
AES256-GCM-SHA384","ECDHE-RSA-AES256-GCM-SHA384","ECDHE-ECDSA-
CHACHA20-POLY1305","ECDHE-RSA-CHACHA20-POLY1305","DHE-RSA-AES128-
GCM-SHA256","DHE-RSA-AES256-GCM-
SHA384","TLS_CHACHA20_POLY1305_SHA256","TLS_ECDHE_RSA_WITH_AES_128
_CBC_SHA"],"minTLSVersion":"VersionTLS12"}}}}'
```

このパッチは、RSA 証明書の使用時に Windows Server 2008 R2 上の IE 11 からのアクセスを許可する **TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA** 暗号を追加します。

コマンドを実行すると、次のような応答が返されます。

出力例

```
ingresscontroller.operator.openshift.io/default patched
```

b. Custom Domain Operator を使用するクラスターの場合:
Custom Domain Operator を使用するお客様は、各 Ingress Controller をループ処理して、それぞれにパッチを適用する必要があります。クラスターのすべての Ingress Controller にパッチを適用するには、次のコマンドを実行します。

```
$ for ic in $(oc get ingresscontroller -o name -n openshift-ingress-operator); do oc patch
${ic} -n openshift-ingress-operator --type=merge -p '{"spec":{"tlsSecurityProfile":
{"type":"Custom","custom":{"ciphers":
["TLS_AES_128_GCM_SHA256","TLS_AES_256_GCM_SHA384","ECDHE-ECDSA-
AES128-GCM-SHA256","ECDHE-RSA-AES128-GCM-SHA256","ECDHE-ECDSA-
AES256-GCM-SHA384","ECDHE-RSA-AES256-GCM-SHA384","ECDHE-ECDSA-
CHACHA20-POLY1305","ECDHE-RSA-CHACHA20-POLY1305","DHE-RSA-AES128-
GCM-SHA256","DHE-RSA-AES256-GCM-
SHA384","TLS_CHACHA20_POLY1305_SHA256","TLS_ECDHE_RSA_WITH_AES_128
_CBC_SHA"],"minTLSVersion":"VersionTLS12"}}}}'; done
```

コマンドを実行すると、次のような応答が返されます。

出力例

```
ingresscontroller.operator.openshift.io/default patched
ingresscontroller.operator.openshift.io/custom1 patched
ingresscontroller.operator.openshift.io/custom2 patched
```

4. CronJob を作成して、TLS 設定が上書きされないようにします。
場合によっては、クラスターの Ingress Controller が再作成されることがあります。このような場合、Ingress Controller は適用された **tlsSecurityProfile** の変更を保持しない可能性があります。これを回避するには、クラスターの Ingress Controller を経由および更新する CronJob を作成します。このプロセスは、Custom Domain Operator を使用するかどうかによって異なります。

   a. Custom Domain Operator を使用しないクラスターの場合:
   Custom Domain Operator を使用しない場合は、次のコマンドを実行して CronJob を作成します。

```
$ cat << EOF | oc apply -f -
apiVersion: batch/v1
kind: CronJob
metadata:
  name: tls-patch
  namespace: openshift-ingress-operator
spec:
  schedule: '@hourly'
  jobTemplate:
    spec:
      template:
        spec:
          containers:
            - name: tls-patch
              image: registry.redhat.io/openshift4/ose-tools-rhel8:latest
              args:
                - /bin/sh
                - '-c'
                - oc patch ingresscontroller/default -n openshift-ingress-operator --type=merge
-p '{"spec":{"tlsSecurityProfile":{"type":"Custom","custom":{"ciphers":
["TLS_AES_128_GCM_SHA256","TLS_AES_256_GCM_SHA384","ECDHE-ECDSA-
AES128-GCM-SHA256","ECDHE-RSA-AES128-GCM-SHA256","ECDHE-ECDSA-
AES256-GCM-SHA384","ECDHE-RSA-AES256-GCM-SHA384","ECDHE-ECDSA-
CHACHA20-POLY1305","ECDHE-RSA-CHACHA20-POLY1305","DHE-RSA-AES128-
GCM-SHA256","DHE-RSA-AES256-GCM-
SHA384","TLS_CHACHA20_POLY1305_SHA256","TLS_ECDHE_RSA_WITH_AES_128
_CBC_SHA"],"minTLSVersion":"VersionTLS12"}}}}'
          restartPolicy: Never
          serviceAccountName: cron-ingress-patch-sa
EOF
```

注記

この CronJob は 1 時間ごとに実行され、必要に応じて Ingress Controller にパッチを適用します。この CronJob は常時実行しないことが重要です。調整がトリガーされ、OpenShift Ingress Operator に過負荷がかかる可能性があるためです。ほとんどの場合、何も変更されないため、CronJob Pod のログは次の例のようになります。

出力例

```
ingresscontroller.operator.openshift.io/default patched (no change)
```

b. Custom Domain Operator を使用するクラスターの場合:
Custom Domain Operator を使用する場合、CronJob は各 Ingress Controller をループ処理して、それぞれにパッチを適用する必要があります。この CronJob を作成するには、次のコマンドを実行します。

```
$ cat << EOF | oc apply -f -
apiVersion: batch/v1
kind: CronJob
metadata:
  name: tls-patch
  namespace: openshift-ingress-operator
spec:
  schedule: '@hourly'
  jobTemplate:
    spec:
      template:
        spec:
          containers:
            - name: tls-patch
              image: registry.redhat.io/openshift4/ose-tools-rhel8:latest
              args:
                - /bin/sh
                - '-c'
                - for ic in $(oc get ingresscontroller -o name -n openshift-ingress-operator); do
oc patch ${ic} -n openshift-ingress-operator --type=merge -p '{"spec":{"tlsSecurityProfile":
{"type":"Custom","custom":{"ciphers":
["TLS_AES_128_GCM_SHA256","TLS_AES_256_GCM_SHA384","ECDHE-ECDSA-
AES128-GCM-SHA256","ECDHE-RSA-AES128-GCM-SHA256","ECDHE-ECDSA-
AES256-GCM-SHA384","ECDHE-RSA-AES256-GCM-SHA384","ECDHE-ECDSA-
CHACHA20-POLY1305","ECDHE-RSA-CHACHA20-POLY1305","DHE-RSA-AES128-
GCM-SHA256","DHE-RSA-AES256-GCM-
SHA384","TLS_CHACHA20_POLY1305_SHA256","TLS_ECDHE_RSA_WITH_AES_128
_CBC_SHA"],"minTLSVersion":"VersionTLS12"}}}}'; done
          restartPolicy: Never
          serviceAccountName: cron-ingress-patch-sa
EOF
```

注記

この CronJob は 1 時間ごとに実行され、必要に応じて Ingress Controller にパッチを適用します。この CronJob は常時実行しないことが重要です。調整がトリガーされ、OpenShift Ingress Operator に過負荷がかかる可能性があるためです。ほとんどの場合、何も変更されないため、CronJob Pod のログは次のようになります。

出力例

```
ingresscontroller.operator.openshift.io/default patched (no change)
ingresscontroller.operator.openshift.io/custom1 patched (no change)
ingresscontroller.operator.openshift.io/custom2 patched (no change)
```

第10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 (ROSA) 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。

10.1. 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。

10.2. 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。

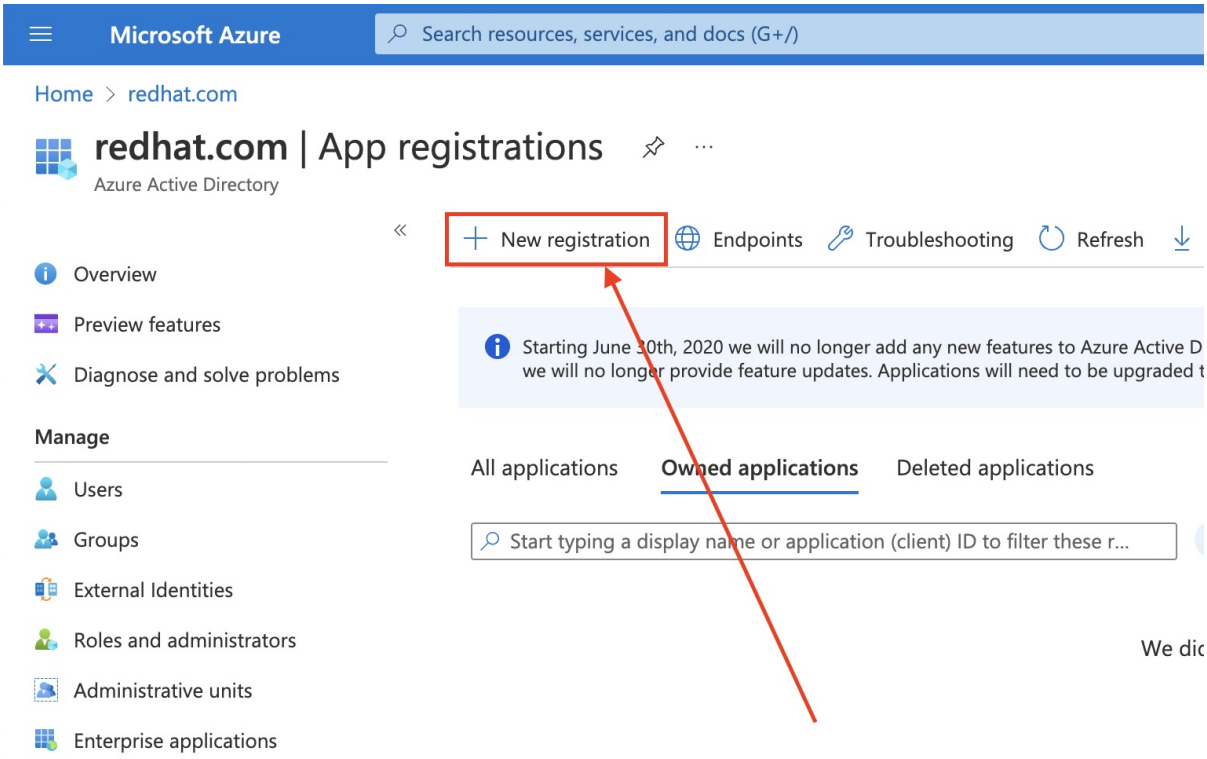
注記

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-openshift.apps.$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。

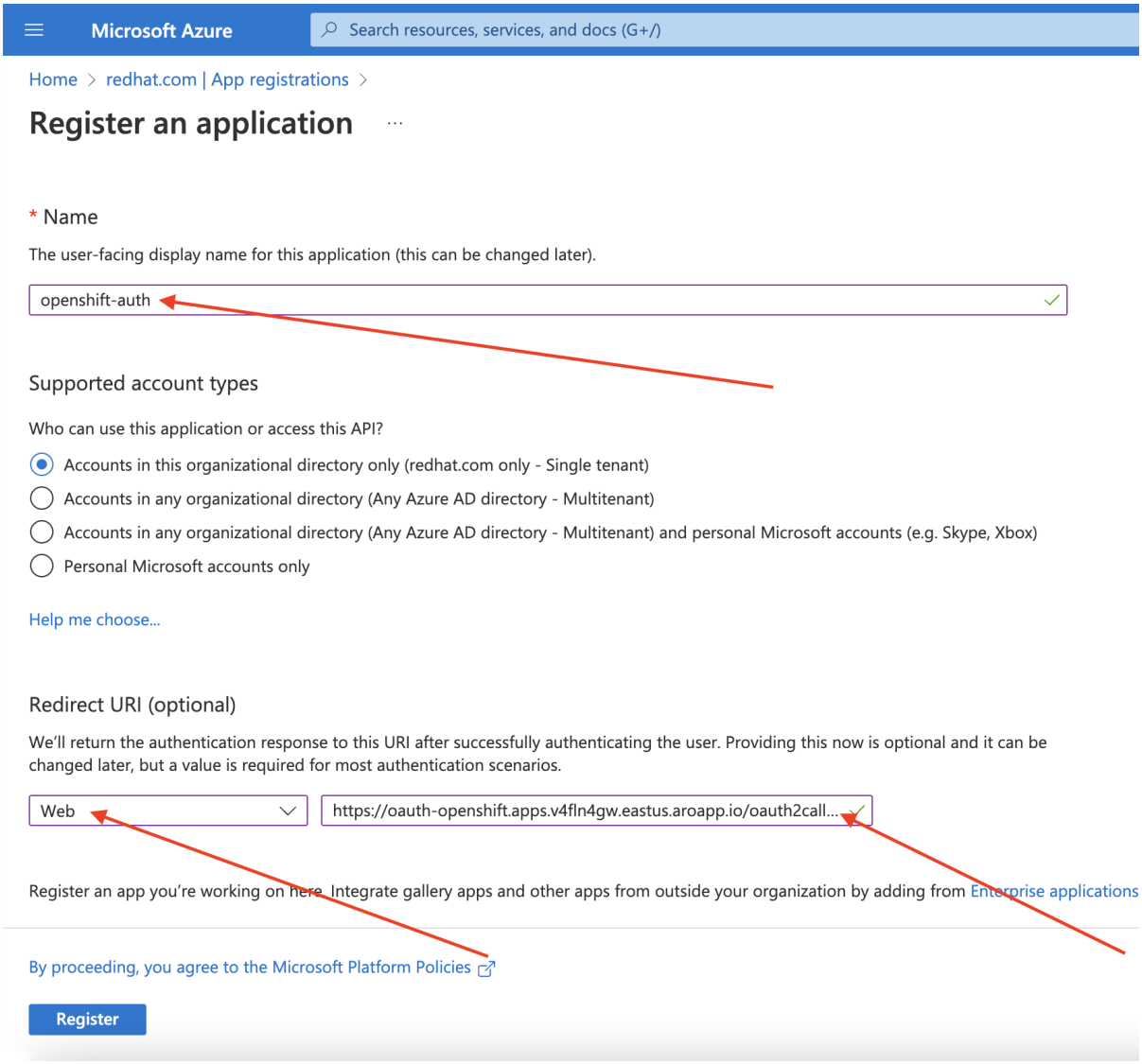
2. Azure portal にログインして Entra ID アプリケーションを作成し、App registrations ブレードを選択します。次に、**New registration** を選択して新しいアプリケーションを作成します。



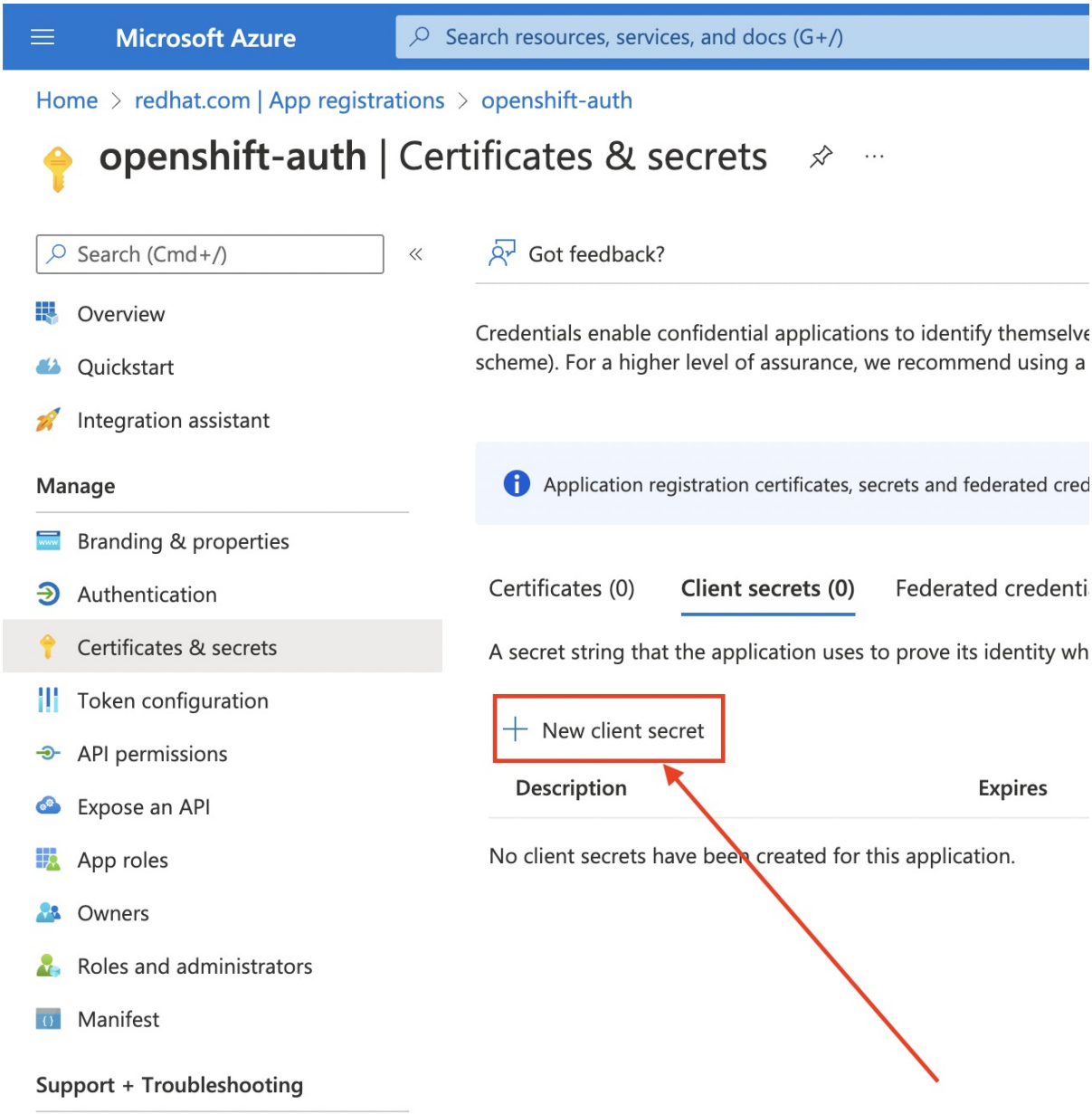
3. アプリケーションに名前を付けます (例: **openshift-auth**)。

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。



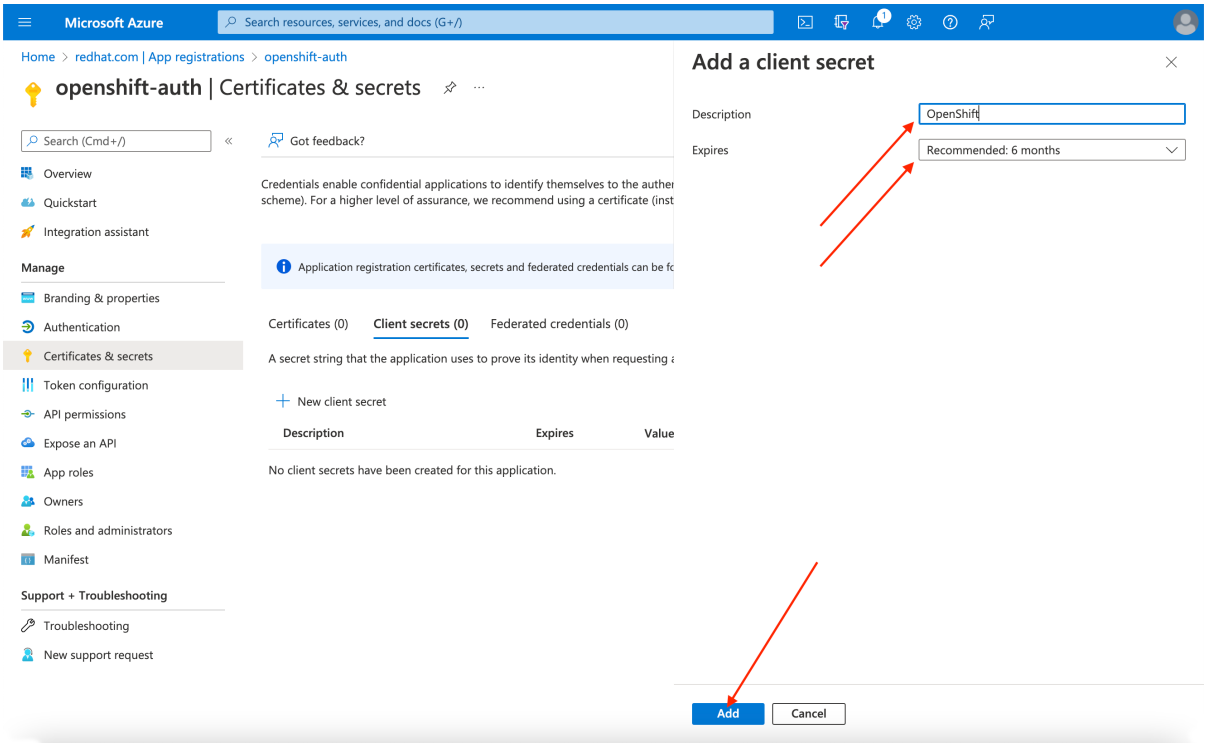
6. **Certificates & secrets** サブブレードを選択し、**New client secret** を選択します。



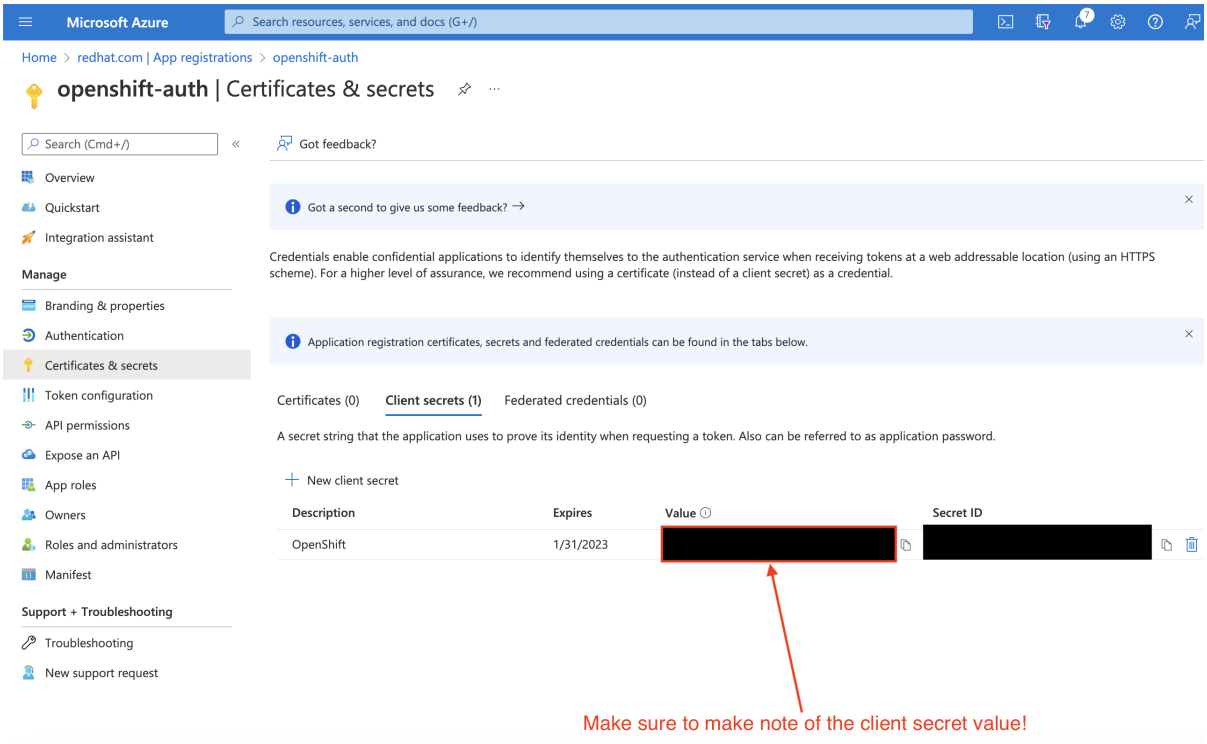
7. 要求された詳細を入力し、生成されたクライアントシークレット値を保存します。このシークレットは、このプロセスで後で必要になります。

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。



10.3. 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。

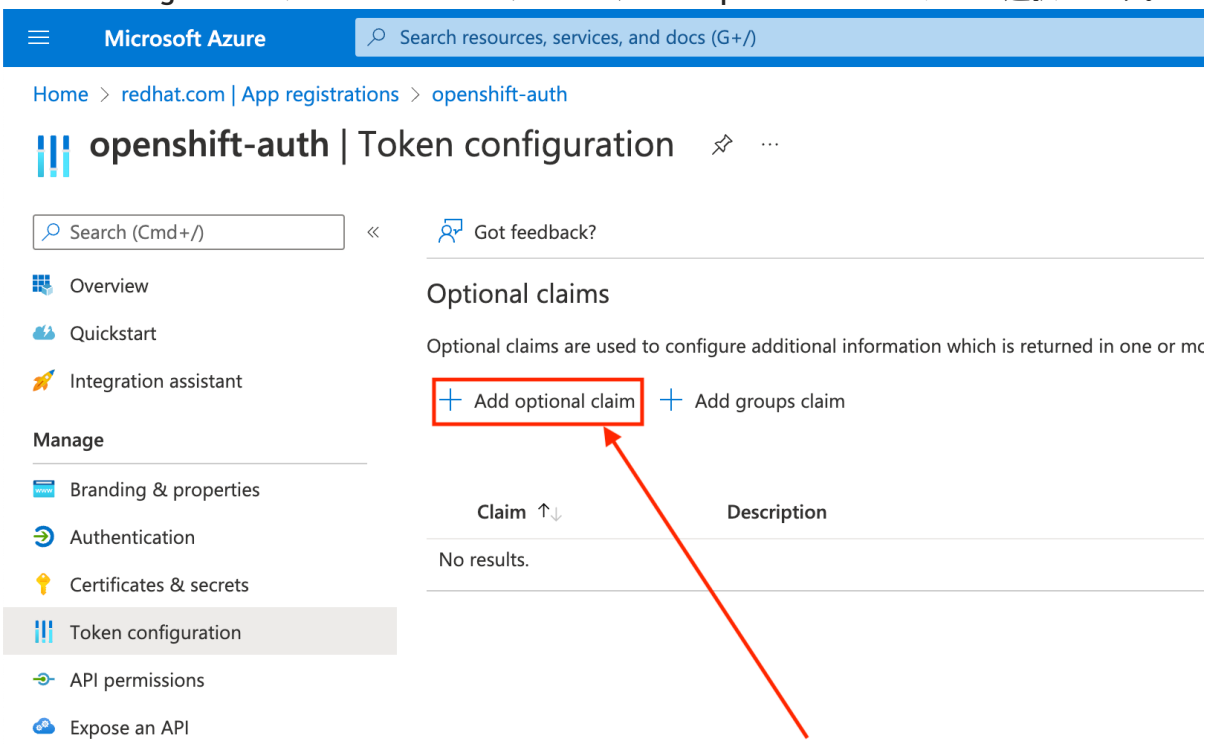
個々のユーザー認証に加えて、Red Hat OpenShift Service on AWS はグループクレーム機能を提供します。この機能により、Entra ID などの OpenID Connect (OIDC) アイデンティティープロバイダーが、Red Hat OpenShift Service on AWS 内で使用するユーザーのグループメンバーシップを提供できるよう

になります。

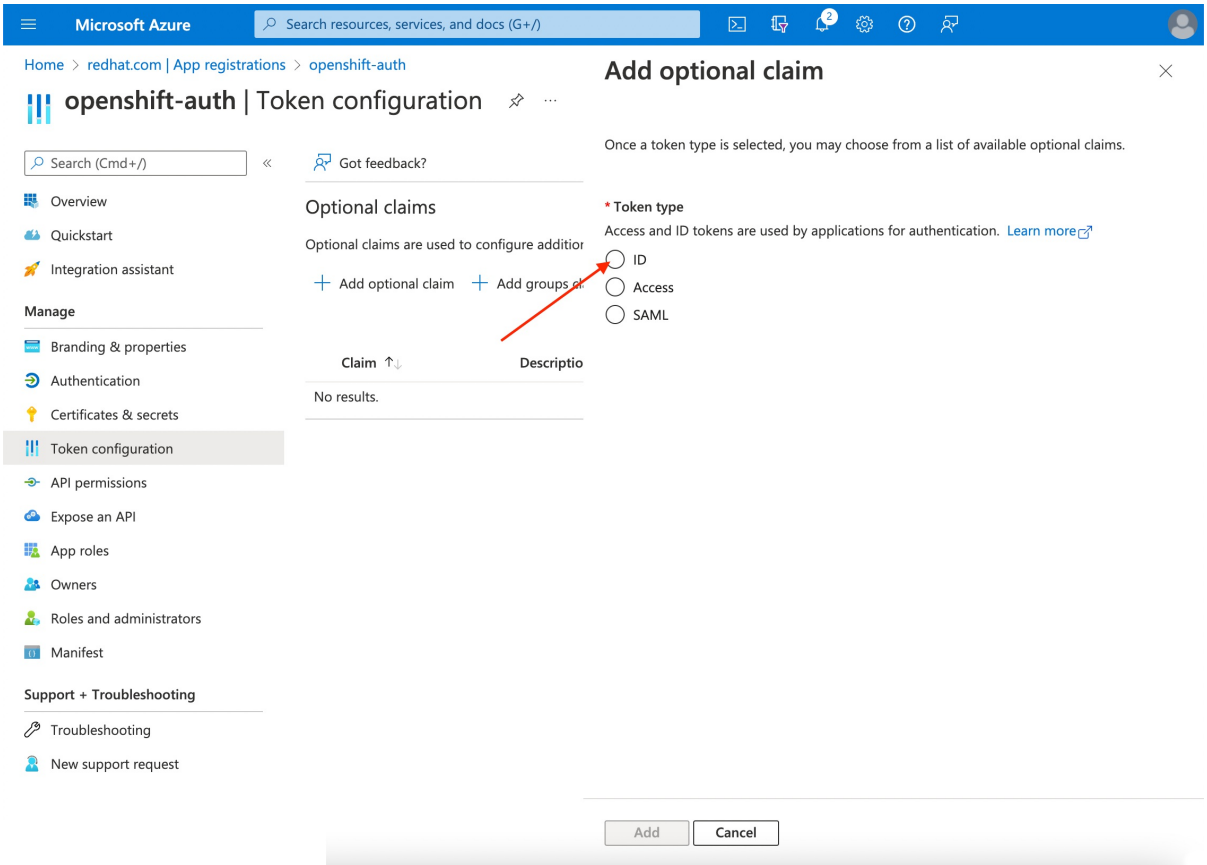
任意のクレームの設定

Entra ID の任意のクレームを設定できます。

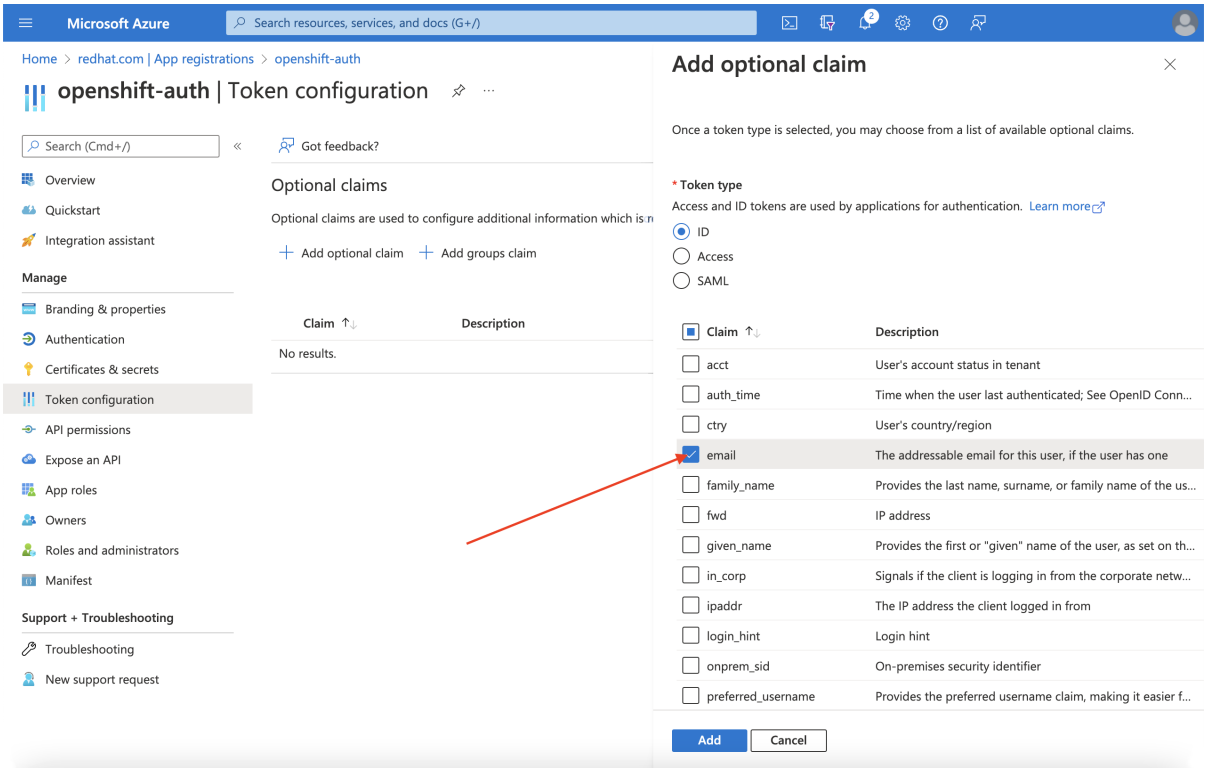
1. **Token configuration** サブブレードをクリックし、**Add optional claim** ボタンを選択します。



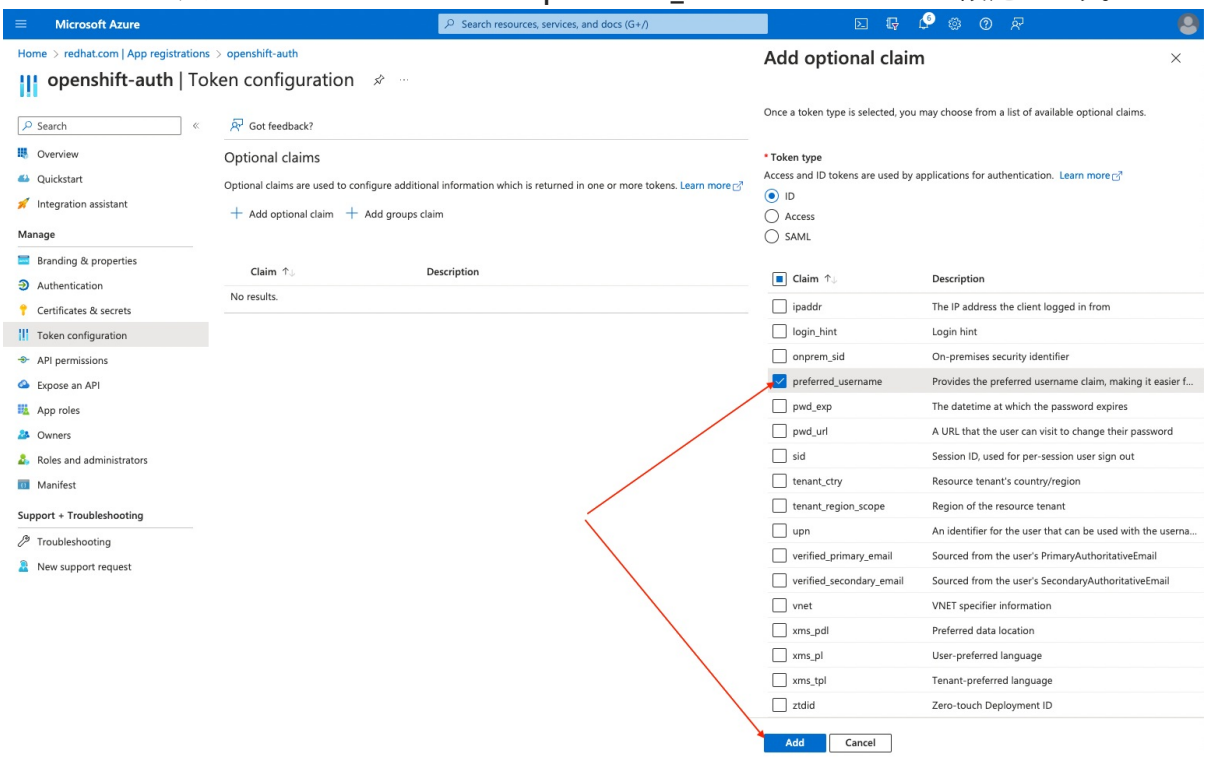
2. **ID** ラジオボタンを選択します。



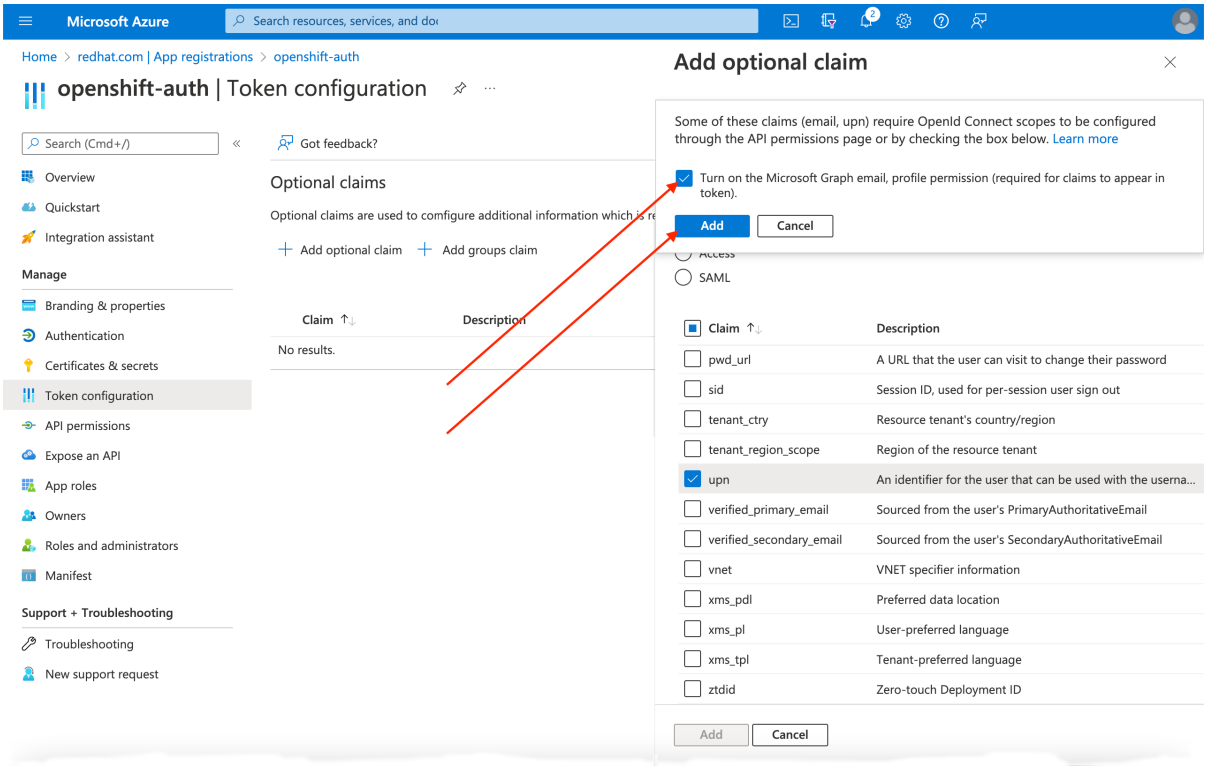
3. **email** クレームのチェックボックスを選択します。



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。

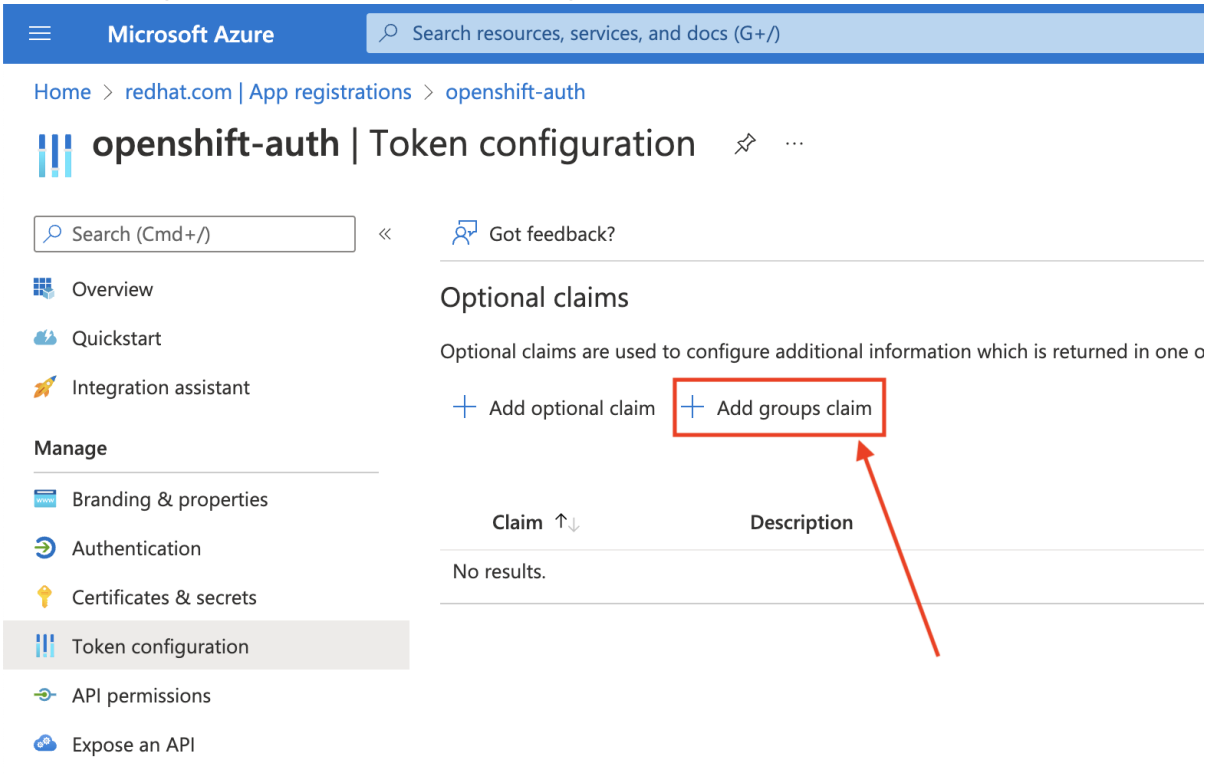


グループクレームの設定 (オプション)

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

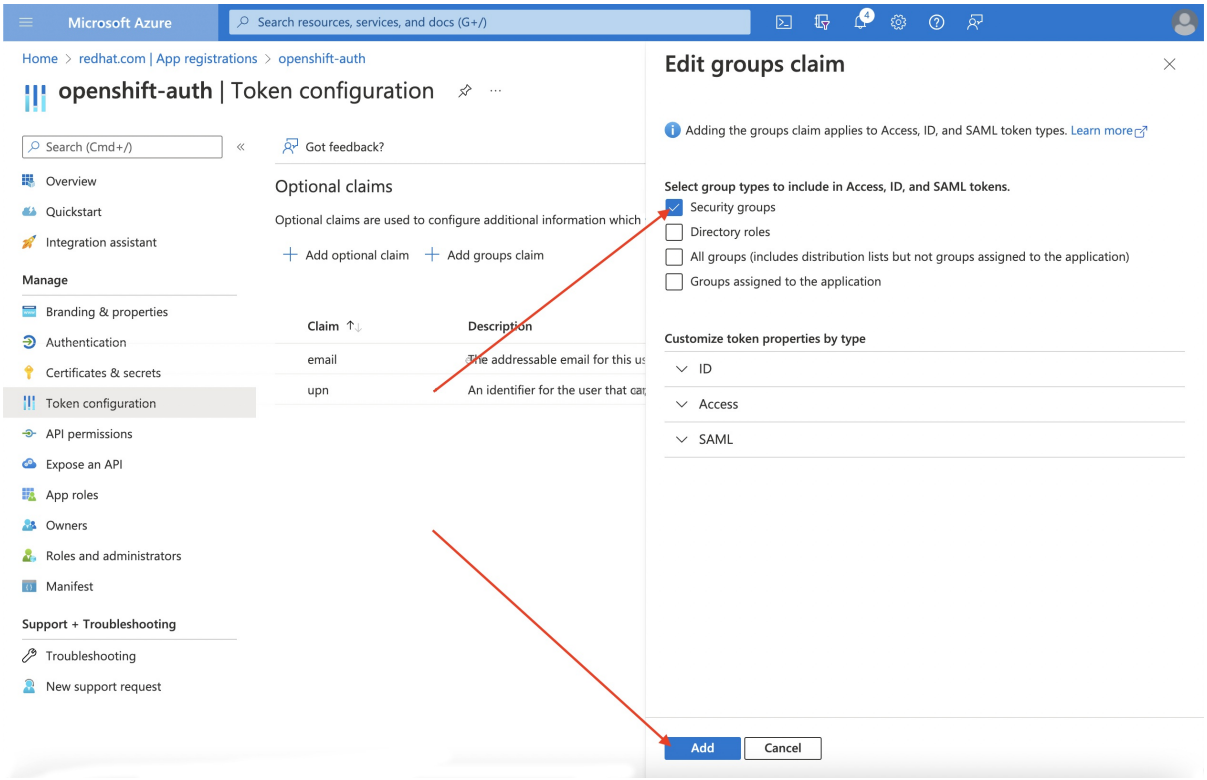
1. **Token configuration** サブブレードで、**Add groups claim** をクリックします。



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。



10.4. 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。

ROSA 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 ROSA 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。

10.5. 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。

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。

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 **RoleBinding** 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。

10.6. 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11章 チュートリアル: STS を使用する ROSA 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 (ROSA) 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。

11.1. 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 とともにデプロイされた ROSA クラスター
- Helm 3
- **aws** CLI
- **oc** CLI
- **jq** CLI

その他の環境要件

1. 次のコマンドを実行して、ROSA 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=""
```

11.2. 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
```

11.3. 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
```

11.4. 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
```

11.5. 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12章 チュートリアル: ROSA 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 (ROSA) 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。

OperatorHub からさまざまな ACK Operator を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 OperatorHub 内の他の ACK Operator に合わせて調整することもできます。

12.1. 前提条件

- ROSA 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**)

12.2. 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。

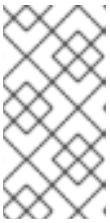
```
$ echo "Cluster: ${ROSA_CLUSTER_NAME}, Region: ${REGION}, OIDC Endpoint:
${OIDC_ENDPOINT}, AWS Account ID: ${AWS_ACCOUNT_ID}"
```

12.3. 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。

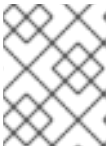
```
$ ROLE_ARN=$(aws iam create-role --role-name "ack-${ACK_SERVICE}-controller" \
   --assume-role-policy-document "file://${SCRATCH}/trust-policy.json" \
   --query Role.Arn --output text)
$ echo $ROLE_ARN

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

12.4. 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=
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. OperatorHub 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
```

12.5. 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
```

12.6. 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
```

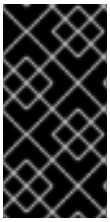
第13章 チュートリアル: ROSA への EXTERNAL DNS OPERATOR のデプロイ

注記

Red Hat OpenShift Service on AWS 4.14 以降、Custom Domain Operator は非推奨になりました。Red Hat OpenShift Service on AWS 4.14 で Ingress を管理するには、Ingress Operator を使用します。Red Hat OpenShift Service on AWS 4.13 以前のバージョンでは機能に変更はありません。

Custom Domain Operator を設定するには、Amazon Route 53 ホストゾーンにワイルドカード CNAME DNS レコードが必要です。ワイルドカードレコードを使用しない場合は、**External DNS** Operator を使用してルートの個別のエントリーを作成できます。

このチュートリアルを使用して、Red Hat OpenShift Service on AWS (ROSA) のカスタムドメインで **External DNS** Operator をデプロイおよび設定します。



重要

External DNS Operator は、IAM Roles for Service Accounts (IRSA) を使用した STS をサポートせず、代わりに有効期間の長い Identity Access Management (IAM) 認証情報を使用します。このチュートリアルは、Operator で STS がサポートされた際に更新される予定です。

13.1. 前提条件

- ROSA クラスター
- **dedicated-admin** 権限を持つユーザーアカウント
- OpenShift CLI (**oc**)
- Amazon Web Services (AWS) CLI (**aws**)
- 一意のドメイン (***.apps.<company_name>.io** など)
- 上記ドメイン用の Amazon Route 53 パブリックホストゾーン

13.2. 環境の設定

1. 次の環境変数を設定します。**CLUSTER_NAME** はクラスターの名前に置き換えます。

```
$ export DOMAIN=apps.<company_name>.io (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_NAME}/external-dns"
$ mkdir -p ${SCRATCH}
```

1 カスタムドメイン。

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: ${CLUSTER_NAME}, Region: ${REGION}, AWS Account ID:
${AWS_ACCOUNT_ID}"
```

13.3. カスタムドメインの設定

ROSA は、**Custom Domain** Operator を使用してセカンダリー Ingress Controller を管理します。カスタムドメインを使用してセカンダリー Ingress Controller をデプロイするには、次の手順を実行します。

前提条件

- 一意のドメイン (***.apps.<company_name>.io** など)
- カスタム SAN またはワイルドカード証明書 (**CN=*.apps.<company_name>.io** など)

手順

1. 新しいプロジェクトを作成します。

```
$ oc new-project external-dns-operator
```

2. 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。

```
$ oc -n external-dns-operator create secret tls external-dns-tls --cert=fullchain.pem --
key=privkey.pem
```

3. 新規の **CustomDomain** カスタムリソース (CR) を作成します。

external-dns-custom-domain.yaml の例

```
apiVersion: managed.openshift.io/v1alpha1
kind: CustomDomain
metadata:
  name: external-dns
spec:
  domain: apps.<company_name>.io 1
  scope: External
  loadBalancerType: NLB
  certificate:
    name: external-dns-tls
    namespace: external-dns-operator
```

1 カスタムドメイン。

4. CR を適用します。

```
$ oc apply -f external-dns-custom-domain.yaml
```

5. カスタムドメイン Ingress Controller がデプロイされ、**Ready** ステータスになっていることを確認します。

```
$ oc get customdomains
```

出力例

```
NAME            ENDPOINT                                                  DOMAIN                  STATUS
external-dns    xxrywp.<company_name>.cluster-01.opln.s1.openshiftapps.com  *.apps.<company_name>.io    Ready
```

13.4. AWS アカウントの準備

1. Amazon Route 53 のパブリックホストゾーン ID を取得します。

```
$ export ZONE_ID=$(aws route53 list-hosted-zones-by-name --output json \
  --dns-name "${DOMAIN}." --query 'HostedZones[0]'.Id --out text | sed 's/\/hostedzone\///')
```

2. **External DNS** Operator がカスタムドメインのパブリックホストゾーン ***のみ*** を更新できるようにする AWS IAM ポリシードキュメントを作成します。

```
$ cat << EOF > "${SCRATCH}/external-dns-policy.json"
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "route53:ChangeResourceRecordSets"
      ],
      "Resource": [
        "arn:aws:route53:::hostedzone/${ZONE_ID}"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "route53:ListHostedZones",
        "route53:ListResourceRecordSets"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
EOF
```

3. AWS IAM ポリシーを作成します。

```
$ export POLICY_ARN=$(aws iam create-policy --policy-name "${CLUSTER_NAME}-
AllowExternalDNSUpdates" \
  --policy-document file://${SCRATCH}/external-dns-policy.json \
  --query 'Policy.Arn' --output text)
```

4. AWS IAM ユーザーを作成します。

```
$ aws iam create-user --user-name "${CLUSTER_NAME}-external-dns-operator"
```

5. ポリシーを割り当てします。

```
$ aws iam attach-user-policy --user-name "${CLUSTER_NAME}-external-dns-operator" --
policy-arn $POLICY_ARN
```

注記

これは将来的には IRSA を使用した STS に変更される予定です。

6. IAM ユーザーの AWS キーを作成します。

```
$ SECRET_ACCESS_KEY=$(aws iam create-access-key --user-name
"${CLUSTER_NAME}-external-dns-operator")
```

7. 静的認証情報を作成します。

```
$ cat << EOF > "${SCRATCH}/credentials"
[default]
aws_access_key_id = $(echo $SECRET_ACCESS_KEY | jq -r '.AccessKey.AccessKeyId')
aws_secret_access_key = $(echo $SECRET_ACCESS_KEY | jq -r
'.AccessKey.SecretAccessKey')
EOF
```

13.5. EXTERNAL DNS OPERATOR のインストール

1. OperatorHub から **External DNS** 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: external-dns-group
  namespace: external-dns-operator
spec:
  targetNamespaces:
  - external-dns-operator
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: external-dns-operator
  namespace: external-dns-operator
spec:
```

```
  channel: stable-v1.1
  installPlanApproval: Automatic
  name: external-dns-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
EOF
```

2. **External DNS** Operator が実行されるまで待ちます。

```
$ oc rollout status deploy external-dns-operator --timeout=300s
```

3. AWS IAM ユーザー認証情報からシークレットを作成します。

```
$ oc -n external-dns-operator create secret generic external-dns \
  --from-file "${SCRATCH}/credentials"
```

4. **ExternalDNS** コントローラーをデプロイします。

```
$ cat << EOF | oc apply -f -
apiVersion: externaldns.olm.openshift.io/v1beta1
kind: ExternalDNS
metadata:
  name: ${DOMAIN}
spec:
  domains:
    - filterType: Include
      matchType: Exact
      name: ${DOMAIN}
  provider:
    aws:
      credentials:
        name: external-dns
    type: AWS
  source:
    openshiftRouteOptions:
      routerName: external-dns
    type: OpenShiftRoute
  zones:
    - ${ZONE_ID}
EOF
```

5. コントローラーが実行されるまで待ちます。

```
$ oc rollout status deploy external-dns-${DOMAIN} --timeout=300s
```

13.6. サンプルアプリケーションのデプロイ

ExternalDNS コントローラーが実行されたら、サンプルアプリケーションをデプロイして、新しいルートの公開時にカスタムドメインが設定され、信頼されることを確認します。

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. 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。

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

4. DNS レコードが ExternalDNS によって自動的に作成されたかどうかを確認します。

注記

レコードが Amazon Route 53 に表示されるまでに数分かかる場合があります。

```
$ aws route53 list-resource-record-sets --hosted-zone-id ${ZONE_ID} \
  --query "ResourceRecordSets[?Type == 'CNAME']" | grep hello-openshift
```

5. オプション: TXT レコードを表示して、レコードが ExternalDNS によって作成されたことを確認することもできます。

```
$ aws route53 list-resource-record-sets --hosted-zone-id ${ZONE_ID} \
  --query "ResourceRecordSets[?Type == 'TXT']" | grep ${DOMAIN}
```

6. ブラウザーでカスタムコンソールドメインに移動すると、OpenShift ログインが表示されます。

```
$ echo console.${DOMAIN}
```

第14章 チュートリアル: ROSA 上の CERT-MANAGER OPERATOR を使用した証明書の動的発行

ワイルドカード証明書は、特定のドメインのファーストレベルサブドメインすべてを 1 つの証明書で保護することで簡素化を実現しますが、ユースケースによっては、ドメインごとに個別の証明書を使用する必要がある場合があります。

cert-manager Operator for Red Hat OpenShift と Let's Encrypt を使用して、カスタムドメインを使用して作成したルートの証明書を動的に発行する方法を説明します。

14.1. 前提条件

- ROSA 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**)
- 一意のドメイン (***.apps.<company_name>.io** など)
- 上記ドメイン用の Amazon Route 53 パブリックホストゾーン

14.2. 環境の設定

1. 次の環境変数を設定します。

```
$ export DOMAIN=apps.<company_name>.io (1)
$ export EMAIL=<youremail@company_name.io> (2)
$ export AWS_PAGER=""
$ export CLUSTER_NAME=$(oc get infrastructure cluster -o=jsonpath="
{.status.infrastructureName}"  | sed 's/-[a-z0-9]\{5\}$//')
$ export OIDC_ENDPOINT=$(oc get authentication.config.openshift.io cluster -o json | jq -r
.spec.serviceAccountIssuer | sed  's|^https://||')
$ 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_NAME}/dynamic-certs"
$ mkdir -p ${SCRATCH}
```

(1) カスタムドメイン。

(2) Let's Encrypt が証明書に関する通知を送信するために使用するメールアドレス。

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

14.3. AWS アカウントの準備

cert-manager が Let's Encrypt (または別の ACME 証明書発行者) に証明書を要求すると、Let's Encrypt のサーバーが **チャレンジ** を使用して、その証明書内のドメイン名がお客様によって制御されていることを検証します。このチュートリアルでは、ドメイン名の TXT レコードに特定の値を入力することで、ドメイン名の DNS を制御していることを証明する DNS-01 チャレンジ を使用します。これはすべて cert-manager によって自動的に行われます。cert-manager 権限によってドメインの Amazon Route 53 パブリックホストゾーンを変更できるように、Pod へのアクセスを許可する信頼関係と特定のポリシー権限を持つ Identity Access Management (IAM) ロールを作成する必要があります。

このチュートリアルで使用するパブリックホストゾーンは、ROSA クラスターと同じ AWS アカウント内にあります。パブリックホストゾーンが別のアカウントにある場合は、クロスアカウントアクセス のためにいくつかの追加手順が必要です。

1. Amazon Route 53 のパブリックホストゾーン ID を取得します。

注記

このコマンドは、**DOMAIN** 環境変数として以前指定したカスタムドメインに一致するパブリックホストゾーンを検索します。Amazon Route 53 パブリックホストゾーンを手動で指定するには、**export ZONE_ID=<zone_ID>** を実行し、**<zone_ID>** を特定の Amazon Route 53 パブリックホストゾーン ID に置き換えます。

```
$ export ZONE_ID=$(aws route53 list-hosted-zones-by-name --output json \
  --dns-name "${DOMAIN}." --query 'HostedZones[0]'.Id --out text | sed 's/\/hostedzone\///')
```

2. 指定したパブリックホストゾーン ***のみ*** を更新できる権限を与える **cert-manager** Operator 用の AWS IAM ポリシードキュメントを作成します。

```
$ cat <<EOF > "${SCRATCH}/cert-manager-policy.json"
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "route53:GetChange",
      "Resource": "arn:aws:route53:::change/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "route53:ChangeResourceRecordSets",
        "route53:ListResourceRecordSets"
      ],
      "Resource": "arn:aws:route53:::hostedzone/${ZONE_ID}"
    },
    {
      "Effect": "Allow",
      "Action": "route53:ListHostedZonesByName",
      "Resource": "*"
    }
  ]
}
EOF
```

3. 前のステップで作成したファイルを使用して IAM ポリシーを作成します。

```
$ POLICY_ARN=$(aws iam create-policy --policy-name "${CLUSTER_NAME}-cert-manager-
policy" \
  --policy-document file://${SCRATCH}/cert-manager-policy.json \
  --query 'Policy.Arn' --output text)
```

4. **cert-manager** 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:cert-manager:cert-manager"
   }
 },
 "Principal": {
   "Federated": "arn:aws:iam::$AWS_ACCOUNT_ID:oidc-provider/${OIDC_ENDPOINT}"
 },
 "Action": "sts:AssumeRoleWithWebIdentity"
 }
 ]
}
EOF
```

5. 前のステップで作成した信頼ポリシーを使用して、**cert-manager** Operator の IAM ロールを作成します。

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

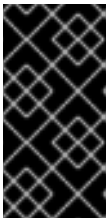
6. 権限ポリシーをロールに割り当てます。

```
$ aws iam attach-role-policy --role-name "${CLUSTER_NAME}-cert-manager-operator" \
  --policy-arn ${POLICY_ARN}
```

14.4. CERT-MANAGER OPERATOR のインストール

1. **cert-manager** 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 cert-manager-operator
```

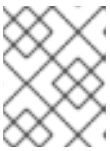


重要

クラスター内で複数の **cert-manager** Operator を使用しないでください。クラスターにコミュニティーの **cert-manager** Operator がインストールされている場合は、それをアンインストールしてから **cert-manager** Operator for Red Hat OpenShift をインストールする必要があります。

2. **cert-manager** Operator for Red Hat OpenShift をインストールします。

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



注記

この Operator がインストールされ、セットアップが完了するまでに数分かかります。

3. **cert-manager** Operator が実行されていることを確認します。

```
$ oc -n cert-manager-operator get pods
```

出力例

```
NAME                                                READY   STATUS    RESTARTS   AGE
cert-manager-operator-controller-manager-84b8799db5-gv8mx   2/2     Running   0          12s
```

4. **cert-manager** Pod によって使用されるサービスアカウントに、前に作成した AWS IAM ロールのアノテーションを付けます。

```
$ oc -n cert-manager annotate serviceaccount cert-manager eks.amazonaws.com/role-arn=${ROLE_ARN}
```

5. 次のコマンドを実行して、既存の **cert-manager** コントローラー Pod を再起動します。

```
$ oc -n cert-manager delete pods -l app.kubernetes.io/name=cert-manager
```

6. DNS-01 チャレンジ解決の問題を防ぐために、外部ネームサーバーを使用するように Operator の設定にパッチを適用します。

```
$ oc patch certmanager.operator.openshift.io/cluster --type merge \
  -p '{"spec":{"controllerConfig":{"overrideArgs":["--dns01-recursive-nameservers-only","--
dns01-recursive-nameservers=1.1.1.1:53"]}}}'
```

7. 次のコマンドを実行して、Let's Encrypt を使用する **ClusterIssuer** リソースを作成します。

```
$ cat << EOF | oc apply -f -
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: letsencrypt-production
spec:
  acme:
    server: https://acme-v02.api.letsencrypt.org/directory
    email: ${EMAIL}
    # This key doesn't exist, cert-manager creates it
    privateKeySecretRef:
      name: prod-letsencrypt-issuer-account-key
    solvers:
    - dns01:
        route53:
         hostedZoneID: ${ZONE_ID}
         region: ${REGION}
         secretAccessKeySecretRef:
           name: ''
EOF
```

8. **ClusterIssuer** リソースの準備ができていることを確認します。

```
$ oc get clusterissuer.cert-manager.io/letsencrypt-production
```

出力例

```
NAME                    READY   AGE
letsencrypt-production   True    47s
```

14.5. カスタムドメイン INGRESS CONTROLLER の作成

1. 新しいプロジェクトを作成します。

```
$ oc new-project custom-domain-ingress
```

2. 証明書リソースを作成して設定し、カスタムドメイン Ingress Controller の証明書をプロビジョニングします。

注記

次の例では、単一のドメイン証明書を使用します。SAN およびワイルドカード証明書もサポートされています。

```
$ cat << EOF | oc apply -f -
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: custom-domain-ingress-cert
  namespace: custom-domain-ingress
spec:
  secretName: custom-domain-ingress-cert-tls
  issuerRef:
     name: letsencrypt-production
     kind: ClusterIssuer
  commonName: "${DOMAIN}"
  dnsNames:
  - "${DOMAIN}"
EOF
```

3. 証明書が発行されたことを確認します。

注記

Let's Encrypt によってこの証明書が発行されるまでに数分かかります。5 分以上かかる場合は、**oc -n custom-domain-ingress describe certificate.cert-manager.io/custom-domain-ingress-cert** を実行して、cert-manager によって報告される問題を確認します。

```
$ oc -n custom-domain-ingress get certificate.cert-manager.io/custom-domain-ingress-cert
```

出力例

```
NAME                        READY   SECRET                          AGE
custom-domain-ingress-cert   True    custom-domain-ingress-cert-tls   9m53s
```

4. 新規の **CustomDomain** カスタムリソース (CR) を作成します。

```
$ cat << EOF | oc apply -f -
apiVersion: managed.openshift.io/v1alpha1
kind: CustomDomain
metadata:
  name: custom-domain-ingress
spec:
  domain: ${DOMAIN}
  scope: External
  loadBalancerType: NLB
  certificate:
    name: custom-domain-ingress-cert-tls
    namespace: custom-domain-ingress
EOF
```

5. カスタムドメイン Ingress Controller がデプロイされ、**Ready** ステータスになっていることを確認します。

```
$ oc get customdomains
```

出力例

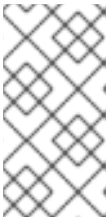
```
NAME                    ENDPOINT                                                DOMAIN
STATUS
custom-domain-ingress   tfoxdx.custom-domain-ingress.cluster.1234.p1.openshiftapps.com
example.com       Ready
```

6. カスタムドメイン Ingress Controller の DNS 解決を有効にするために必要な DNS 変更を含むドキュメントを準備します。

```
$ INGRESS=$(oc get customdomain.managed.openshift.io/custom-domain-ingress --
template={{.status.endpoint}})
$ cat << EOF > "${SCRATCH}/create-cname.json"
{
  "Comment":"Add CNAME to custom domain endpoint",
  "Changes":[{
      "Action":"CREATE",
      "ResourceRecordSet":{
        "Name": "*.${DOMAIN}",
      "Type":"CNAME",
      "TTL":30,
      "ResourceRecords":[{
        "Value": "${INGRESS}"
      }]
    }
  }]
}
EOF
```

7. 変更を伝播するために Amazon Route 53 に送信します。

```
$ aws route53 change-resource-record-sets \
  --hosted-zone-id ${ZONE_ID} \
  --change-batch file://${SCRATCH}/create-cname.json
```



注記

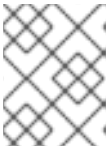
ワイルドカード CNAME レコードを使用すると、カスタムドメイン Ingress Controller を使用してデプロイする新しいアプリケーションごとに新しいレコードを作成する必要がなくなりますが、これらの各アプリケーションが使用する証明書は、ワイルドカード証明書 **ではありません**。

14.6. カスタムドメインルートの動的証明書の設定

これで、指定したドメインのファーストレベルサブドメインでクラスターアプリケーションを公開できるようになります。しかし、アプリケーションのドメインと一致する TLS 証明書で接続が保護されません。このクラスターアプリケーションに各ドメイン名の有効な証明書を確実に提供するために、この

ドメインの配下に作成されたすべての新しいルートに証明書を動的に発行するように cert-manager を設定します。

1. cert-manager による OpenShift ルートの証明書の管理に必要な OpenShift リソースを作成します。
このステップでは、クラスター内のアノテーション付きルートを特別に監視する新しいデプロイメント (とその Pod) を作成します。新しいルートで **issuer-kind** および **issuer-name** アノテーションが検出された場合、ルートの作成時に指定されたホスト名を受け入れる、このルートに固有の新しい証明書を発行者 (この場合は ClusterIssuer) に要求します。



注記

クラスターから GitHub にアクセスできない場合は、raw コンテンツをローカルに保存し、**oc apply -f localfilename.yaml -n cert-manager** を実行します。

```
$ oc -n cert-manager apply -f https://github.com/cert-manager/openshift-
routes/releases/latest/download/cert-manager-openshift-routes.yaml
```

このステップでは、次の追加の OpenShift リソースも作成されます。

- **ClusterRole** - クラスター全体のルートを監視および更新する権限を付与します。
- **ServiceAccount** - 新しく作成された Pod を実行する権限を使用します。
- **ClusterRoleBinding** - これら 2 つのリソースをバインドします。

2. 新しい **cert-manager-openshift-routes** Pod が正常に実行されていることを確認します。

```
$ oc -n cert-manager get pods
```

結果の例:

```
NAME                                     READY   STATUS    RESTARTS   AGE
cert-manager-866d8f788c-9kspc            1/1     Running   0          4h21m
cert-manager-cainjector-6885c585bd-znws8 1/1     Running   0          4h41m
cert-manager-openshift-routes-75b6bb44cd-f8kd5   1/1     Running   0          6s
cert-manager-webhook-8498785dd9-bvfdf    1/1     Running   0          4h41m
```

14.7. サンプルアプリケーションのデプロイ

動的証明書を設定したら、サンプルアプリケーションをデプロイして、新しいルートの公開時に証明書がプロビジョニングされ、信頼されることを確認します。

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

3. クラスターの外部からアプリケーションを公開するためのルートを作成します。

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

4. ルートの証明書が信頼されていないことを確認します。

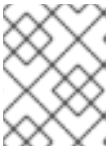
```
$ curl -I https://hello.${DOMAIN}
```

出力例

```
curl: (60) SSL: no alternative certificate subject name matches target host name
'hello.example.com'
More details here: https://curl.se/docs/sslcerts.html

curl failed to verify the legitimacy of the server and therefore could not
establish a secure connection to it. To learn more about this situation and
how to fix it, please visit the web page mentioned above.
```

5. ルートにアノテーションを付けて、cert-manager をトリガーしてカスタムドメインの証明書をプロビジョニングします。

```
$ oc -n hello-world annotate route hello-openshift-tls cert-manager.io/issuer-
kind=ClusterIssuer cert-manager.io/issuer-name=letsencrypt-production
```



注記

証明書の作成には 2 - 3 分かかります。証明書の更新は、有効期限が近づくと、**cert-manager** Operator によって自動的に処理されます。

6. ルートの証明書が信頼されていることを確認します。

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

出力例

```
HTTP/2 200
date: Thu, 05 Oct 2023 23:45:33 GMT
content-length: 17
content-type: text/plain; charset=utf-8
set-cookie: 52e4465485b6fb4f8a1b1bed128d0f3b=68676068bb32d24f0f558f094ed8e4d7;
path=/; HttpOnly; Secure; SameSite=None
cache-control: private
```

14.8. 動的証明書のプロビジョニングのトラブルシューティング

注記

証明書の作成中、検証プロセスが完了するまでに通常 2 - 3 分かかります。

ルートにアノテーションを付けても、証明書作成ステップ中に証明書の作成がトリガーされない場合は、**certificate**、**certificaterequest**、**order**、および **Challenge** リソースのそれぞれに対して **oc description** を実行して、問題の原因の特定に役立つイベントや理由を表示します。

```
$ oc get certificate,certificaterequest,order,challenge
```

トラブルシューティングについては、証明書のデバッグに役立つこちらのガイド を参照してください。

cmctl CLI ツールを使用して、証明書のステータスの確認や更新のテストなど、さまざまな証明書管理アクティビティーを行うこともできます。

第15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

IP ベースの設定を必要とするセキュリティーグループやその他の種類のセキュリティー制御などの項目を設定する場合、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望ましい場合があります。デフォルトでは、Red Hat OpenShift Service on AWS (ROSA) (OVN-Kubernetes CNI を使用) は、プールからランダムな IP アドレスを割り当てるため、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、必要以上にオープンな状態になったりします。このガイドでは、一般的なセキュリティー標準やガイダンス、その他の潜在的なユースケースに対応するために、Egress クラスタートラフィック用の予測可能な IP アドレスのセットを設定する方法を説明します。

詳細は、このトピックに関する OpenShift ドキュメント を参照してください。

15.1. 前提条件

- OVN-Kubernetes とともにデプロイされた ROSA クラスター
- OpenShift CLI (**oc**)
- ROSA CLI (**rosa**)
- **jq**

15.1.1. 環境

以下を実行すると、このチュートリアルの環境変数が設定されるため、独自の環境変数をコピー/ペーストする必要がなくなります。別のマシンプールをターゲットにする場合は、必ず **ROSA_MACHINE_POOL_NAME** 変数を置き換えてください。

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

15.2. 容量の確保

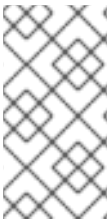
各パブリッククラウドプロバイダーには、ノードごとに割り当てられる IP アドレスの数に制限があります。これは、Egress IP アドレスを割り当てる機能に影響を与える可能性があります。容量が十分であることを確認するには、次のコマンドを実行すると、現在割り当てられている IP アドレスと総容量を出力し、影響を受ける可能性のあるノードを特定できます。

```
$ 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
}
---
```



注記

上記の例では、使いやすい **jq** をフィルターとして使用しています。**jq** がインストールされていない場合は、各ノードの **metadata.annotations['cloud.network.openshift.io/egress-ipconfig']** フィールドを手動で確認すると、ノードの容量を確認できます。

15.3. EGRESS IP ルールの作成

15.3.1. Egress IP の特定

ルールを作成する前に、どの Egress IP を使用するかを特定する必要があります。選択する Egress IP はワーカーノードのプロビジョニング先のサブネットに属するものである必要がある点に注意してください。

15.3.2. Egress IP の予約

AWS VPC DHCP サービスとの競合を避けるために、要求した Egress IP を予約することを推奨します。ただし、これは必須ではありません。これを行う場合は、CIDR 予約に関する AWS ドキュメントの手順を実行 すると、明示的な IP 予約を要求できます。

15.4. NAMESPACE への EGRESS IP のデプロイ

namespace の選択に基づいて Egress IP アドレスを割り当てる方法のデモを行うために、プロジェクトを作成します。

```
$ oc create -f demo-egress-pod.yaml
```

Egress ルールを作成します。このルールにより、先ほど **spec.namespaceSelector** フィールドを使用して作成した namespace 内のすべての Pod に Egress トラフィックが適用されます。

```
$ 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
```

15.4.1. Pod への Egress IP の割り当て

Pod の選択に基づいて Egress IP アドレスを割り当てる方法のデモを行うために、プロジェクトを作成します。

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

Egress ルールを作成します。このルールにより、先ほど **spec.podSelector** フィールドを使用して作成した Pod に Egress トラフィックが適用されます。**spec.namespaceSelector** は必須フィールドであることに注意してください。

```
$ 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
```

15.4.2. ノードへのラベルの適用

oc get egressips を実行すると、Egress IP の割り当てが保留中であることが確認できます。

出力例

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

Egress IP の割り当てを完了するには、ノードに特定のラベルを割り当てる必要があります。前のス

テップで作成した 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。環境変数の設定ステップで環境変数を使用して設定したときのように、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する必要があります。

次のコマンドを使用して、必要なラベルをマシンプールに割り当てます。

警告

マシンプールのノードラベルを使用している場合は、このコマンドを実行すると、そ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="
```

15.4.3. Egress IP の確認

oc get egressips を実行すると、Egress IP の割り当てを確認できます。これにより、次のような Egress が生成されます。

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
```

15.5. EGRESS IP ルールのテスト

15.5.1. サンプルアプリケーションのデプロイ

ルールをテストするために、指定の Egress IP アドレスのみに制限されたサービスを作成します。このサービスは、ごく一部の IP アドレスを受け付ける外部サービスをシミュレートしたものです。

有用な情報を提供する echoserver を実行します。

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

Pod をサービスとして公開し、(**.spec.loadBalancerSourceRanges** フィールドを使用して) サービスへの Ingress を、Pod で使用するように指定した Egress IP アドレスのみに制限します。

```
$ 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
```

ロードバランサーのホスト名を **LOAD_BALANCER_HOSTNAME** 環境変数として取得します。これをコピーすることで、次の手順で使用できます。

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

15.5.2. namespace の Egress のテスト

以前に作成した namespace Egress ルールをテストします。以下を実行すると、デモサービスに対して curl を実行できる対話型シェルが起動します。

```
$ 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
```

Pod 内に入ったら、ロードバランサーに要求を送信して、正常に接続できることを確認します。

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

接続が成功したことを示す、次のような出力が表示されるはずです。以下の **client_address** は、Egress IP ではなく、ロードバランサーの内部 IP アドレスであることに注意してください。サービスを **.spec.loadBalancerSourceRanges** に制限した上で接続が成功すれば、デモは成功です。

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-
```

完了したら、Pod を終了しても問題ありません。

```
$ exit
```

15.5.3. Pod の Egress のテスト

以前に作成した Pod Egress ルールをテストします。以下を実行すると、デモサービスに対して curl を実行できる対話型シェルが起動します。

```
$ 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
```

Pod 内に入ったら、ロードバランサーに要求を送信して、正常に接続できることを確認します。

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

接続が成功したことを示す、次のような出力が表示されるはずです。以下の **client_address** は、Egress IP ではなく、ロードバランサーの内部 IP アドレスであることに注意してください。サービスを **.spec.loadBalancerSourceRanges** に制限した上で接続が成功すれば、デモは成功です。

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-
```

完了したら、Pod を終了しても問題ありません。

```
$ exit
```

15.5.4. ブロックされた Egress のテスト

接続の成功の代わりに、Egress ルールが適用されない場合にトラフィックが正しくブロックされることを確認することもできます。サービスを **.spec.loadBalancerSourceRanges** に制限した上で接続が失敗すれば、デモは成功です。

```
$ 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
```

Pod 内に入ったら、ロードバランサーに要求を送信できます。

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

上記のコマンドはハングするはずです。完了したら、Pod を終了しても問題ありません。

```
$ exit
```

15.6. クリーンアップ

次のコマンドを実行してクラスターをクリーンアップできます。

```
$ 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
```

次のコマンドを実行すると、割り当てられたノードラベルをクリーンアップできます。

警告

マシンプールのノードラベルを使用している場合は、このコマンドを実行すると、そ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 ""
```

第16章 ROSA のスタートガイド

16.1. チュートリアル: ROSA とは

Red Hat OpenShift Service on AWS (ROSA) は、フルマネージドのターンキーアプリケーションプラットフォームです。ROSA を利用すると、アプリケーションを構築およびデプロイして顧客に価値を提供するという、最も重要なことに集中できます。Red Hat と AWS の SRE エキスパートが基盤となるプラットフォームを管理するため、お客様がインフラストラクチャーの管理について心配する必要はありません。ROSA は、幅広い AWS コンピュート、データベース、分析、機械学習、ネットワーク、モバイル、およびその他のサービスとのシームレスな統合を提供し、差別化されたエクスペリエンスの構築とお客様への提供をさらに加速します。

ROSA は、AWS Security Token Service (STS) を利用して、AWS アカウントのインフラストラクチャーを管理するための認証情報を取得します。AWS STS は、IAM ユーザーまたはフェデレーションユーザーの一時的な認証情報を作成するグローバル Web サービスです。ROSA はこれを使用して、権限が限られた短期間のセキュリティー認証情報を割り当てます。これらの認証情報は、AWS API 呼び出しを行う各コンポーネントに固有の IAM ロールに関連付けられます。この方法は、クラウドサービスのリソース管理における最小権限と安全なプラクティスの原則に沿ったものです。ROSA コマンドラインインターフェイス (CLI) ツールは、固有のタスクに割り当てられた STS 認証情報を管理し、OpenShift 機能の一部として AWS リソースに対してアクションを実行します。

16.1.1. ROSA の主な特徴

- **ネイティブ AWS サービス:** AWS マネジメントコンソールから、セルフサービスのオンボーディングにより、オンデマンドで Red Hat OpenShift にアクセスし、使用できます。
- **柔軟な従量制の価格設定:** ビジネスニーズに合わせたスケールアップが可能で、柔軟な価格設定と時間単位または年単位のオンデマンド請求モデルに基づく従量課金制が導入されています。
- **Red Hat OpenShift と AWS の使用に対する一括請求書:** お客様は、Red Hat OpenShift と AWS の両方の使用に対して、AWS から単一の請求書を受け取ります。
- **完全に統合されたサポートエクスペリエンス:** インストール、管理、メンテナンス、アップグレードは、Red Hat と Amazon の共同サポートと 99.95 % のサービスレベルアグリーメント (SLA) に基づき、Red Hat サイトリライアビリティーエンジニア (SRE) が行います。
- **AWS サービスインテグレーション:** AWS には、コンピューティング、ストレージ、ネットワーキング、データベース、分析、機械学習など、堅牢なクラウドサービスポートフォリオがあります。これらのサービスはすべて、ROSA を通じて直接アクセスできます。これにより、使い慣れた管理インターフェイスを通じて、グローバルかつオンデマンドでの構築、運用、拡張が容易になります。
- **可用性の最大化:** サポート対象リージョン内の複数のアベイラビリティーゾーンにクラスターをデプロイして可用性を最大化し、最も要求の厳しいミッションクリティカルなアプリケーションとデータの高可用性を維持します。
- **クラスターノードのスケーリング:** リソースのニーズに合わせてコンピューティングノードを簡単に追加または削除できます。
- **最適化されたクラスター:** ニーズを満たすサイズのクラスターを備えた EC2 インスタンスタイプ (メモリー最適化型、コンピュート最適型、汎用型) を選択できます。
- **グローバルな可用性:** ROSA が使用できる世界の地域を確認するには、製品の地域別可用性ページ を参照してください。

16.1.2. ROSA と Kubernetes

ROSA には、コンテナー管理、Operator、ネットワーキング、負荷分散、サービスメッシュ、CI/CD、ファイアウォール、モニタリング、レジストリー、認証、および認可機能など、コンテナーのデプロイと管理に必要なすべての機能がバンドルされています。これらのコンポーネントは、完全なプラットフォームとして統合された操作を行うために一緒にテストされます。無線によるプラットフォームアップグレードを含む自動化されたクラスター操作により、Kubernetes エクスペリエンスがさらに強化されています。

16.1.3. 基本的な責任

一般的に、クラスターのデプロイメントと維持は Red Hat または AWS がその責任を果たし、アプリケーション、ユーザー、データについてはお客様が責任を果たします。責任の詳細な内訳については、責任マトリクス を参照してください。

16.1.4. ロードマップと機能リクエスト

現在開発中の機能の最新ステータスは、ROSA ロードマップ で確認してください。製品チームに提案がある場合は、新規 Issue を作成してください。

16.1.5. リージョン別の AWS 可用性

最新の ROSA 可用性については、リージョン別の製品可用性 ページを参照してください。

16.1.6. コンプライアンス認証

ROSA は現在、SOC-2 タイプ 2、SOC 3、ISO-27001、ISO 27017、ISO 27018、HIPAA、GDPR、および PCI-DSS に準拠しています。現在、FedRAMP High に向けて取り組んでいます。

16.1.7. ノード

16.1.7.1. 複数の AWS リージョンにまたがるワーカーノード

ROSA クラスター内のすべてのノードは、同じ AWS リージョンに配置する必要があります。複数のアベイラビリティーゾーン用に設定されたクラスターの場合、コントロールプレーンノードとワーカーノードはアベイラビリティーゾーンをまたいで分散的に配置されます。

16.1.7.2. ワーカーノードの最小数

ROSA クラスターにおけるワーカーノードの最小数は、アベイラビリティーゾーンが 1 つの場合は 2 台、アベイラビリティーゾーンが複数の場合は 3 台です。

16.1.7.3. 基盤となるノードのオペレーティングシステム

すべての OpenShift v4.x 製品と同様に、コントロールプレーン、インフラおよびワーカーノードは Red Hat Enterprise Linux CoreOS (RHCOS) を実行します。

16.1.7.4. ノードのハイバネートまたはシャットダウン

現時点では、ROSA にノードのハイバネート機能またはシャットダウン機能はありません。シャットダウンおよびハイバネート機能は OpenShift プラットフォームの機能ですが、広範なクラウドサービスで使用できるほど十分には成熟していません。

16.1.7.5. ワーカーノードでサポートされるインスタンス

ワーカーノードでサポートされるインスタンスの完全なリストは、AWS インスタンスタイプ を参照してください。スポットインスタンスもサポートされています。

16.1.7.6. ノードの自動スケーリング

自動スケーリングを使用すると、現在のワークロードに基づいてクラスターのサイズを自動的に調整できます。詳細は、クラスター上のノードの自動スケーリング を参照してください。

16.1.7.7. ワーカーノードの最大数

ワーカーノードの最大数は、ROSA クラスターごとに 180 台です。ノード数の詳細は、制限とスケーラビリティ を参照してください。

アカウント全体およびクラスターごとのロールのリストは、ROSA のドキュメント に記載されています。

16.1.8. 管理者

お客様側の ROSA 管理者は、ユーザーが作成したすべてのプロジェクトにアクセスできる他に、ユーザーとクォータを管理できます。

16.1.9. OpenShift のバージョンとアップグレード

ROSA は、OpenShift Container Platform をベースとするマネージドサービスです。現行バージョンとライフサイクルの日付は、ROSA のドキュメント で確認できます。

お客様は、OpenShift の最新バージョンにアップグレードして、そのバージョンの OpenShift 機能を使用できます。詳細は、ライフサイクルの日付 を参照してください。ROSA では、一部の OpenShift 機能を使用できません。詳細は、サービス定義 を参照してください。

16.1.10. サポート

Red Hat OpenShift Cluster Manager から直接チケットを作成できます。サポートを受ける方法について、詳しくは ROSA サポートドキュメント を参照してください。

Red Hat カスタマーポータル にアクセスして、Red Hat ナレッジベースで Red Hat 製品に関連する記事やソリューションを検索または参照したり、Red Hat サポートに対してサポートケースを作成したりすることもできます。

16.1.10.1. Limited support

ROSA クラスターが "サポート終了日" までにアップグレードされなかった場合、クラスターは限定サポートステータスで引き続き動作します。そのクラスターの SLA は適用されなくなりますが、そのクラスターのサポートはその後も受けることができます。詳細は、限定サポートステータス のドキュメントを参照してください。

その他のサポートリソース

- Red Hat サポート
- AWS Support
  お客様が AWS サポートを使用するためには、有効な AWS サポート契約が必要です。

16.1.11. Service Level Agreement (SLA)

詳細は、ROSA SLA のページを参照してください。

16.1.12. 通知と連絡

Red Hat は、Red Hat および AWS の新機能、更新、定期メンテナンスに関する通知を、メールおよび Red Hat コンソールサービスログを通じて提供します。

16.1.13. Open Service Broker for AWS (OBSA)

ROSA では OSBA を使用できます。ただし、より新しい AWS Controller for Kubernetes の使用が推奨されます。OSBA の詳細は、Open Service Broker for AWS を参照してください。

16.1.14. オフボーディング

お客様はいつでも ROSA の使用を停止し、アプリケーションをオンプレミス、プライベートクラウド、または他のクラウドプロバイダーに移行できます。標準予約インスタンス (RI) ポリシーは、未使用の RI に適用されます。

16.1.15. 認証

ROSA がサポートする認証メカニズムは、OpenID Connect (OAuth2 のプロファイル)、Google OAuth、GitHub OAuth、GitLab、および LDAP です。

16.1.16. SRE クラスターアクセス

すべての SRE クラスターへのアクセスは、MFA で保護されます。詳細は、SRE アクセス を参照してください。

16.1.17. 暗号化

16.1.17.1. 暗号鍵

ROSA は、KMS に保存されている鍵を使用して EBS ボリュームを暗号化します。オプションとして、お客様はクラスター作成時に独自の KMS キーを指定することもできます。

16.1.17.2. KMS キー

KMS キーを指定すると、コントロールプレーン、インフラストラクチャーおよびワーカーノードのルートボリュームと永続ボリュームがそのキーで暗号化されます。

16.1.17.3. データの暗号化

デフォルトでは、保存時に暗号化されます。AWS Storage プラットフォームは、データを永続化する前に自動的に暗号化し、取得する前にデータを復号化します。詳細は、AWS EBS Encryption を参照してください。

AWS ストレージ暗号化と組み合わせて、クラスター内の etcd を暗号化することもできます。その結果、暗号化が 2 倍になり、パフォーマンスが最大 20% 低下します。詳細は、etcd 暗号化 のドキュメントを参照してください。

16.1.17.4. etcd 暗号化

etcd 暗号化は、クラスター作成時にのみ有効にできます。

注記

etcd 暗号化では追加のオーバーヘッドが発生しますが、セキュリティーリスクはほとんど軽減されません。

16.1.17.5. etcd 暗号化の設定

etcd 暗号化は、OpenShift Container Platform と同じように設定されます。aescbc 暗号が使用され、設定はクラスターのデプロイメント中にパッチが適用されます。詳細は、Kubernetes のドキュメントを参照してください。

16.1.17.6. EBS 暗号化用のマルチリージョン KMS キー

現在、ROSA CLI は EBS 暗号化用のマルチリージョン KMS キーを許可しません。この機能は、製品更新のバックログに入っています。クラスター作成時に定義されている場合、ROSA CLI は EBS 暗号化用のシングルリージョン KMS キーを許可します。

16.1.18. インフラストラクチャー

ROSA は、仮想マシン、ストレージ、ロードバランサーなど、複数のクラウドサービスを使用します。定義済みのリストは、AWS の前提条件 で確認できます。

16.1.19. 認証情報メソッド

AWS アカウントで必要な操作を行うために必要な権限を付与する認証情報メソッドには、AWS with STS を使用する方法と、管理者権限を持つ IAM ユーザーを使用する方法の 2 つがあります。推奨されるのは AWS with STS で、IAM ユーザーを使用する方法はいずれ非推奨になる予定です。AWS with STS は、クラウドサービスのリソース管理における最小権限と安全なプラクティスの原則に沿うものです。

16.1.20. 前提条件となる権限または失敗エラー

ROSA CLI の新しいバージョンを確認してください。ROSA CLI の各リリースは、Github と Red Hat 署名付きバイナリーリリース の 2 カ所にあります。

16.1.21. Storage

サービス定義の ストレージ セクションを参照してください。

OpenShift には、AWS EFS 用の CSI ドライバーが含まれています。詳細は、Red Hat OpenShift Service on AWS 用の AWS EFS のセットアップ を参照してください。

16.1.22. VPC の使用

インストール時に、既存の VPC にデプロイするか、独自の VPC を使用するかを選択できます。次に、必要なサブネットを選択し、それらのサブネットを使用する際にインストールプログラムのサブネットを含む有効な CIDR 範囲を指定できます。

ROSA を使用すると、複数のクラスターが同じ VPC を共有できます。1 つの VPC 上のクラスター数は、残りの AWS リソースクォータと重複不可の CIDR 範囲で制限されます。詳細は、CIDR 範囲の定義を参照してください。

16.1.23. ネットワークプラグイン

ROSA は、OpenShift OVN-Kubernetes のデフォルトの CNI ネットワークプロバイダーを使用します。

16.1.24. cross-namespace ネットワーキング

クラスター管理者は、NetworkPolicy オブジェクトを使用して、プロジェクトごとに cross-namespace をカスタマイズまたは拒否できます。詳細は ネットワークポリシーを使用してマルチテナント分離を設定する を参照してください。

16.1.25. Prometheus と Grafana の使用

Prometheus と Grafana を使用してコンテナーを監視し、OpenShift User Workload Monitoring を使用してキャパシティを管理できます。これは、OpenShift Cluster Manager のチェックボックスオプションです。

16.1.26. クラスターコントロールプレーンから出力される監査ログ

Cluster Logging Operator アドオンがクラスターに追加されている場合は、CloudWatch を通じて監査ログを利用できます。そうでない場合は、サポートリクエストで監査ログをリクエストできます。対象を絞り、タイムボックス化された小規模なログのエクスポートをリクエストして、お客様に送信できます。利用可能な監査ログは、プラットフォームのセキュリティーとコンプライアンスのカテゴリーにおける SRE の裁量によって選ばれます。クラスターのログ全体のエクスポートリクエストは拒否されます。

16.1.27. AWS アクセス許可の境界

クラスターのポリシーの周囲に AWS アクセス許可の境界を使用できます。

16.1.28. アミ

ROSA ワーカーノードは、OSD や OpenShift Container Platform とは異なる AMI を使用します。コントロールプレーンとインフラノード AMI は、同じバージョンの製品間で共通しています。

16.1.29. クラスターのバックアップ

ROSA STS クラスターにはバックアップがありません。ユーザーは、アプリケーションとデータに対する独自のバックアップポリシーを持つ必要があります。詳細は、バックアップポリシー を参照してください。

16.1.30. カスタムドメイン

アプリケーションのカスタムドメインを定義できます。詳細は、アプリケーションのカスタムドメインの設定 を参照してください。

16.1.31. ROSA ドメイン証明書

Red Hat インフラストラクチャー (Hive) は、デフォルトのアプリケーション ingress に使用する証明書ローテーションを管理します。

16.1.32. 非接続環境

ROSA は、エアギャップのある非接続環境をサポートしません。ROSA クラスターには、レジスト

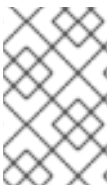
リー、S3 にアクセスしてメトリクスを送信するために、インターネットへの egress が必要です。サービスには多数の egress エンドポイントが必要です。ingress は、Red Hat SRE 用の PrivateLink とお客様のアクセス用の VPN に制限できます。

関連情報

- ROSA 製品ページ:
  - Red Hat の製品ページ
  - AWS の製品ページ
  - Red Hat カスタマーポータル
- ROSA 固有のリソース
  - AWS ROSA スタートガイド
  - ROSA ドキュメント
  - ROSA サービス定義
  - ROSA 責任分担マトリクス
  - プロセスとセキュリティーについて
  - 可用性について
  - 更新ライフサイクル
  - 制限およびスケーラビリティー
  - ROSA ロードマップ
- OpenShift について理解する
- OpenShift Cluster Manager
- Red Hat サポート

16.2. チュートリアル: AWS STS を使用する ROSA の説明

このチュートリアルでは、Red Hat OpenShift Service on AWS (ROSA) がユーザーの Amazon Web Services (AWS) アカウント内のリソースと対話できるようにする方法を 2 つ紹介します。ここでは、Security Token Service (STS) を使用する ROSA が必要な認証情報の取得に使用するコンポーネントとプロセスについて詳しく説明します。また、STS を使用する ROSA がよりセキュアで推奨される方法である理由も説明します。



注記

このコンテンツは現在、AWS STS を使用する ROSA Classic を対象としています。AWS STS を使用する ROSA with Hosted Control Plane (HCP) についてはまだ考慮していません。

このチュートリアルでは次のことを行います。

- 2 つのデプロイ方法を示します。
  - IAM ユーザーを使用する ROSA
  - STS を使用する ROSA
- 2 つの方法の違いを説明します。
- STS を使用する ROSA がよりセキュアで推奨される方法である理由を説明します。
- STS を使用する ROSA がどのように機能するかを説明します。

16.2.1. ROSA をデプロイするための各種認証方法

Red Hat は、ROSA の一部として、お客様の AWS アカウント内のインフラストラクチャーリソースを管理します。そのため、Red Hat に必要な権限を付与する必要があります。必要な権限を付与する方法は、現在、次の 2 つのものがサポートされています。

- **AdministratorAccess** ポリシーで静的 IAM ユーザー認証情報を使用する
  このチュートリアルでは、この方法を "IAM ユーザーを使用する ROSA" と呼びます。これは推奨される認証方法ではありません。
- AWS STS と有効期間の短い動的トークンを使用する
  このチュートリアルでは、この方法を "STS を使用する ROSA" と呼びます。これは推奨される認証方法です。

16.2.1.1. IAM ユーザーを使用する ROSA

ROSA のリリース当初、認証方法は IAM ユーザーを使用する ROSA だけでした。この方法では、**AdministratorAccess** ポリシーを持つ IAM ユーザーに対して、ROSA を使用する AWS アカウントに必要なリソースを作成するためのフルアクセスを付与します。その後、必要に応じてクラスターで認証情報を作成および拡張できます。

16.2.1.2. STS を使用する ROSA

STS を使用する ROSA では、AWS アカウント内のリソースへの限定的かつ短期間のアクセスをユーザーに付与します。STS による方法では、事前定義されたロールとポリシーを使用して、IAM ユーザーまたは認証されたフェデレーションユーザーに一時的な最小限の権限を付与します。通常、認証情報は要求されてから 1 時間後に期限切れになります。有効期限が切れると、認証情報は AWS によって認識されなくなり、その認証情報を使用して実行される API 要求からアカウントにアクセスできなくなります。詳細は、AWS のドキュメント を参照してください。現在、IAM ユーザーを使用する ROSA と STS を使用する ROSA の両方が有効ですが、STS を使用する ROSA が優先および推奨される方法です。

16.2.2. STS を使用する ROSA のセキュリティー

STS を使用する ROSA は、以下に示すいくつかの重要な要素により、IAM ユーザーを使用する ROSA よりもセキュリティーが向上します。

- ユーザーが事前に作成する明示的かつ限定的なロールとポリシーのセット。要求されたすべての権限と使用されるすべてのロールをユーザーが把握することになります。
- サービスは、これらの権限以外の操作を一切行うことができません。
- サービスは、操作を実行する必要が生じるたびに、1 時間以内に期限切れになる認証情報を取得します。そのため、認証情報のローテーションや取り消しが不要になります。さらに、認証情報の有効期限により、認証情報の漏洩や再利用のリスクが軽減されます。

16.2.3. AWS STS の説明

ROSA は、AWS STS を使用して、短期間のセキュリティー認証情報を持つ最小限の権限を、分離された特定の IAM ロールに付与します。認証情報は、AWS の API 呼び出しを実行する各コンポーネントおよびクラスターに固有の IAM ロールに関連付けられます。この方法は、クラウドサービスのリソース管理における最小権限と安全なプラクティスの原則に沿ったものです。ROSA コマンドラインインターフェイス (CLI) ツールは、固有のタスクに割り当てられた STS のロールとポリシーを管理し、OpenShift 機能の一部として AWS リソースに対してアクションを実行します。

STS のロールとポリシーは、ROSA クラスターごとに作成する必要があります。これを容易にするために、インストールツールには、ロールをポリシーとして作成するために必要なすべてのコマンドおよびファイルと、CLI でロールとポリシーを自動的に作成できるようにするオプションが用意されています。さまざまな **--mode** オプションの詳細は、カスタマイズを使用した STS を使用する ROSA クラスターの作成 を参照してください。

16.2.4. STS を使用する ROSA に固有のコンポーネント

- **AWS インフラストラクチャー** - クラスターに必要なインフラストラクチャーを提供します。これには、実際の EC2 インスタンス、ストレージ、ネットワークコンポーネントが含まれます。コンピュートノードでサポートされているインスタンスタイプと、コントロールプレーンとインフラストラクチャーノードの設定用に プロビジョニングされる AWS インフラストラクチャー を確認するには、AWS コンピュートタイプ を参照してください。

- **AWS STS** - 上記の認証方法のセクションを参照してください。

- **OpenID Connect (OIDC)** - クラスター Operator が AWS で認証し、信頼ポリシーを通じてクラスターのロールを引き受け、必要な API 呼び出しを実行するために STS から一時的な認証情報を取得するためのメカニズムを提供します。

- **ロールとポリシー** - ロールとポリシーは、STS を使用する ROSA と IAM ユーザーを使用する ROSA の主な違いの 1 つです。STS を使用する ROSA の場合、ROSA で使用されるロールとポリシーは、アカウント全体のロールとポリシーと Operator のロールとポリシーに分類されます。
  ポリシーは、各ロールに対して許可されるアクションを決定します。個々のロールとポリシーの詳細は、STS を使用する ROSA クラスターの IAM リソースについて を参照してください。

  - アカウント全体のロールは次のとおりです。
    - ManagedOpenShift-Installer-Role
    - ManagedOpenShift-ControlPlane-Role
    - ManagedOpenShift-Worker-Role
    - ManagedOpenShift-Support-Role
  - アカウント全体のポリシーは次のとおりです。
    - ManagedOpenShift-Installer-Role-Policy
    - ManagedOpenShift-ControlPlane-Role-Policy
    - ManagedOpenShift-Worker-Role-Policy
    - ManagedOpenShift-Support-Role-Policy
    - ManagedOpenShift-openshift-ingress-operator-cloud-credentials [1]

- ManagedOpenShift-openshift-cluster-csi-drivers-ebs-cloud-credent [1]
- ManagedOpenShift-openshift-cloud-network-config-controller-cloud [1]
- ManagedOpenShift-openshift-machine-api-aws-cloud-credentials [1]
- ManagedOpenShift-openshift-cloud-credential-operator-cloud-crede [1]
- ManagedOpenShift-openshift-image-registry-installer-cloud-creden [1]

1. このポリシーは、下記のクラスター Operator ロールによって使用されます。Operator ロールは既存のクラスター名に依存しており、アカウント全体のロールと同時に作成できないため、2 番目のステップで作成されます。

- Operator のロールは次のとおりです。
  - <cluster-name\>-xxxx-openshift-cluster-csi-drivers-ebs-cloud-credent
  - <cluster-name\>-xxxx-openshift-cloud-network-config-controller-cloud
  - <cluster-name\>-xxxx-openshift-machine-api-aws-cloud-credentials
  - <cluster-name\>-xxxx-openshift-cloud-credential-operator-cloud-crede
  - <cluster-name\>-xxxx-openshift-image-registry-installer-cloud-creden
  - <cluster-name\>-xxxx-openshift-ingress-operator-cloud-credentials
- 信頼ポリシーは、アカウント全体および Operator のロールごとに作成されます。

16.2.5. ROSA STS クラスターのデプロイ

以下の手順にリストされているリソースを最初から作成する必要はありません。ROSA CLI が必要な JSON ファイルを作成し、必要なコマンドを出力します。さらに、必要に応じて ROSA CLI にコマンドを実行させることもできます。

STS を使用する ROSA クラスターをデプロイする手順

1. アカウント全体のロールとポリシーを作成します。
2. 権限ポリシーを、対応するアカウント全体のロールに割り当てます。
3. クラスターを作成します。
4. Operator のロールとポリシーを作成します。
5. 権限ポリシーを、対応する Operator のロールに割り当てます。
6. OIDC プロバイダーを作成します。

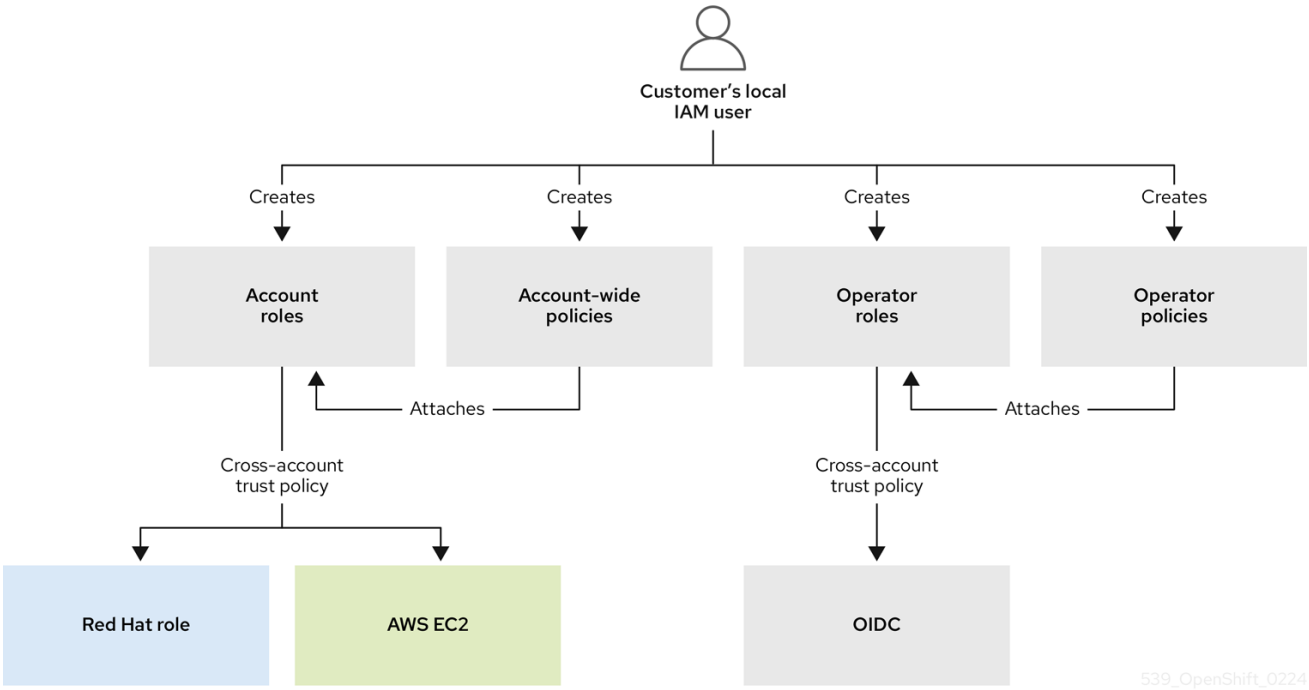
ロールとポリシーは、ROSA CLI によって自動的に作成することも、ROSA CLI で **--mode manual** フラグまたは **--mode auto** フラグを利用して手動で作成することもできます。デプロイの詳細は、カスタマイズしたクラスターを作成する または クラスターのデプロイのチュートリアル を参照してください。

16.2.6. STS を使用する ROSA のワークフロー

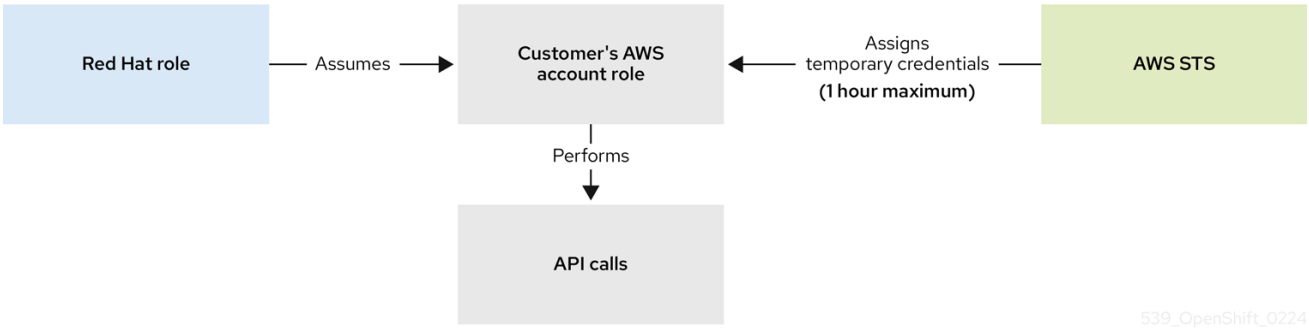
ユーザーは、必要なアカウント全体のロールとアカウント全体のポリシーを作成します。詳細は、このチュートリアルのコンポーネントのセクションを参照してください。ロールの作成時に、クロスアカウント信頼ポリシーという信頼ポリシーが作成されます。このポリシーは、Red Hat 所有のロールがこのロールを引き受けることを許可するものです。また、EC2 サービス用の信頼ポリシーも作成されます。このポリシーは、EC2 インスタンス上のワークロードがロールを引き受けて認証情報を取得することを許可するものです。ユーザーは、対応する権限ポリシーを各ロールに割り当てることができます。

アカウント全体のロールとポリシーを作成した後、ユーザーはクラスターを作成できます。クラスターの作成を開始すると、クラスター Operator が AWS API 呼び出しを実行できるように、複数の Operator ロールが作成されます。これらのロールを、以前に作成された対応する権限ポリシーと、OIDC プロバイダーの信頼ポリシーに割り当てます。Operator ロールは、AWS リソースへのアクセスが必要な Pod を最終的に表すという点で、アカウント全体のロールとは異なります。ユーザーは IAM ロールを Pod に割り当てることができないため、Operator (ひいては Pod) が必要なロールにアクセスできるように、OIDC プロバイダーを使用して信頼ポリシーを作成する必要があります。

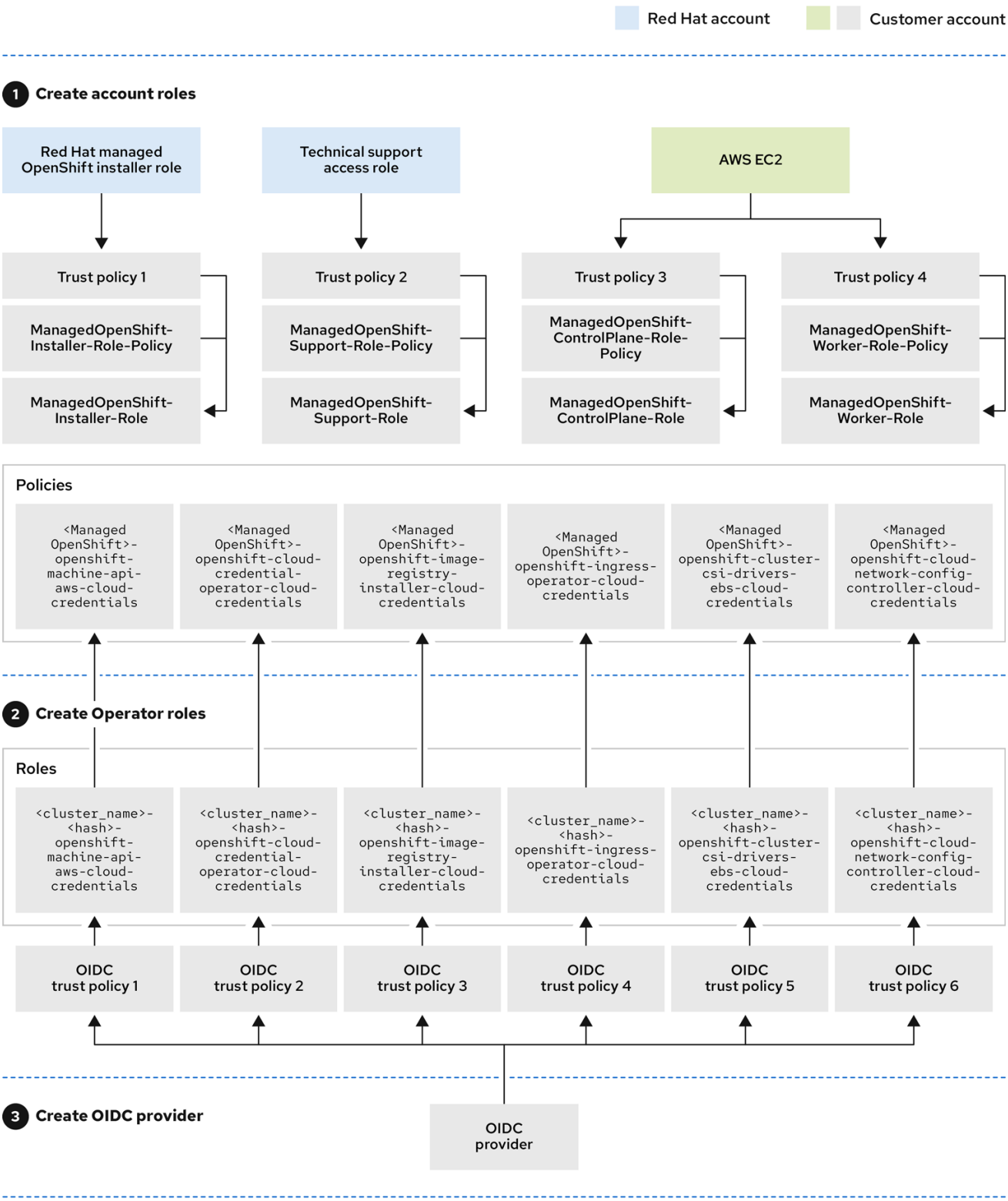
対応する権限ポリシーにロールを割り当てたら、最後のステップとして OIDC プロバイダーを作成します。



新しいロールが必要な場合、現在 Red Hat のロールを使用しているワークロードが AWS アカウントのロールを引き受け、AWS STS から一時的な認証情報を取得し、引き受けたロールの権限ポリシーに従ってお客様の AWS アカウント内の API 呼び出しを使用してアクションの実行を開始します。認証情報は一時的なもので、有効期間は最大 1 時間です。

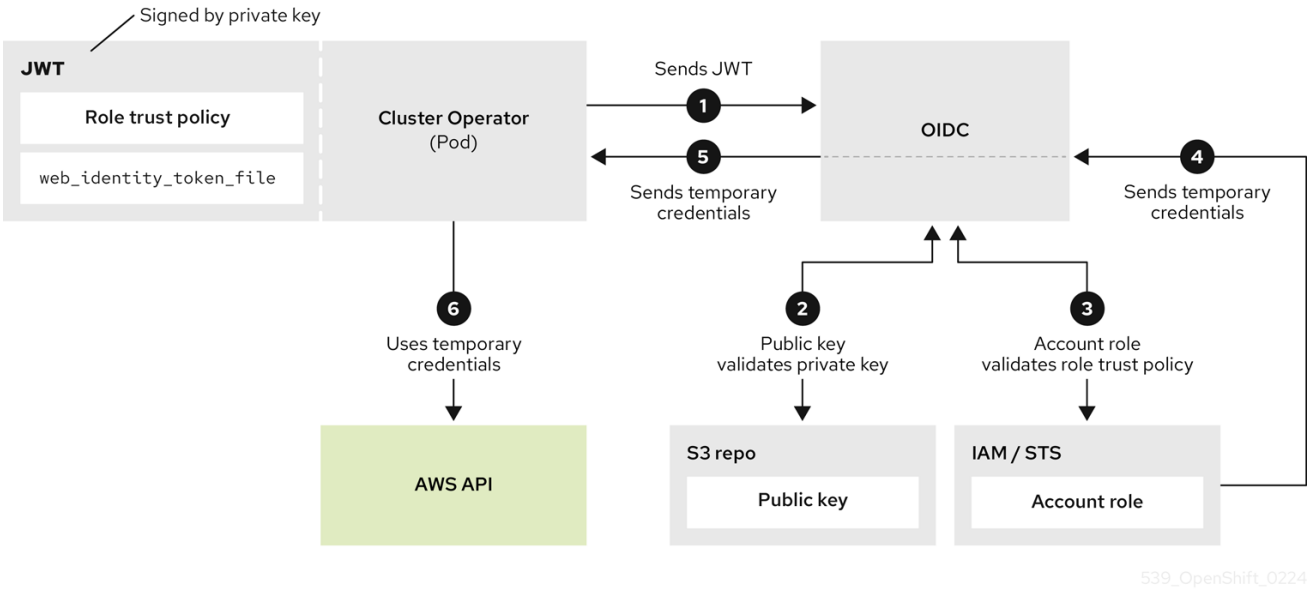


ワークフロー全体を次の図に示します。



Operator は、次のプロセスを使用して、タスクを実行するために必要な認証情報を取得します。各 Operator には、Operator のロール、権限ポリシー、および OIDC プロバイダーの信頼ポリシーが割り当てられます。Operator は、ロールとトークンファイル (**web_identity_token_file**) を含む JSON Web トークンを OIDC プロバイダーに渡すことによってロールを引き受けます。OIDC プロバイダーは署名された鍵を公開鍵で認証します。公開鍵はクラスターの作成時に作成され、S3 バケットに保存されます。次に、Operator は、署名されたトークンファイル内のサブジェクトがロール信頼ポリシー内のロールと一致することを確認します。このロールは、OIDC プロバイダーが許可されたロールのみを取

得できるようにするためのものです。その後、OIDC プロバイダーが一時的な認証情報を Operator に返し、Operator が AWS API 呼び出しを実行できるようにします。視覚的に表した図を以下に示します。



16.2.7. STS を使用する ROSA のユースケース

クラスターのインストール時にノードを作成する

Red Hat のインストールプログラムは、**RH-Managed-OpenShift-Installer** ロールと信頼ポリシーを使用して、お客様のアカウントの **Managed-OpenShift-Installer-Role** ロールを引き受けます。このプロセスにより、AWS STS から一時的な認証情報が返されます。インストールプログラムは、STS から受け取った一時的な認証情報を使用して、必要な API 呼び出しの実行を開始します。インストールプログラムは必要なインフラストラクチャーを AWS に作成します。この認証情報は 1 時間以内に期限切れになり、インストールプログラムはお客様のアカウントにアクセスできなくなります。

このプロセスはサポートケースにも適用されます。サポートケースでは、インストールプログラムの代わりに Red Hat Site Reliability Engineer (SRE) がプロセスを実行します。

クラスターのスケーリング

machine-api-operator は、AssumeRoleWithWebIdentity を使用して **machine-api-aws-cloud-credentials** ロールを引き受けます。これにより、クラスター Operator が認証情報を受け取るシーケンスが開始します。その後、**machine-api-operator** ロールが関連する API 呼び出しを実行して、クラスターに EC2 インスタンスをさらに追加できるようになります。

16.3. クラスターのデプロイ

16.3.1. チュートリアル: デプロイ方法の選択

クラスターをデプロイするにはいくつかの異なる方法があります。このチュートリアルではそれぞれの方法の概要を説明します。希望やニーズに応じて 1 つ選択してください。以下のガイドを使用して、状況に最も適したデプロイ方法を参照してください。

16.3.1.1. デプロイ方法

ご希望に応じて以下から選択してください。

- 実行する必要があるコマンドだけを知りたい - CLI 簡易ガイド
- CLI ツールが好きではなく、ユーザーインターフェイスを使用したい - UI 簡易ガイド
- 詳細が必要で、CLI を使用したい - CLI 詳細ガイド
- 詳細が必要で、ユーザーインターフェイスを使用したい - UI 詳細ガイド
- 最新のテクノロジーを試したい - HCP を備えた ROSA

上記のデプロイ方法はすべて、このワークショップで問題なく使用できます。このワークショップを初めて行う場合は、CLI 簡易ガイド が最も簡単で推奨される方法です。

16.3.2. チュートリアル: CLI 簡易ガイド

このページでは、コマンドラインインターフェイス (CLI) を使用して Red Hat OpenShift Service on AWS (ROSA) クラスターをデプロイするための最小限のコマンドを説明します。

注記

この単純なデプロイメントはチュートリアル環境では問題なく機能しますが、実稼働環境で使用するクラスターはより詳細な方法でデプロイする必要があります。

16.3.2.1. 前提条件

- セットアップのチュートリアルの前提条件を完了している。

16.3.2.2. アカウントロールの作成

AWS アカウントおよび OpenShift の y-stream バージョンごとに次のコマンドを **1 回** 実行します。

```
rosa create account-roles --mode auto --yes
```

16.3.2.3. クラスターのデプロイ

1. 次のコマンドを実行して、デフォルト設定でクラスターを作成します。クラスター名は独自のものに置き換えてください。

   ```
   rosa create cluster --cluster-name <cluster-name> --sts --mode auto --yes
   ```

2. 次のコマンドを実行して、クラスターのステータスを確認します。

   ```
   rosa list clusters
   ```

16.3.3. チュートリアル: CLI 詳細ガイド

このチュートリアルでは、ROSA CLI を使用して ROSA クラスターをデプロイする詳細な手順を説明します。

16.3.3.1. CLI のデプロイメントモード

ROSA クラスターのデプロイには 2 つのモードがあります。1 つは自動モードです。より迅速であり、作業が自動的に実行されます。もう 1 つは手動モードです。追加のコマンドを実行する必要があり、作

成中のロールとポリシーを確認できます。このチュートリアルでは、両方のオプションについて説明します。

クラスターを迅速に作成する場合は、自動オプションを使用します。作成中のロールとポリシーを確認する場合は、手動オプションを使用します。

デプロイメントモードを選択するには、関連するコマンドで **--mode** フラグを使用します。

--mode の有効なオプションは次のとおりです。

- **manual:** ロールとポリシーが作成され、現在のディレクトリーに保存されます。その後、指定のコマンドを手動で実行する必要があります。このオプションを使用すると、ポリシーとロールを作成する前に確認できます。
- **auto:** ロールとポリシーが現在の AWS アカウントを使用して自動的に作成され、適用されます。

ヒント

このチュートリアルではどちらのデプロイ方法も使用できます。**auto** モードはより高速で、ステップ数が少なくなります。

16.3.3.2. デプロイメントワークフロー

全体的なデプロイメントワークフローは、次のステップに基づいています。

1. **rosa create account-roles** - これはアカウントごとに **1回** だけ実行します。アカウントロールを一度作成したら、同じ y-stream バージョンの他のクラスターに対して再度作成する必要は**ありません**。
2. **rosa create cluster**
3. **rosa create operator-roles** - 手動モードのみ。
4. **rosa create oidc-provider** - 手動モードのみ。

自動モードの場合、同じ y-stream バージョンの同じアカウント内のクラスターごとに、ステップ 2 のみを実行する必要があります。手動モードの場合、ステップ 2 - 4 を実行する必要があります。

16.3.3.3. 自動モード

ROSA CLI でロールとポリシーの作成を自動化し、クラスターを迅速に作成する場合は、この方法を使用します。

16.3.3.3.1. アカウントロールの作成

このアカウントに ROSA を **初めて** デプロイし、アカウントロールをまだ **作成していない** 場合は、Operator ポリシーを含むアカウント全体のロールとポリシーを作成します。

次のコマンドを実行して、アカウント全体のロールを作成します。

```
rosa create account-roles --mode auto --yes
```

出力例

```
I: Creating roles using 'arn:aws:iam::000000000000:user/rosa-user'
I: Created role 'ManagedOpenShift-ControlPlane-Role' with ARN
'arn:aws:iam::000000000000:role/ManagedOpenShift-ControlPlane-Role'
I: Created role 'ManagedOpenShift-Worker-Role' with ARN
'arn:aws:iam::000000000000:role/ManagedOpenShift-Worker-Role'
I: Created role 'ManagedOpenShift-Support-Role' with ARN
'arn:aws:iam::000000000000:role/ManagedOpenShift-Support-Role'
I: Created role 'ManagedOpenShift-Installer-Role' with ARN
'arn:aws:iam::000000000000:role/ManagedOpenShift-Installer-Role'
I: Created policy with ARN 'arn:aws:iam::000000000000:policy/ManagedOpenShift-openshift-
machine-api-aws-cloud-credentials'
I: Created policy with ARN 'arn:aws:iam::000000000000:policy/ManagedOpenShift-openshift-cloud-
credential-operator-cloud-crede'
I: Created policy with ARN 'arn:aws:iam::000000000000:policy/ManagedOpenShift-openshift-image-
registry-installer-cloud-creden'
I: Created policy with ARN 'arn:aws:iam::000000000000:policy/ManagedOpenShift-openshift-ingress-
operator-cloud-credentials'
I: Created policy with ARN 'arn:aws:iam::000000000000:policy/ManagedOpenShift-openshift-cluster-
csi-drivers-ebs-cloud-credent'
I: To create a cluster with these roles, run the following command:
    rosa create cluster --sts
```

16.3.3.3.2. クラスターの作成

次のコマンドを実行して、すべてのデフォルトオプションを使用してクラスターを作成します。

```
rosa create cluster --cluster-name <cluster-name> --sts --mode auto --yes
```

注記

これにより、必要な Operator ロールと OIDC プロバイダーも作成されます。クラスターで使用可能なすべてのオプションを確認する場合は、**--help** フラグを使用するか、対話モードの場合は **--interactive** を使用してください。

入力の例

```
$ rosa create cluster --cluster-name my-rosa-cluster --sts --mode auto --yes
```

出力例

```
I: Creating cluster 'my-rosa-cluster'
I: To view a list of clusters and their status, run 'rosa list clusters'
I: Cluster 'my-rosa-cluster' has been created.
I: Once the cluster is installed you will need to add an Identity Provider before you can login into the
cluster. See 'rosa create idp --help' for more information.
I: To determine when your cluster is Ready, run 'rosa describe cluster -c my-rosa-cluster'.
I: To watch your cluster installation logs, run 'rosa logs install -c my-rosa-cluster --watch'.
Name:                       my-rosa-cluster
ID:                         1mlhulb3bo0l54ojd0ji000000000000
External ID:
OpenShift Version:
Channel Group:              stable
DNS:                        my-rosa-cluster.ibhp.p1.openshiftapps.com
```

```
AWS Account:                 000000000000
API URL:
Console URL:
Region:                      us-west-2
Multi-AZ:                    false
Nodes:
- Master:                    3
- Infra:                     2
- Compute:                   2
Network:
- Service CIDR:              172.30.0.0/16
- Machine CIDR:              10.0.0.0/16
- Pod CIDR:                  10.128.0.0/14
- Host Prefix:               /23
STS Role ARN:                arn:aws:iam::000000000000:role/ManagedOpenShift-Installer-Role
Support Role ARN:            arn:aws:iam::000000000000:role/ManagedOpenShift-Support-Role
Instance IAM Roles:
- Master:                    arn:aws:iam::000000000000:role/ManagedOpenShift-ControlPlane-Role
- Worker:                    arn:aws:iam::000000000000:role/ManagedOpenShift-Worker-Role
Operator IAM Roles:
- arn:aws:iam::000000000000:role/my-rosa-cluster-openshift-image-registry-installer-cloud-
credentials
- arn:aws:iam::000000000000:role/my-rosa-cluster-openshift-ingress-operator-cloud-credentials
- arn:aws:iam::000000000000:role/my-rosa-cluster-openshift-cluster-csi-drivers-ebs-cloud-credentials
- arn:aws:iam::000000000000:role/my-rosa-cluster-openshift-machine-api-aws-cloud-credentials
- arn:aws:iam::000000000000:role/my-rosa-cluster-openshift-cloud-credential-operator-cloud-
credential-oper
State:                       waiting (Waiting for OIDC configuration)
Private:                     No
Created:                     Oct 28 2021 20:28:09 UTC
Details Page:
https://console.redhat.com/openshift/details/s/1wupmiQy45xr1nN000000000000
OIDC Endpoint URL:           https://rh-oidc.s3.us-east-
1.amazonaws.com/1mlhulb3bo0l54ojd0ji000000000000
```

16.3.3.3.2.1. デフォルト設定

デフォルトの設定は次のとおりです。

- ノード:
  - 3 つのコントロールプレーンノード
  - 2 つのインフラストラクチャーノード
  - 2 つのワーカーノード
  - 自動スケーリングなし
  - 詳細は、ec2 インスタンス のドキュメントを参照してください。
- リージョン: **aws** CLI の設定どおり
- ネットワーク IP 範囲:
  - Machine CIDR: 10.0.0.0/16

- Service CIDR: 172.30.0.0/16
- Pod CIDR: 10.128.0.0/14

- 新しい VPC
- 暗号化用のデフォルトの AWS KMS キー
- **rosa** で利用可能な OpenShift の最新バージョン
- 単一のアベイラビリティーゾーン
- パブリッククラスター

16.3.3.3.3. インストールステータスの確認

1. 次のコマンドのいずれかを実行して、クラスターのステータスを確認します。
   - ステータスの詳細を表示するには、次を実行します。

     ```
     rosa describe cluster --cluster <cluster-name>
     ```

   - ステータスの要約を表示するには、次を実行します。

     ```
     rosa list clusters
     ```

2. クラスターの状態が、“waiting”、“installing”、"ready" の順に変わります。これには約 40 分かかります。
3. 状態が “ready” に変われば、クラスターのインストールは完了です。

16.3.3.4. 手動モード

ロールとポリシーをクラスターに適用する前に確認する場合は、手動の方法を使用します。この方法では、ロールとポリシーを作成するためにいくつかの追加コマンドを実行する必要があります。

このセクションでは **--interactive** モードを使用します。このセクションのフィールドの説明については、対話モード に関するドキュメントを参照してください。

16.3.3.4.1. アカウントロールの作成

1. このアカウントに ROSA を **初めて** デプロイし、アカウントロールをまだ **作成していない** 場合は、Operator ポリシーを含むアカウント全体のロールとポリシーを作成します。このコマンドは、アカウントに必要なロールとポリシーに必要な JSON ファイルを現在のディレクトリーに作成します。また、これらのオブジェクトを作成するために実行する必要がある **aws** CLI コマンドも出力されます。
   次のコマンドを実行して必要なファイルを作成し、追加のコマンドを出力します。

   ```
   rosa create account-roles --mode manual
   ```

   出力例

   ```
   I: All policy files saved to the current directory
   I: Run the following commands to create the account roles and policies:
   ```

```
aws iam create-role \
--role-name ManagedOpenShift-Worker-Role \
--assume-role-policy-document file://sts_instance_worker_trust_policy.json \
--tags Key=rosa_openshift_version,Value=4.8
Key=rosa_role_prefix,Value=ManagedOpenShift
Key=rosa_role_type,Value=instance_worker
aws iam put-role-policy \
--role-name ManagedOpenShift-Worker-Role \
--policy-name ManagedOpenShift-Worker-Role-Policy \
--policy-document file://sts_instance_worker_permission_policy.json
```

2. 現在のディレクトリーの内容をチェックして、新しいファイルを確認します。これらの各オブジェクトを作成するには、**aws** CLI を使用します。

出力例

```
$ ls
openshift_cloud_credential_operator_cloud_credential_operator_iam_ro_creds_policy.json
sts_instance_controlplane_permission_policy.json
openshift_cluster_csi_drivers_ebs_cloud_credentials_policy.json
sts_instance_controlplane_trust_policy.json
openshift_image_registry_installer_cloud_credentials_policy.json
sts_instance_worker_permission_policy.json
openshift_ingress_operator_cloud_credentials_policy.json
sts_instance_worker_trust_policy.json
openshift_machine_api_aws_cloud_credentials_policy.json
sts_support_permission_policy.json
sts_installer_permission_policy.json                    sts_support_trust_policy.json
sts_installer_trust_policy.json
```

3. **オプション:** ファイルを開いて、作成する内容を確認します。たとえば、**sts_installer_permission_policy.json** を開くと、次のように表示されます。

出力例

```
$ cat sts_installer_permission_policy.json
        {
        "Version": "2012-10-17",
        "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "autoscaling:DescribeAutoScalingGroups",
                "ec2:AllocateAddress",
                "ec2:AssociateAddress",
                "ec2:AssociateDhcpOptions",
                "ec2:AssociateRouteTable",
                "ec2:AttachInternetGateway",
                "ec2:AttachNetworkInterface",
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
                [...]
```

ROSA クラスターの IAM リソースについて ドキュメントの内容も参照してください。

4. ステップ 1 でリストされた **aws** コマンドを実行します。作成した JSON ファイルと同じディレクトリーにいる場合は、コピーして貼り付けることができます。

16.3.3.4.2. クラスターの作成

1. **aws** コマンドが正常に実行されたら、次のコマンドを実行して対話モードで ROSA クラスターの作成を開始します。

```
rosa create cluster --interactive --sts
```

フィールドの説明については、ROSA のドキュメント を参照してください。

2. このチュートリアルでは、次の値をコピーして入力します。

```
Cluster name: **my-rosa-cluster** <br>
OpenShift version: **&lt;choose version&gt;** <br>
External ID (optional): **&lt;leave blank&gt;**<br>
Operator roles prefix: **&lt;accept default&gt;** <br>
Multiple availability zones: **No** <br>
AWS region: **&lt;choose region&gt;** <br>
PrivateLink cluster: **No** <br>
Install into an existing VPC: **No** <br>
Enable Customer Managed key: **No** <br>
Compute nodes instance type: **m5.xlarge** <br>
Enable autoscaling: **No** <br>
Compute nodes: **2** <br>
Machine CIDR: **&lt;accept default&gt;** <br>
Service CIDR: **&lt;accept default&gt;** <br>
Pod CIDR: **&lt;accept default&gt;** <br>
Host prefix: **&lt;accept default&gt;** <br>
Encrypt etcd data (optional): **No** <br>
Disable Workload monitoring: **No** <br>
```

出力例

```
I: Creating cluster 'my-rosa-cluster'
I: To create this cluster again in the future, you can run:
rosa create cluster --cluster-name my-rosa-cluster --role-arn
arn:aws:iam::000000000000:role/ManagedOpenShift-Installer-Role --support-role-arn
arn:aws:iam::000000000000:role/ManagedOpenShift-Support-Role --master-iam-role
arn:aws:iam::000000000000:role/ManagedOpenShift-ControlPlane-Role --worker-iam-role
arn:aws:iam::000000000000:role/ManagedOpenShift-Worker-Role --operator-roles-prefix
my-rosa-cluster --region us-west-2 --version 4.8.13 --compute-nodes 2 --machine-cidr
10.0.0.0/16 --service-cidr 172.30.0.0/16 --pod-cidr 10.128.0.0/14 --host-prefix 23
I: To view a list of clusters and their status, run 'rosa list clusters'
I: Cluster 'my-rosa-cluster' has been created.
I: Once the cluster is installed you will need to add an Identity Provider before you can login
into the cluster. See 'rosa create idp --help' for more information.
Name:                       my-rosa-cluster
ID:                         1t6i760dbum4mqltqh6o000000000000
External ID:
OpenShift Version:
Channel Group:              stable
DNS:                        my-rosa-cluster.abcd.p1.openshiftapps.com
AWS Account:                000000000000
```

```
API URL:
Console URL:
Region:                     us-west-2
Multi-AZ:                   false
Nodes:
- Control plane:            3
- Infra:                    2
- Compute:                  2
Network:
- Service CIDR:             172.30.0.0/16
- Machine CIDR:             10.0.0.0/16
- Pod CIDR:                 10.128.0.0/14
- Host Prefix:              /23
STS Role ARN:               arn:aws:iam::000000000000:role/ManagedOpenShift-Installer-Role
Support Role ARN:           arn:aws:iam::000000000000:role/ManagedOpenShift-Support-Role
Instance IAM Roles:
- Control plane:            arn:aws:iam::000000000000:role/ManagedOpenShift-ControlPlane-
Role
- Worker:                   arn:aws:iam::000000000000:role/ManagedOpenShift-Worker-Role
Operator IAM Roles:
- arn:aws:iam::000000000000:role/my-rosa-cluster-w7i6-openshift-ingress-operator-cloud-
credentials
- arn:aws:iam::000000000000:role/my-rosa-cluster-w7i6-openshift-cluster-csi-drivers-ebs-
cloud-credentials
- arn:aws:iam::000000000000:role/my-rosa-cluster-w7i6-openshift-cloud-network-config-
controller-cloud-cre
- arn:aws:iam::000000000000:role/my-rosa-cluster-openshift-machine-api-aws-cloud-
credentials
- arn:aws:iam::000000000000:role/my-rosa-cluster-openshift-cloud-credential-operator-cloud-
credentia
- arn:aws:iam::000000000000:role/my-rosa-cluster-openshift-image-registry-installer-cloud-
credential
State:                      waiting (Waiting for OIDC configuration)
Private:                    No
Created:                    Jul  1 2022 22:13:50 UTC
Details Page:
https://console.redhat.com/openshift/details/s/2BMQm8xz8Hq5yEN000000000000
OIDC Endpoint URL:          https://rh-oidc.s3.us-east-
1.amazonaws.com/1t6i760dbum4mqltqh6o000000000000
I: Run the following commands to continue the cluster creation:
rosa create operator-roles --cluster my-rosa-cluster
rosa create oidc-provider --cluster my-rosa-cluster
I: To determine when your cluster is Ready, run 'rosa describe cluster -c my-rosa-cluster'.
I: To watch your cluster installation logs, run 'rosa logs install -c my-rosa-cluster --watch'.
```

注記

次の 2 つのステップが完了するまで、クラスターの状態は “waiting” のままです。

16.3.3.4.3. Operator ロールの作成

1. 上記のステップで、次に実行すべきコマンドが出力されます。これらのロールは、クラスター**ごと** に **1回** 作成する必要があります。ロールを作成するには、次のコマンドを実行します。

```
rosa create operator-roles --mode manual --cluster <cluster-name>
```

出力例

```
I: Run the following commands to create the operator roles:
   aws iam create-role \
       --role-name my-rosa-cluster-openshift-image-registry-installer-cloud-credentials \
       --assume-role-policy-document
file://operator_image_registry_installer_cloud_credentials_policy.json \
       --tags Key=rosa_cluster_id,Value=1mkesci269png3tck000000000000000
Key=rosa_openshift_version,Value=4.8 Key=rosa_role_prefix,Value=
Key=operator_namespace,Value=openshift-image-registry
Key=operator_name,Value=installer-cloud-credentials

   aws iam attach-role-policy \
       --role-name my-rosa-cluster-openshift-image-registry-installer-cloud-credentials \
       --policy-arn arn:aws:iam::000000000000:policy/ManagedOpenShift-openshift-image-
registry-installer-cloud-creden
   [...]
```

2. 各 **aws** コマンドを実行します。

16.3.3.4.4. OIDC プロバイダーの作成

1. 次のコマンドを実行して、OIDC プロバイダーを作成します。

```
rosa create oidc-provider --mode manual --cluster <cluster-name>
```

2. これにより、実行する必要がある **aws** コマンドが表示されます。

出力例

```
I: Run the following commands to create the OIDC provider:
$ aws iam create-open-id-connect-provider \
--url https://rh-oidc.s3.us-east-1.amazonaws.com/1mkesci269png3tckknhh0rfs2da5fj9 \
--client-id-list openshift sts.amazonaws.com \
--thumbprint-list a9d53002e97e00e043244f3d170d000000000000

$ aws iam create-open-id-connect-provider \
--url https://rh-oidc.s3.us-east-1.amazonaws.com/1mkesci269png3tckknhh0rfs2da5fj9 \
--client-id-list openshift sts.amazonaws.com \
--thumbprint-list a9d53002e97e00e043244f3d170d000000000000
```

3. クラスターがインストールプロセスを続行します。

16.3.3.4.5. インストールステータスの確認

1. 次のコマンドのいずれかを実行して、クラスターのステータスを確認します。

   - ステータスの詳細を表示するには、次を実行します。

     ```
     rosa describe cluster --cluster <cluster-name>
     ```

- ステータスの要約を表示するには、次を実行します。

```
rosa list clusters
```

2. クラスターの状態が、“waiting”、“installing”、"ready" の順に変わります。これには約 40 分かかります。

3. 状態が “ready” に変われば、クラスターのインストールは完了です。

16.3.3.5. コンソール URL の取得

- コンソール URL を取得するには、次のコマンドを実行します。

```
rosa describe cluster -c <cluster-name> | grep Console
```

これで、クラスターが正常にデプロイされました。次のチュートリアルでは、クラスターをすぐに使用できるように、管理ユーザーを作成する方法を示します。

16.3.4. チュートリアル: Hosted Control Plane ガイド

このチュートリアルでは、Hosted Control Plane (HCP) を備えた Red Hat OpenShift Service on AWS (ROSA) クラスターのデプロイについて概説します。

HCP を備えた ROSA を使用すると、コントロールプレーンをデータプレーンから分離できます。これは、コントロールプレーンを Red Hat 所有の AWS アカウントでホストする、ROSA の新しいデプロイメントモデルです。コントロールプレーンが AWS アカウントでホストされないため、AWS インフラストラクチャーの費用が削減されます。コントロールプレーンは単一のクラスター専用であり、高可用性を実現します。詳細は、HCP を備えた ROSA のドキュメント を参照してください。

16.3.4.1. 前提条件

HCP を備えた ROSA クラスターをデプロイするには、次のリソースが必要です。

- VPC - これは持ち込み型の VPC モデルであり、BYO-VPC とも呼ばれます。
- OIDC - OIDC 設定と特定の設定を持つ OIDC プロバイダー。
- ROSA バージョン 1.2.31 以降

このチュートリアルでは、最初にこれらのリソースを作成します。また、HCP を備えた ROSA クラスターを作成するコマンドを簡単に実行できるように、いくつかの環境変数も設定します。

16.3.4.1.1. VPC の作成

1. まず、HCP を備えた ROSA を利用できるリージョンを使用するように AWS CLI (**aws**) が設定されていることを確認します。サポート対象のリージョンを確認するには、次のコマンドを実行します。

```
rosa list regions --hosted-cp
```

2. VPC を作成します。このチュートリアル用の次のスクリプトを実行すると、VPC とそれに必要なコンポーネントが作成されます。これは、**aws** CLI 用に設定されたリージョンを使用します。

```
curl https://raw.githubusercontent.com/openshift-cs/rosaworkshop/master/rosa-workshop/rosa/resources/setup-vpc.sh | bash
```

VPC の要件の詳細は、VPC のドキュメント を参照してください。

3. 上記のスクリプトは 2 つのコマンドを出力します。これらのコマンドを環境変数として設定すると、**create cluster** コマンドの実行が容易になります。出力からコマンドをコピーし、次のように実行します。

```
export PUBLIC_SUBNET_ID=<public subnet id here>
export PRIVATE_SUBNET_ID=<private subnet id here>
```

4. 次のコマンドを実行して、環境変数が設定されていることを確認します。

```
echo "Public Subnet: $PUBLIC_SUBNET_ID"; echo "Private Subnet: $PRIVATE_SUBNET_ID"
```

出力例

```
Public Subnet: subnet-0faeeeb0000000000
Private Subnet: subnet-011fe340000000000
```

16.3.4.1.2. OIDC 設定の作成

このチュートリアルでは、OIDC 設定を作成するときに自動モードを使用します。また、後で使用できるように、OIDC ID を環境変数として保存します。このコマンドは、ROSA CLI を使用して、クラスターの一意の OIDC 設定を作成します。

- このチュートリアル用の OIDC 設定を作成するには、次のコマンドを実行します。

```
export OIDC_ID=$(rosa create oidc-config --mode auto --managed --yes -o json | jq -r '.id')
```

16.3.4.1.3. 追加の環境変数の作成

- 次のコマンドを実行していくつかの環境変数を設定し、HCP を備えた ROSA クラスターを作成するコマンドを簡単に実行できるようにします。

```
export CLUSTER_NAME=<enter cluster name>
export REGION=<region VPC was created in>
```

ヒント

VPC リージョンを確認するには、**rosa whoami** を実行します。

16.3.4.2. クラスターの作成

このアカウントに ROSA を **初めて** デプロイし、アカウントロールをまだ **作成していない** 場合は、Operator ポリシーを含むアカウント全体のロールとポリシーを作成します。ROSA は AWS Security Token Service (STS) を使用するため、このステップでは、ROSA がアカウントと対話するために必要な AWS IAM ロールとポリシーを作成します。

1. 次のコマンドを実行して、アカウント全体のロールを作成します。

```
rosa create account-roles --mode auto --yes
```

2. 次のコマンドを実行してクラスターを作成します。

```
rosa create cluster --cluster-name $CLUSTER_NAME \
    --subnet-ids ${PUBLIC_SUBNET_ID},${PRIVATE_SUBNET_ID} \
    --hosted-cp \
    --region $REGION \
    --oidc-config-id $OIDC_ID \
    --sts --mode auto --yes
```

10 分ほど経過すると、クラスターが準備完了状態になり、完全に使用できるようになります。選択したリージョン内の 3 つの AWS アベイラビリティーゾーンにまたがるコントロールプレーンがクラスターに作成され、AWS アカウントに 2 つのワーカーノードが作成されます。

16.3.4.3. インストールステータスの確認

1. 次のコマンドのいずれかを実行して、クラスターのステータスを確認します。
   - クラスターのステータスの詳細を表示するには、次のコマンドを実行します。

     ```
     rosa describe cluster --cluster $CLUSTER_NAME
     ```

   - クラスターのステータスの概要を表示するには、次のコマンドを実行します。

     ```
     rosa list clusters
     ```

   - ログの進行状況を監視するには、次を実行します。

     ```
     rosa logs install --cluster $CLUSTER_NAME --watch
     ```

2. 状態が "ready" に変われば、クラスターのインストールは完了です。ワーカーノードがオンラインになるまでにさらに数分かかる場合があります。

16.3.5. チュートリアル: UI 簡易ガイド

このページでは、ユーザーインターフェイス (UI) を使用して ROSA クラスターをデプロイするための最小限のコマンドを説明します。

注記

この単純なデプロイメントはワークショップ環境では問題なく機能しますが、実稼働環境で使用するクラスターはより詳細な方法でデプロイする必要があります。

16.3.5.1. 前提条件

- セットアップのチュートリアルの前提条件を完了している。

16.3.5.2. アカウントロールの作成

AWS アカウントおよび OpenShift の y-stream バージョンごとに次のコマンドを **1回** 実行します。

```
rosa create account-roles --mode auto --yes
```

16.3.5.3. Red Hat OpenShift Cluster Manager ロールの作成

1. 次のコマンドを実行して、AWS アカウントごとに 1 つの OpenShift Cluster Manager ロールを作成します。

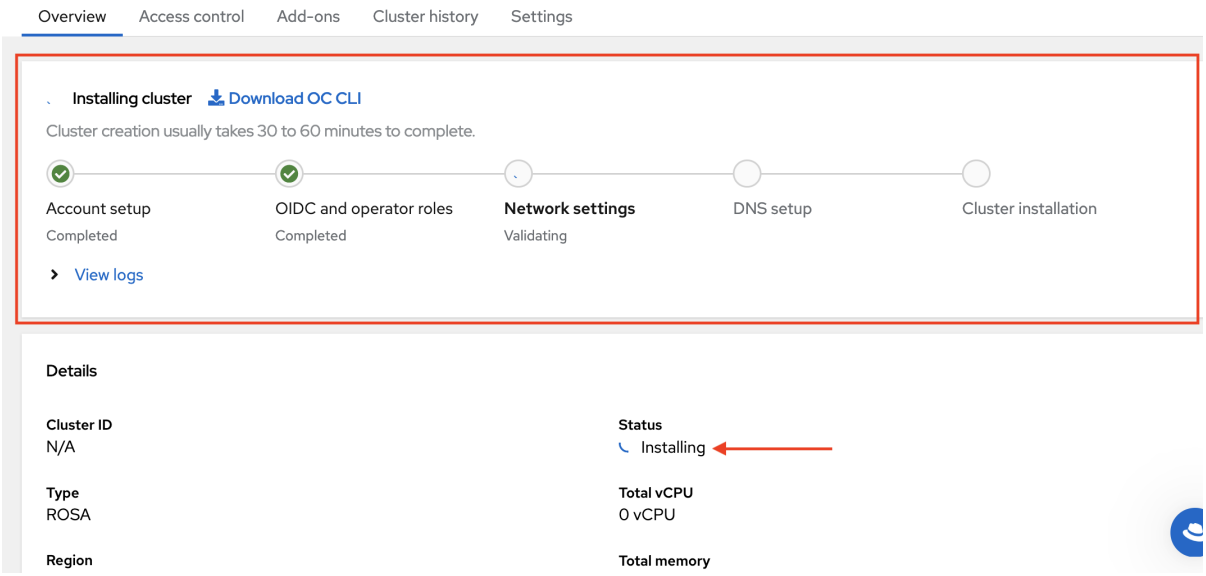
```
rosa create ocm-role --mode auto --admin --yes
```

2. 次のコマンドを実行して、AWS アカウントごとに 1 つの OpenShift Cluster Manager ユーザーロールを作成します。

```
rosa create user-role --mode auto --yes
```

3. Red Hat OpenShift Cluster Manager UI を使用して、AWS アカウント、クラスターオプションを選択し、デプロイを開始します。

4. OpenShift Cluster Manager UI にクラスターのステータスが表示されます。



16.3.6. チュートリアル: UI 詳細ガイド

このチュートリアルでは、Red Hat OpenShift Cluster Manager ユーザーインターフェイス (UI) を使用して Red Hat OpenShift Service on AWS (ROSA) クラスターをデプロイする詳細な手順を説明します。

16.3.6.1. デプロイメントワークフロー

全体的なデプロイメントワークフローは、次のステップに基づいています。

1. アカウント全体のロールとポリシーを作成します。
2. AWS アカウントを Red Hat アカウントに関連付けます。
   a. Red Hat OpenShift Cluster Manager ロールを作成してリンクします。
   b. ユーザーロールを作成してリンクします。
3. クラスターを作成します。

ステップ 1 は、**初めて** AWS アカウントにデプロイする場合にのみ実行する必要があります。ステップ 2 は、UI を **初めて** 使用する場合にのみ実行する必要があります。同じ y-stream バージョンのクラスターを連続して作成する場合は、クラスターを作成するだけで済みます。

16.3.6.2. アカウント全体のロールの作成

注記

以前のデプロイメントのアカウントロールをすでに持っている場合は、この手順をスキップしてください。関連付けられた AWS アカウントを選択すると、UI で既存のロールが検出されます。

このアカウントに ROSA を **初めて** デプロイし、アカウントロールをまだ **作成していない** 場合は、Operator ポリシーを含むアカウント全体のロールとポリシーを作成します。

- ターミナルで次のコマンドを実行して、アカウント全体のロールを作成します。

```
$ rosa create account-roles --mode auto --yes
```

出力例

```
I: Creating roles using 'arn:aws:iam::000000000000:user/rosa-user'
I: Created role 'ManagedOpenShift-ControlPlane-Role' with ARN
'arn:aws:iam::000000000000:role/ManagedOpenShift-ControlPlane-Role'
I: Created role 'ManagedOpenShift-Worker-Role' with ARN
'arn:aws:iam::000000000000:role/ManagedOpenShift-Worker-Role'
I: Created role 'ManagedOpenShift-Support-Role' with ARN
'arn:aws:iam::000000000000:role/ManagedOpenShift-Support-Role'
I: Created role 'ManagedOpenShift-Installer-Role' with ARN
'arn:aws:iam::000000000000:role/ManagedOpenShift-Installer-Role'
I: Created policy with ARN 'arn:aws:iam::000000000000:policy/ManagedOpenShift-openshift-
machine-api-aws-cloud-credentials'
I: Created policy with ARN 'arn:aws:iam::000000000000:policy/ManagedOpenShift-openshift-
cloud-credential-operator-cloud-crede'
I: Created policy with ARN 'arn:aws:iam::000000000000:policy/ManagedOpenShift-openshift-
image-registry-installer-cloud-creden'
I: Created policy with ARN 'arn:aws:iam::000000000000:policy/ManagedOpenShift-openshift-
ingress-operator-cloud-credentials'
I: Created policy with ARN 'arn:aws:iam::000000000000:policy/ManagedOpenShift-openshift-
cluster-csi-drivers-ebs-cloud-credent'
I: To create a cluster with these roles, run the following command:
rosa create cluster --sts
```

16.3.6.3. AWS アカウントと Red Hat アカウントの関連付け

このステップでは、ROSA をデプロイするときに使用する AWS アカウントを OpenShift Cluster Manager に伝えます。

注記

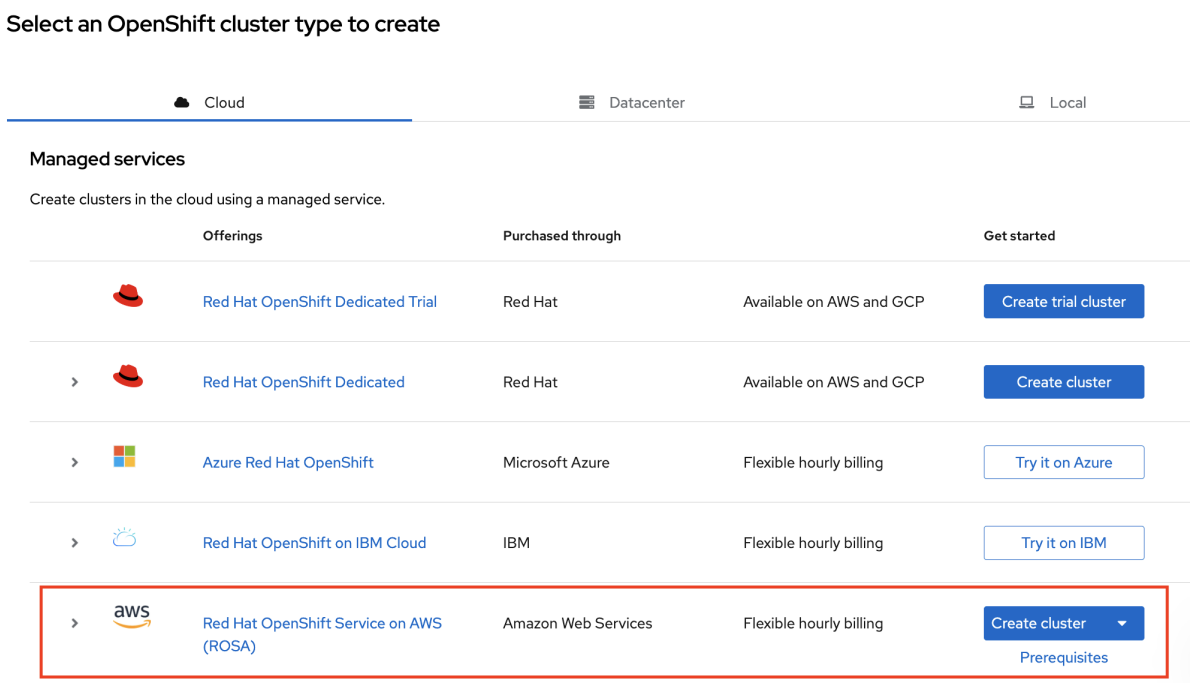
すでに AWS アカウントを関連付けている場合は、この手順をスキップしてください。

1. Red Hat コンソール にログインし、Red Hat アカウントにログインして OpenShift Cluster

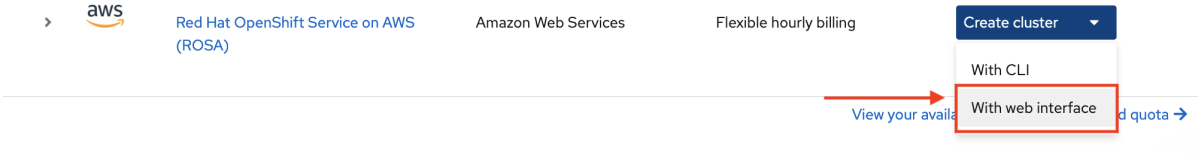
1. Red Hat コンソール にアクセスし、Red Hat アカウントにログインして、OpenShift Cluster Manager を開きます。

2. **Create Cluster** をクリックします。

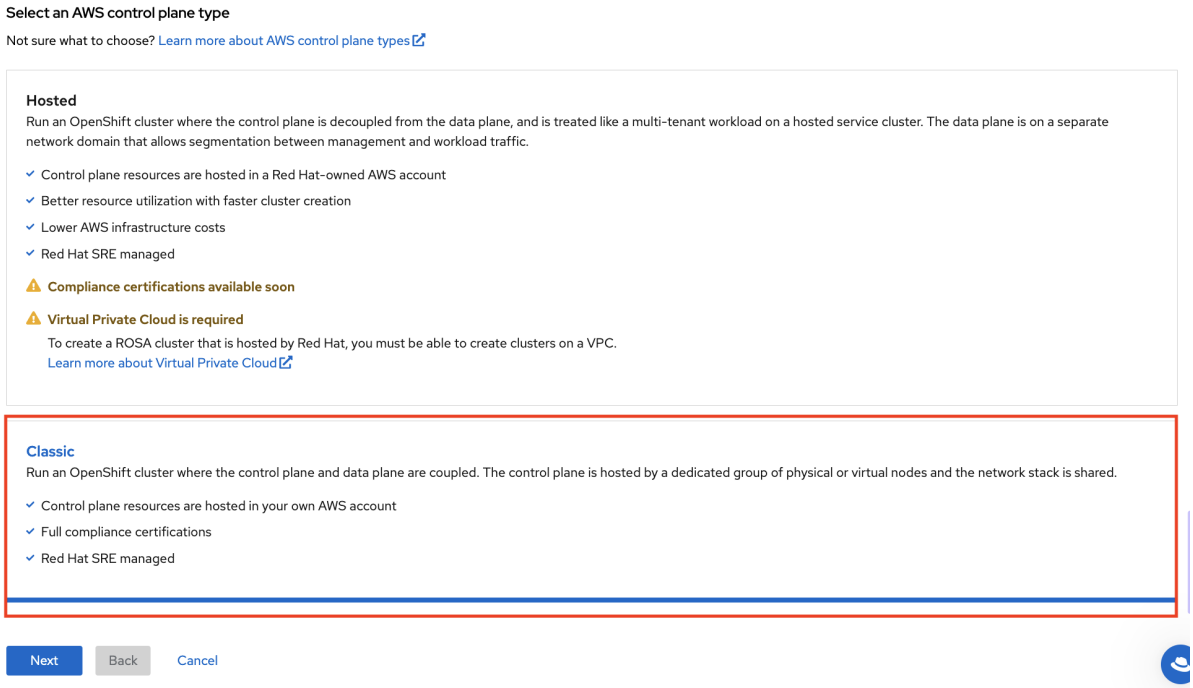
3. Red Hat OpenShift Service on AWS (ROSA) 行まで下にスクロールし、**Create Cluster** をクリックします。



4. ドロップダウンメニューが表示されます。**With web interface** をクリックします。

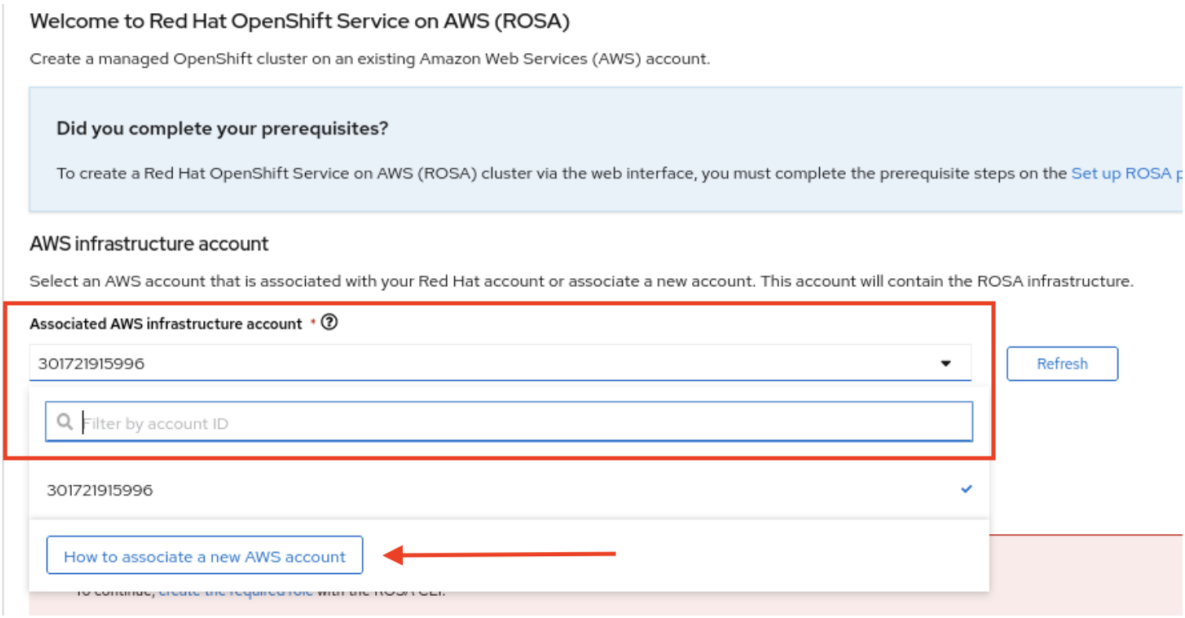


5. "Select an AWS control plane type" の選択で、**Classic** を選選択します。**Next** をクリックします。

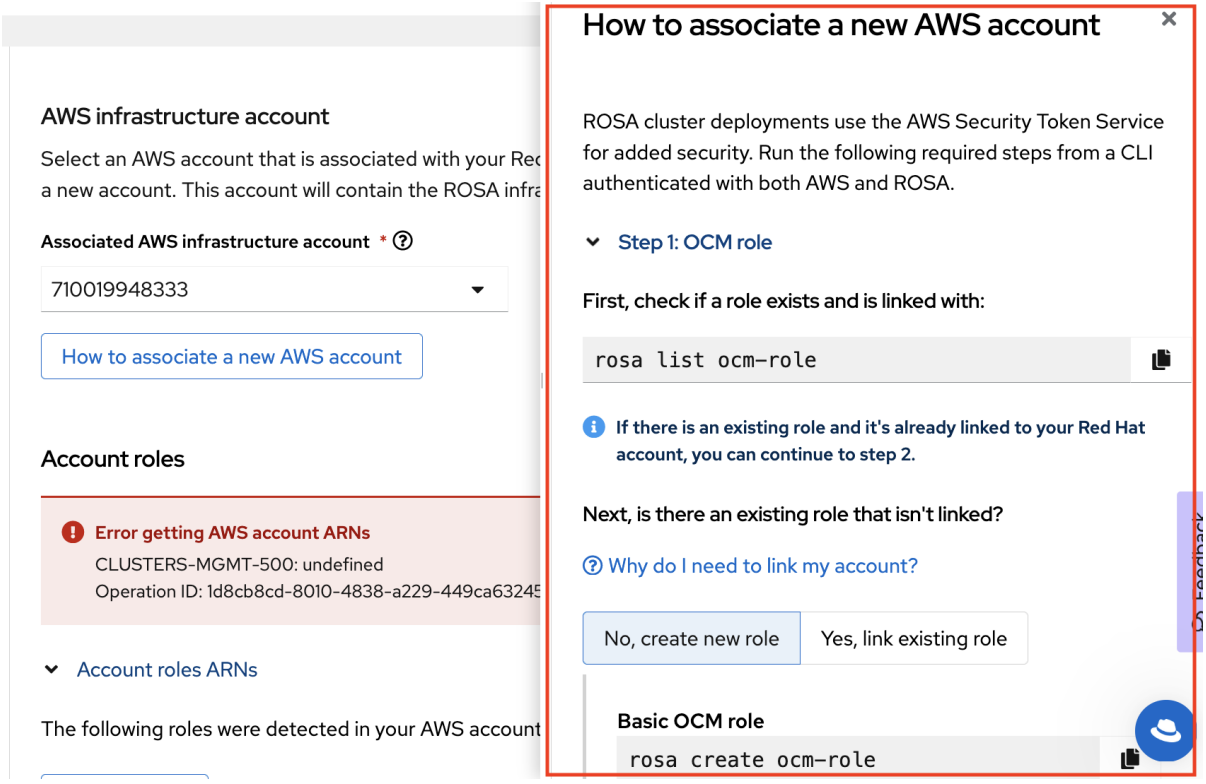


6. **Associated AWS infrastructure account**の下にあるドロップボックスをクリックします。AWS アカウントをまだ関連付けていない場合、ドロップボックスは空の可能性があります。

7. **How to associate a new AWS account**をクリックします。



8. 新しい AWS アカウントを関連付ける手順を示すサイドバーが表示されます。



16.3.6.4. OpenShift Cluster Manager ロールの作成と関連付け

1. 次のコマンドを実行して、OpenShift Cluster Manager ロールが存在するかどうかを確認します。

```
$ rosa list ocm-role
```

2. UI に、権限のレベルが異なる 2 種類の OpenShift Cluster Manager ロールを作成するコマンドが表示されます。

- **Basic OpenShift Cluster Manager role:** OpenShift Cluster Manager にアカウントへの読み取り専用アクセス権を付与し、クラスターを作成する前に ROSA に必要なロールとポリシーが存在するかどうかを確認できるようにします。CLI を使用して、必要なロール、ポリシー、および OIDC プロバイダーを手動で作成する必要があります。

- **Admin OpenShift Cluster Manager role:** ROSA に必要なロール、ポリシー、および OIDC プロバイダーを作成する追加の権限を OpenShift Cluster Manager に付与します。これを使用すると、OpenShift Cluster Manager が必要なリソースを作成できるため、ROSA クラスターのデプロイがより迅速になります。
これらのロールの詳細は、ドキュメントの OpenShift Cluster Manager のロールおよび権限 セクションを参照してください。

  このチュートリアルでは、最も簡単かつ迅速なアプローチである **Admin OpenShift Cluster Manager role** を使用します。

3. サイドバーからコマンドをコピーして Admin OpenShift Cluster Manager role を作成するか、ターミナルに切り替えて次のコマンドを入力します。

```
$ rosa create ocm-role --mode auto --admin --yes
```

このコマンドは、OpenShift Cluster Manager ロールを作成し、それを Red Hat アカウントに関連付けます。

出力例

```
I: Creating ocm role
I: Creating role using 'arn:aws:iam::000000000000:user/rosa-user'
I: Created role 'ManagedOpenShift-OCM-Role-12561000' with ARN
'arn:aws:iam::000000000000:role/ManagedOpenShift-OCM-Role-12561000'
I: Linking OCM role
I: Successfully linked role-arn 'arn:aws:iam::000000000000:role/ManagedOpenShift-OCM-
Role-12561000' with organization account '1MpZfntsZeUdjWHg7XRgP000000'
```

4. **Step 2: User role** をクリックします。

16.3.6.4.1. その他の OpenShift Cluster Manager ロール作成オプション

- **手動モード:** AWS CLI コマンドを自分で実行する場合は、モードを **auto** ではなく **manual** と定義します。CLI に AWS コマンドが出力され、関連する JSON ファイルが現在のディレクトリーに作成されます。
次のコマンドを使用して、手動モードで OpenShift Cluster Manager ロールを作成します。

```
$ rosa create ocm-role --mode manual --admin --yes
```

- **Basic OpenShift Cluster Manager role:** OpenShift Cluster Manager にアカウントへの読み取り専用アクセス権を付与する場合は、Basic OpenShift Cluster Manager role を作成します。この場合、CLI を使用して必要なロール、ポリシー、OIDC プロバイダーを手動で作成する必要があります。
次のコマンドを使用して、Basic OpenShift Cluster Manager role を作成します。

```
$ rosa create ocm-role --mode auto --yes
```

16.3.6.5. OpenShift Cluster Manager ユーザーロールの作成

ユーザーロールのドキュメント で定義されているように、ROSA が AWS ID を検証できるようにユーザーロールを作成する必要があります。このロールには権限を付与せず、インストールプログラムアカウントと OpenShift Cluster Manager ロールリソースの間に信頼関係を構築するためにのみ使用します。

1. 次のコマンドを実行して、ユーザーロールがすでに存在するかどうかを確認します。

```
$ rosa list user-role
```

2. 次のコマンドを実行してユーザーロールを作成し、それを Red Hat アカウントにリンクします。

```
$ rosa create user-role --mode auto --yes
```

出力例

```
I: Creating User role
I: Creating ocm user role using 'arn:aws:iam::000000000000:user/rosa-user'
I: Created role 'ManagedOpenShift-User-rosa-user-Role' with ARN
'arn:aws:iam::000000000000:role/ManagedOpenShift-User-rosa-user-Role'
I: Linking User role
I: Successfully linked role ARN 'arn:aws:iam::000000000000:role/ManagedOpenShift-User-
rosa-user-Role' with account '1rbOQez0z5j1YoIlnhcXY000000'
```

注記

前述のように、AWS CLI コマンドを自分で実行する場合は、**--mode manual** を定義します。CLI に AWS コマンドが出力され、関連する JSON ファイルが現在のディレクトリーに作成されます。必ずロールをリンクしてください。

3. **Step 3: Account roles** をクリックします。

16.3.6.6. アカウントロールの作成

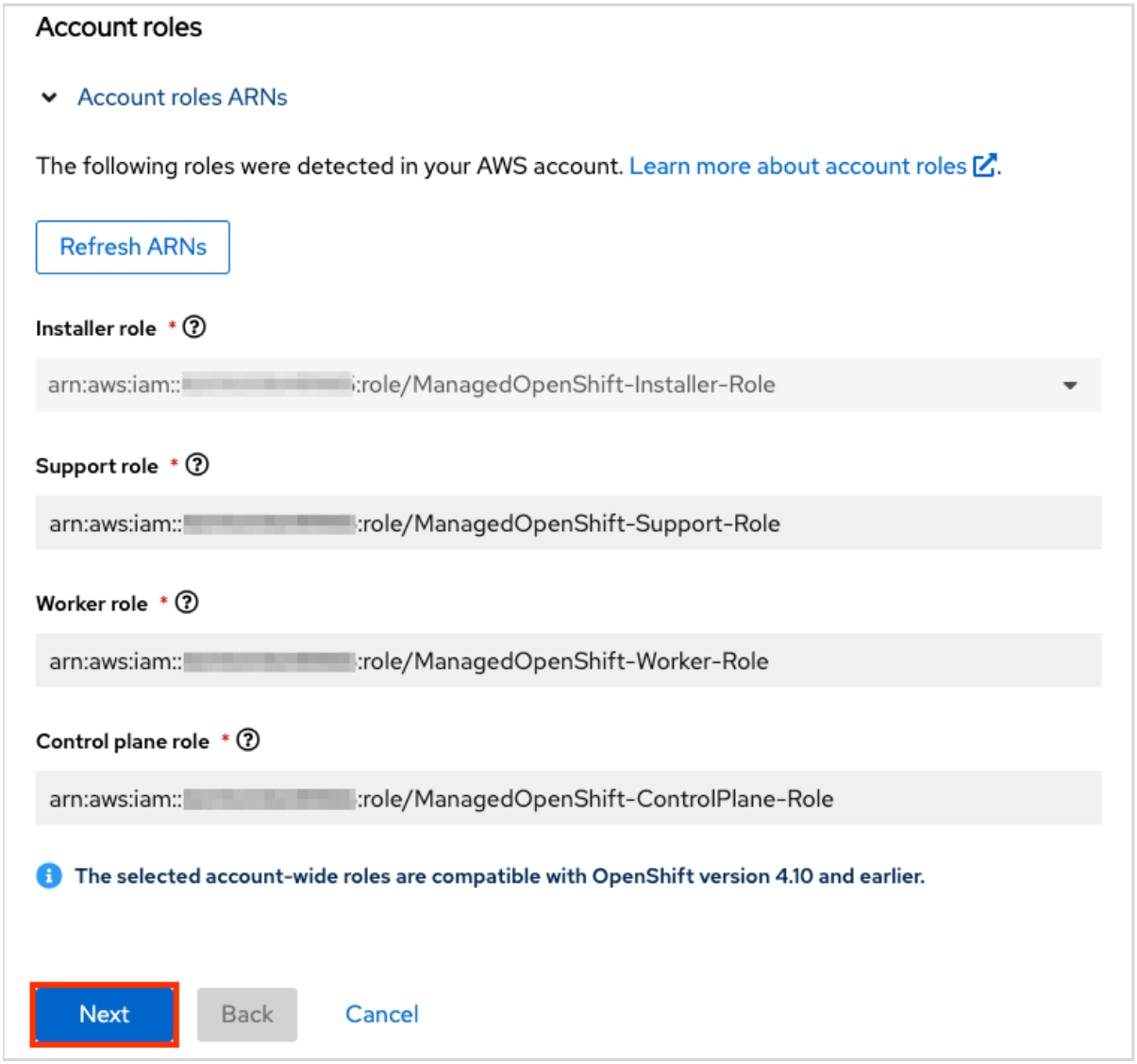
1. 次のコマンドを実行して、アカウントのロールを作成します。

```
$ rosa create account-roles --mode auto
```

2. **OK** をクリックしてサイドバーを閉じます。

16.3.6.7. アカウントの関連付けが成功したことの確認

1. **Associated AWS infrastructure account** ドロップダウンメニューに AWS アカウントが表示されるはずです。自分のアカウントが表示されれば、アカウントの関連付けは成功しています。
2. アカウントを選択します。
3. アカウントのロール ARN が下に入力されて表示されます。



4. **Next** をクリックします。

16.3.6.8. クラスターの作成

1. このチュートリアルでは、以下を選択してください。

クラスター設定

- Cluster name: **<名前を選択\>**
- Version: **<最新バージョンを選択\>**
- Region: **<リージョンを選択\>**
- Availability: **Single zone**
- Enable user workload monitoring: **オンのまま**
- Enable additional etcd encryption: **オフのまま**
- Encrypt persistent volumes with customer keys: **オフのまま**

2. **Next** をクリックします。

3. マシンプールのデフォルト設定はオンのままにします。

デフォルトのマシンプール設定

- Compute node instance type: **m5.xlarge - 4 vCPU 16 GiB RAM**
- Enable autoscaling: **オフ**
- Compute node count: **2**
- ノードラベルは空白のままにする

4. **Next** をクリックします。

16.3.6.8.1. ネットワーク

1. 設定はすべてデフォルト値のままにします。
2. **Next** をクリックします。
3. CIDR 範囲のデフォルト値はすべてそのままにしておきます。
4. **Next** をクリックします。

16.3.6.8.2. クラスターのロールおよびポリシー

このチュートリアルでは、**Auto** を選択したままにしておきます。これにより、クラスターのデプロイメントプロセスが容易かつ迅速になります。

注記

前に **Basic OpenShift Cluster Manager role** を選択した場合は、手動モードのみを使用できます。Operator ロールと OIDC プロバイダーを手動で作成する必要があります。「クラスターの更新」セクションを完了してクラスターの作成を開始したら、後述の「Basic OpenShift Cluster Manager role」セクションを参照してください。

16.3.6.8.3. クラスターの更新

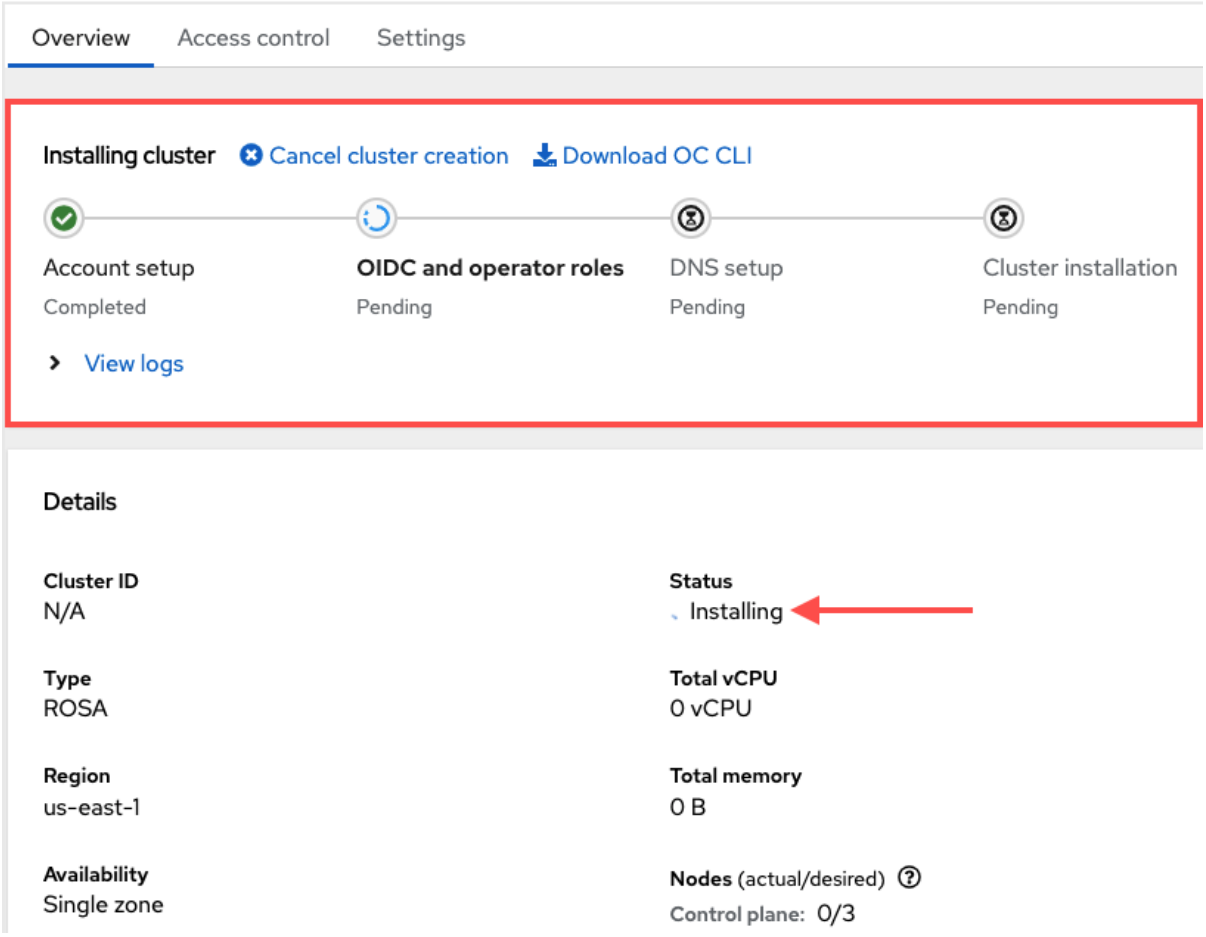
- このセクションでは、すべてのオプションをデフォルトのままにしておきます。

16.3.6.8.4. クラスターの確認と作成

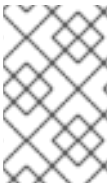
1. クラスター設定の内容を確認します。
2. **Create cluster** をクリックします。

16.3.6.8.5. インストールの進行状況の監視

- 現在のページに留まり、インストールの進行状況を監視します。所要時間は約 40 分です。



16.3.6.9. Basic OpenShift Cluster Manager Role



注記

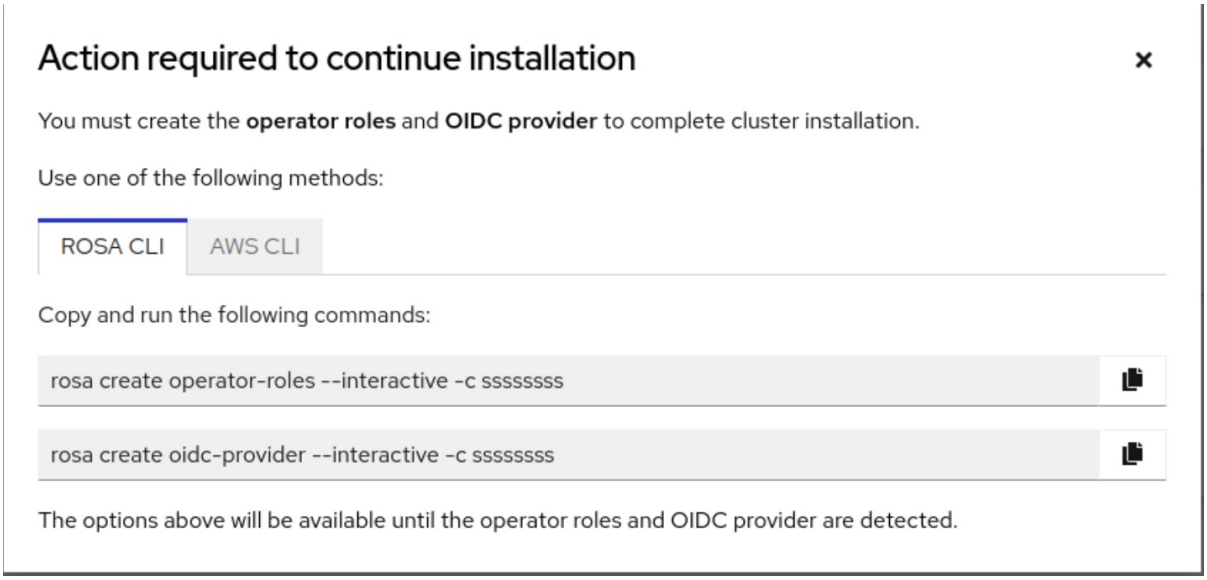
前述の手順に従って **Admin OpenShift Cluster Manager role** を作成した場合は、このセクション全体を **無視してください**。OpenShift Cluster Manager によってリソースが作成されます。

前に **Basic OpenShift Cluster Manager role** を作成した場合は、クラスターのインストールを続行する前に、さらに 2 つの要素を手動で作成する必要があります。

- Operator ロール
- OIDC プロバイダー

16.3.6.9.1. Operator ロールの作成

1. ポップアップウィンドウに、実行するコマンドが表示されます。



2. ターミナルのウィンドウからコマンドを実行して、対話モードを起動します。または、簡素化のために、次のコマンドを実行して Operator ロールを作成します。

```
$ rosa create operator-roles --mode auto --cluster <cluster-name> --yes
```

出力例

```
I: Creating roles using 'arn:aws:iam::000000000000:user/rosauser'
I: Created role 'rosacluster-b736-openshift-ingress-operator-cloud-credentials' with ARN 'arn:aws:iam::000000000000:role/rosacluster-b736-openshift-ingress-operator-cloud-credentials'
I: Created role 'rosacluster-b736-openshift-cluster-csi-drivers-ebs-cloud-credent' with ARN 'arn:aws:iam::000000000000:role/rosacluster-b736-openshift-cluster-csi-drivers-ebs-cloud-credent'
I: Created role 'rosacluster-b736-openshift-cloud-network-config-controller-cloud' with ARN 'arn:aws:iam::000000000000:role/rosacluster-b736-openshift-cloud-network-config-controller-cloud'
I: Created role 'rosacluster-b736-openshift-machine-api-aws-cloud-credentials' with ARN 'arn:aws:iam::000000000000:role/rosacluster-b736-openshift-machine-api-aws-cloud-credentials'
I: Created role 'rosacluster-b736-openshift-cloud-credential-operator-cloud-crede' with ARN 'arn:aws:iam::000000000000:role/rosacluster-b736-openshift-cloud-credential-operator-cloud-crede'
I: Created role 'rosacluster-b736-openshift-image-registry-installer-cloud-creden' with ARN 'arn:aws:iam::000000000000:role/rosacluster-b736-openshift-image-registry-installer-cloud-creden'
```

16.3.6.9.2. OIDC プロバイダーの作成

- ターミナルで次のコマンドを実行して OIDC プロバイダーを作成します。

```
$ rosa create oidc-provider --mode auto --cluster <cluster-name> --yes
```

出力例

```
I: Creating OIDC provider using 'arn:aws:iam::000000000000:user/rosauser'
I: Created OIDC provider with ARN 'arn:aws:iam::000000000000:oidc-provider/rh-oidc.s3.us-east-1.amazonaws.com/1tt4kvrr2kha2rgs8gjfvf0000000000'
```

16.4. チュートリアル: 管理ユーザーの作成

管理 (admin) ユーザーを作成すると、クラスターにすばやくアクセスできるようになります。管理ユーザーを作成するには、次の手順に従います。

注記

管理ユーザーは、このチュートリアル設定では問題なく機能します。実際のデプロイメントでは、正式なアイデンティティープロバイダー を使用してクラスターにアクセスし、ユーザーに管理権限を付与してください。

1. 次のコマンドを実行して管理ユーザーを作成します。

```
rosa create admin --cluster=<cluster-name>
```

出力例

```
W: It is recommended to add an identity provider to login to this cluster. See 'rosa create idp --help' for more information.
I: Admin account has been added to cluster 'my-rosa-cluster'. It may take up to a minute for the account to become active.
I: To login, run the following command:
oc login https://api.my-rosa-cluster.abcd.p1.openshiftapps.com:6443 \
--username cluster-admin \
--password FWGYL-2mkJI-00000-00000
```

2. 前のステップで返されたログインコマンドをコピーし、ターミナルに貼り付けます。これにより、CLI を使用してクラスターにログインし、クラスターの使用を開始できるようになります。

```
$ oc login https://api.my-rosa-cluster.abcd.p1.openshiftapps.com:6443 \
>    --username cluster-admin \
>    --password FWGYL-2mkJI-00000-00000
```

出力例

```
Login successful.

You have access to 79 projects, the list has been suppressed. You can list all projects with ' projects'

Using project "default".
```

3. 管理ユーザーとしてログインしていることを確認するには、次のコマンドのいずれかを実行します。

- オプション 1:

```
$ oc whoami
```

出力例

```
cluster-admin
```

- オプション 2:

```
oc get all -n openshift-apiserver
```

このコマンドをエラーなしで実行できるのは管理ユーザーのみです。

4. これでクラスターを管理ユーザーとして使用できるようになりました。このチュートリアルではこれで十分です。実際のデプロイメントでは、アイデンティティープロバイダーをセットアップすることを強く推奨します。これについては、次のチュートリアル で説明します。

16.5. チュートリアル: アイデンティティープロバイダーのセットアップ

クラスターにログインするには、アイデンティティープロバイダー (IDP) をセットアップします。このチュートリアルでは、IDP の例として GitHub を使用します。ROSA でサポートされる IDP の完全なリストを参照してください。

- すべての IDP オプションを表示するには、次のコマンドを実行します。

```
rosa create idp --help
```

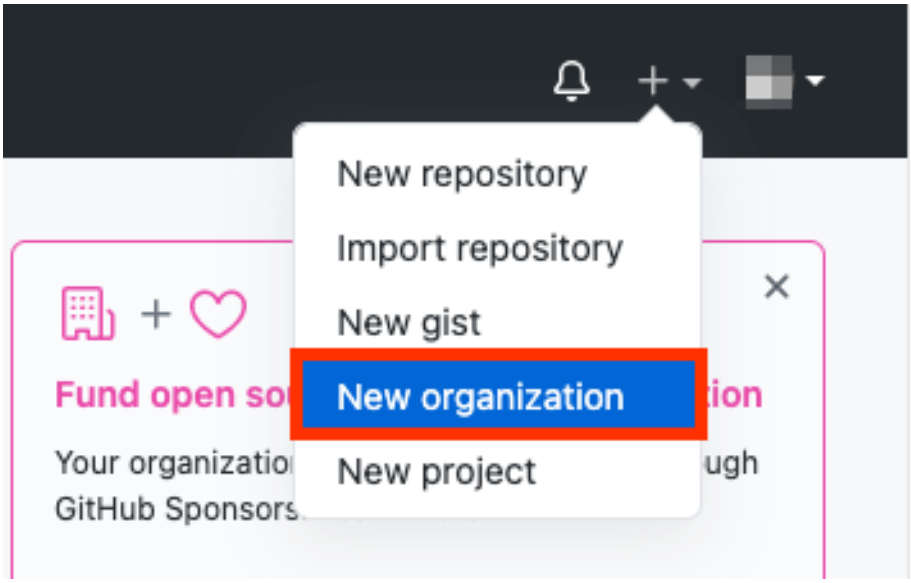
16.5.1. GitHub を使用した IDP のセットアップ

1. GitHub アカウントにログインします。
2. 自分が管理者となる新しい GitHub 組織を作成します。

ヒント

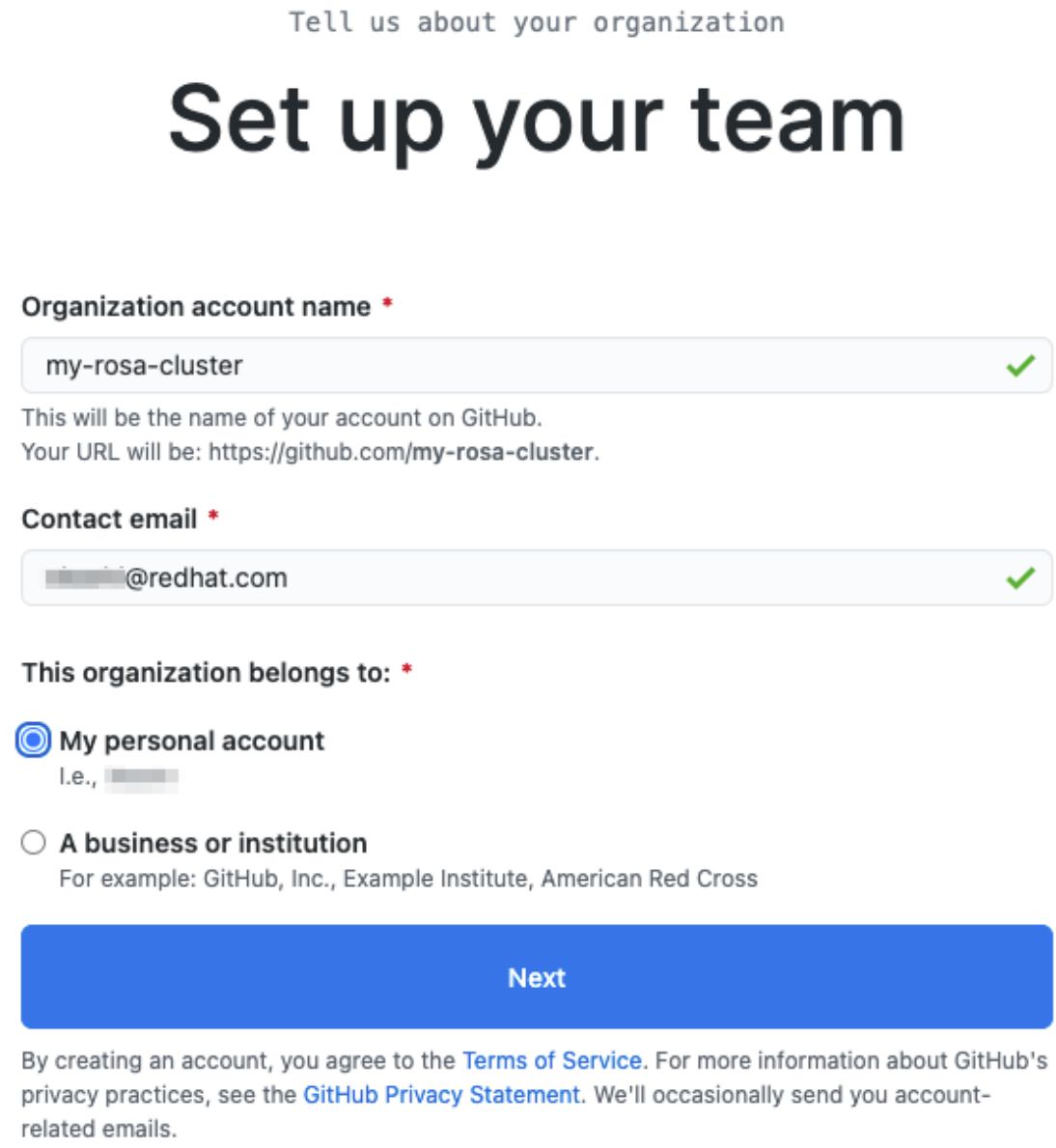
すでに既存の組織の管理者であり、その組織を使用する場合は、ステップ 9 に進みます。

+ アイコンをクリックし、**New Organization** をクリックします。



3. 状況に最も適したプランを選択するか、**Join for free** をクリックします。

4. 組織のアカウント名、メールアドレス、および個人アカウントかビジネスアカウントかを入力します。**Next** をクリックします。



5. **オプション:** 他のユーザーの GitHub ID を追加して、ROSA クラスターへの追加のアクセス権を付与します。後で追加することもできます。

6. **Complete Setup** をクリックします。

7. **オプション:** 次のページで、要求される情報を入力します。

8. **Submit** をクリックします。

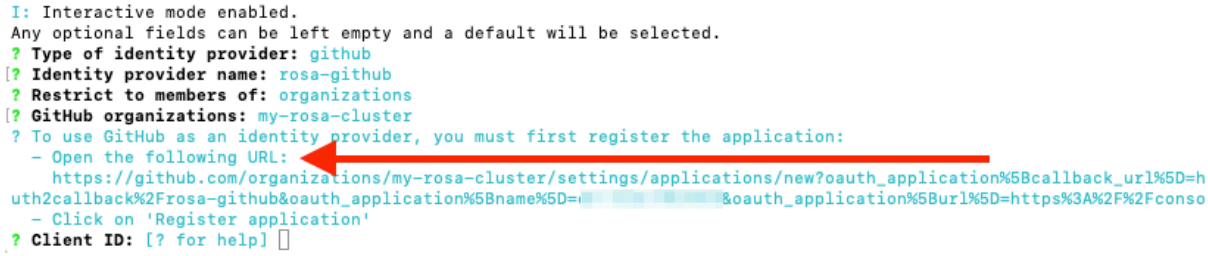
9. ターミナルに戻り、次のコマンドを入力して GitHub IDP を設定します。

```
rosa create idp --cluster=<cluster name> --interactive
```

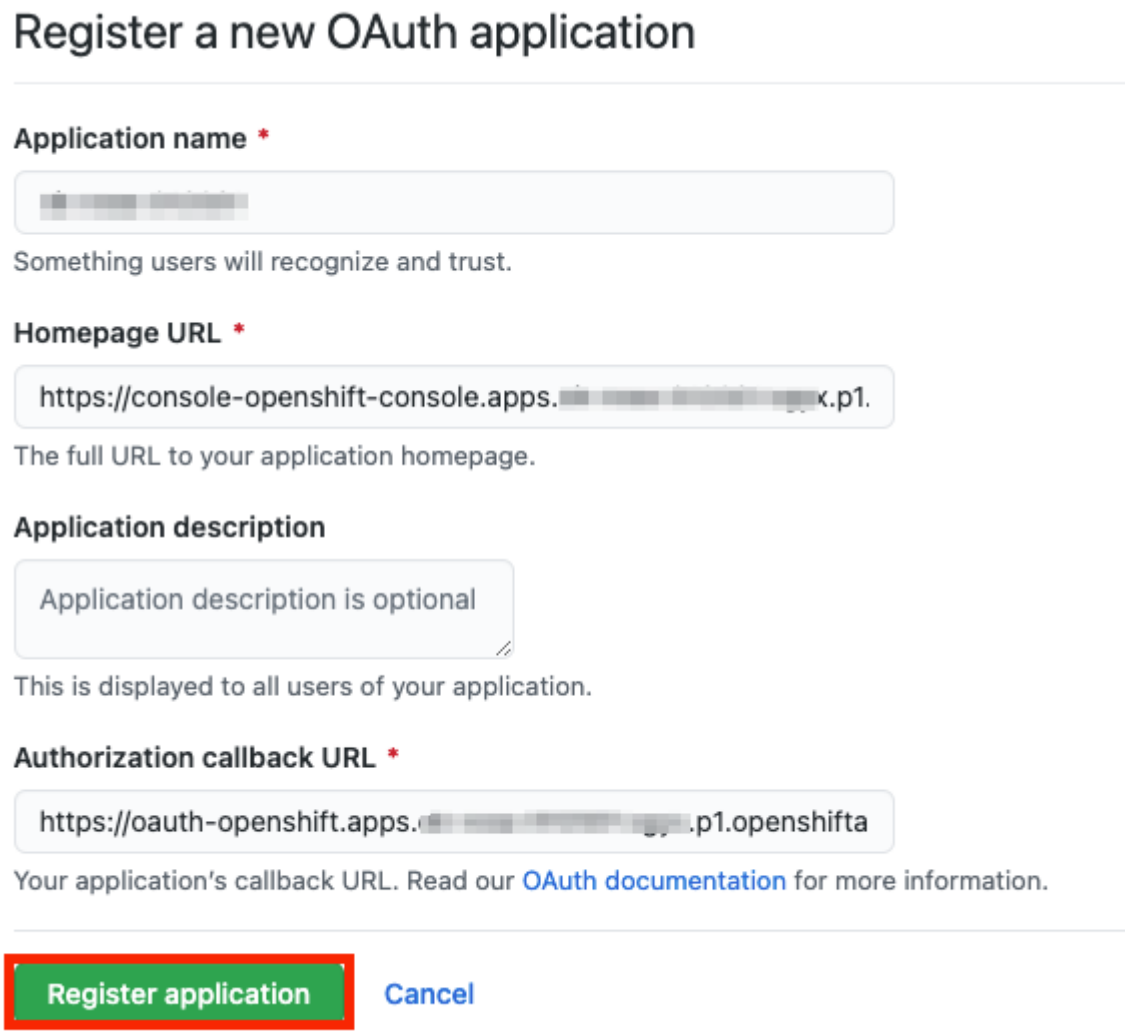
10. 以下の値を設定します。

```
Type of identity provider: github
Identity Provider Name: <IDP-name>
Restrict to members of: organizations
GitHub organizations: <organization-account-name>
```

11. CLI にリンクが表示されます。リンクをコピーしてブラウザーに貼り付け、**Enter** を押します。これにより、このアプリケーションを OAuth に登録するために必要な情報が入力されます。情報を変更する必要はありません。



12. **Register application** をクリックします。

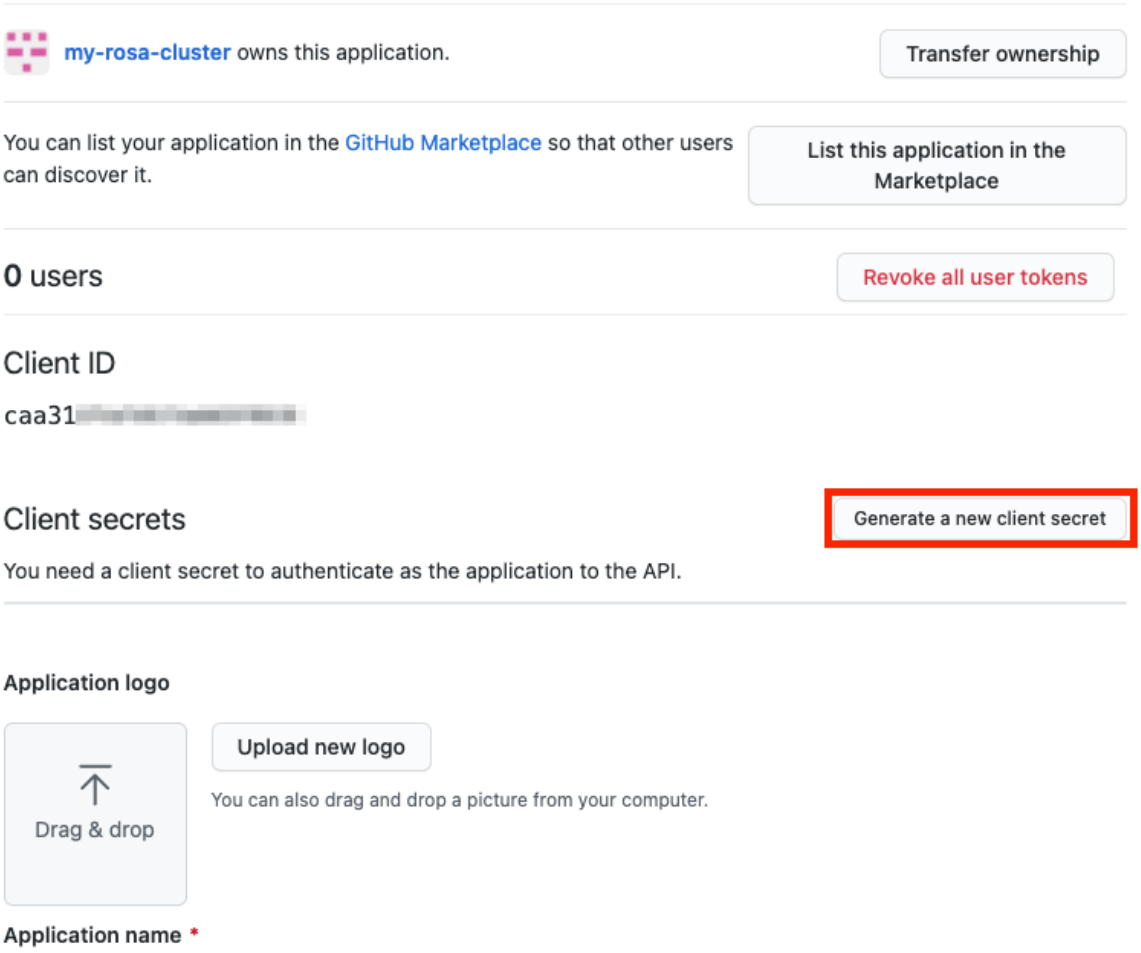


13. 次のページに **Client ID** が表示されます。ID をコピーし、 **Client ID** を要求するターミナルに ID を貼り付けます。

注記

タブは閉じないでください。

14. CLI で **Client Secret** を求められます。ブラウザーに戻り、 **Generate a new client secret** をクリックします。



15. シークレットが生成されます。シークレットは二度と表示されないため、コピーしてください。

16. シークレットをターミナルに貼り付けて **Enter** を押します。

17. **GitHub Enterprise Hostname** は空白のままにします。

18. **claim** を選択します。

19. IDP が作成され、設定がクラスターに反映されるまで、約 1 分間待ちます。

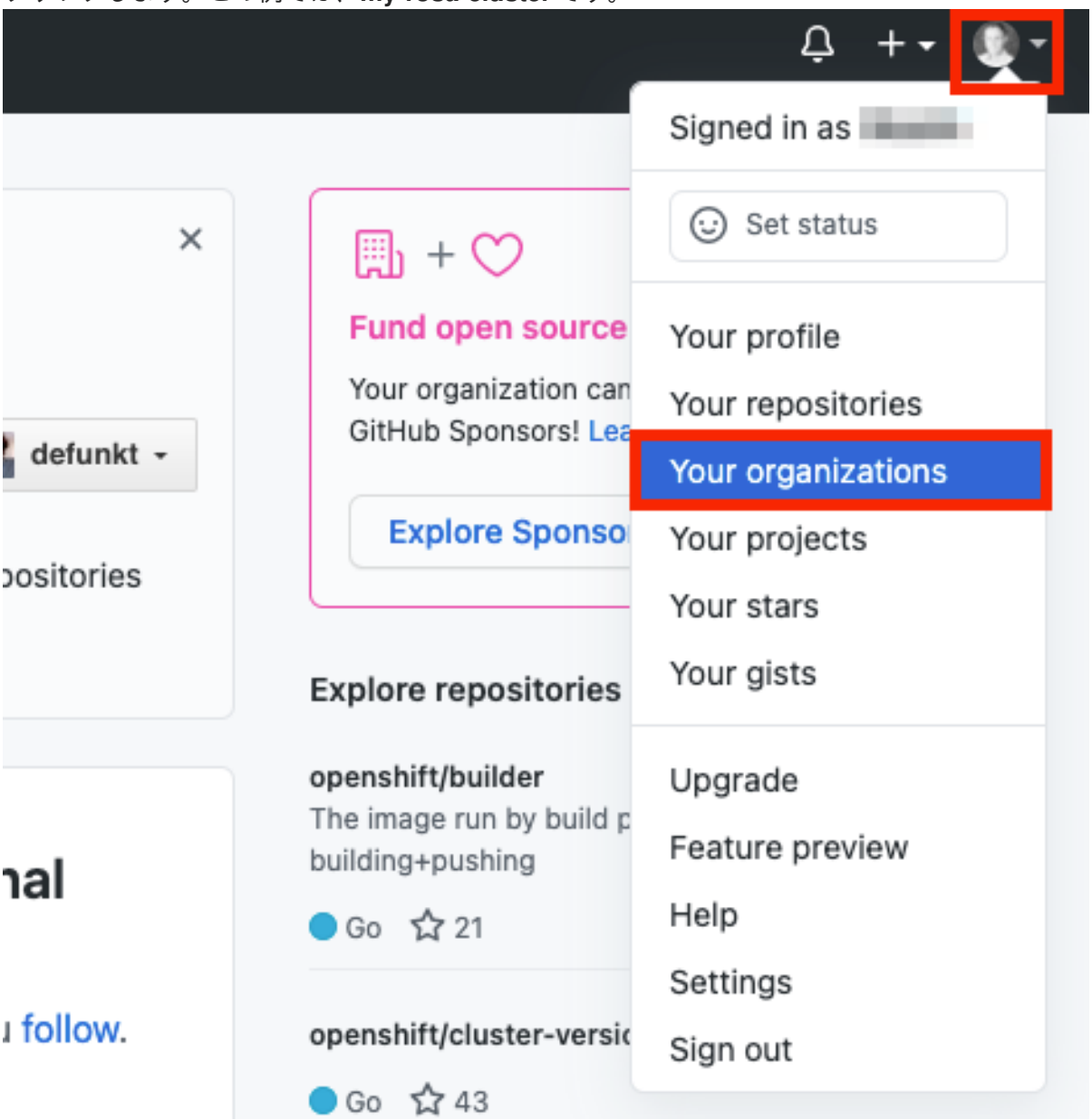
```
I: Interactive mode enabled.
Any optional fields can be left empty and a default will be selected.
? Type of identity provider: github
? Identity provider name: rosa-github
? Restrict to members of: organizations
? GitHub organizations: my-rosa-cluster
? To use GitHub as an identity provider, you must first register the application:
  - Open the following URL:
    https://github.com/organizations/my-rosa-cluster/settings/applications/new?oauth_application%5Bcallback_url%5D=
https%3A%2F%2Foauth-openshift.apps.[redacted].p1.openshiftapps.com%2Foauth2callback%2Frosa-github&oauth_ap
plication%5Bname%5D=[redacted]&oauth_application%5Burl%5D=https%3A%2F%2Fconsole-openshift-console.apps.[redacted]
[redacted].p1.openshiftapps.com
  - Click on 'Register application'
? Client ID: caa31[redacted]
? Client Secret: [? for help] ****************************************
? GitHub Enterprise Hostname (optional):
? Mapping method: claim
I: Configuring IDP for cluster '[redacted]'
I: Identity Provider 'rosa-github' has been created.
   It will take up to 1 minute for this configuration to be enabled.
   To add cluster administrators, see 'rosa create user --help'.
   To login into the console, open https://console-openshift-console.apps.[redacted].p1.openshiftapps.com
and click on rosa-github.
```

20. 返されたリンクをコピーし、ブラウザーに貼り付けます。新しい IDP を、選択した名前で利用できるはずです。IDP をクリックし、GitHub 認証情報を使用してクラスターにアクセスします。

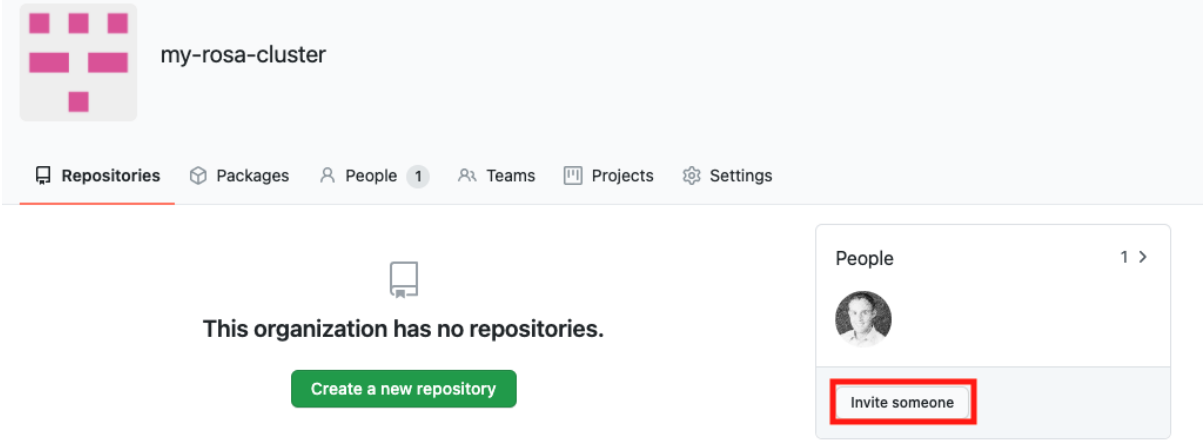
16.5.2. 他のユーザーにクラスターへのアクセス権を付与する

他のクラスターユーザーにアクセス権を付与するには、その GitHub ユーザー ID を、このクラスターに使用する GitHub 組織に追加する必要があります。

1. GitHub で、**Your organizations** ページに移動します。
2. **profile icon** をクリックし、**Your organizations** をクリックします。次に、**<自分の組織名>** をクリックします。この例では、**my-rosa-cluster** です。



3. **Invite someone** をクリックします。



4. 新しいユーザーの GitHub ID を入力し、正しいユーザーを選択して、**Invite** をクリックします。

5. 新しいユーザーが招待を受け入れると、コンソールのリンクと GitHub 認証情報を使用して ROSA クラスターにログインできるようになります。

16.6. チュートリアル: 管理権限の付与

管理 (admin) 権限は、クラスターに追加したユーザーに自動的には付与されません。特定のユーザーに管理レベルの権限を付与する場合は、各ユーザーに手動で権限を付与する必要があります。ROSA コマンドラインインターフェイス (CLI) または Red Hat OpenShift Cluster Manager Web ユーザーインターフェイス (UI) のいずれかから管理権限を付与できます。

Red Hat は 2 種類の管理権限を提供します。

- **cluster-admin**: **cluster-admin** 権限は、管理ユーザーにクラスター内のすべての権限を付与します。

- **dedicated-admin**: **dedicated-admin** 権限を持つ管理ユーザーは、ほとんどの管理タスクを完了できますが、クラスターの破損を防ぐために一定の制限があります。権限の昇格が必要な場合は、**dedicated-admin** を使用することを推奨します。

管理権限の詳細は、クラスターの管理 に関するドキュメントを参照してください。

16.6.1. ROSA CLI の使用

1. クラスターを作成したユーザーが、次のコマンドのいずれかを実行して管理権限を付与します。

   - **cluster-admin** の場合:

     ```
     $ rosa grant user cluster-admin --user <idp_user_name> --cluster=<cluster-name>
     ```

   - **dedicated-admin** の場合:

     ```
     $ rosa grant user dedicated-admin --user <idp_user_name> --cluster=<cluster-name>
     ```

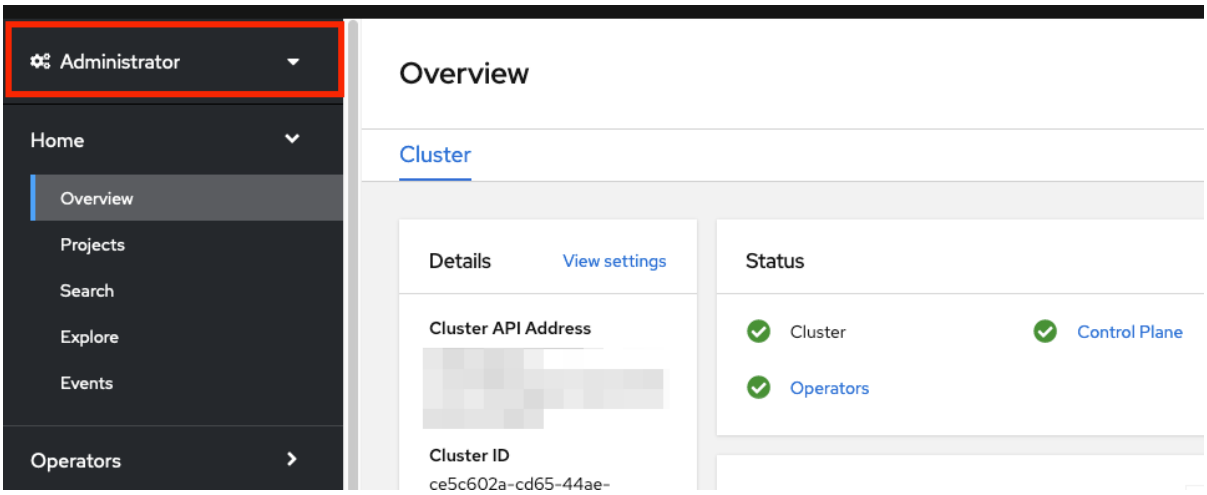
2. 次のコマンドを実行して、管理権限が追加されたことを確認します。

   ```
   $ rosa list users --cluster=<cluster-name>
   ```

出力例

```
$ rosa list users --cluster=my-rosa-cluster
ID                GROUPS
<idp_user_name>   cluster-admins
```

3. 現在 Red Hat コンソールにログインしている場合は、コンソールからログアウトし、クラスターに再度ログインして、"Administrator パネル" がある新しいパースペクティブを表示します。シークレットウィンドウまたはプライベートウィンドウが必要になる場合があります。

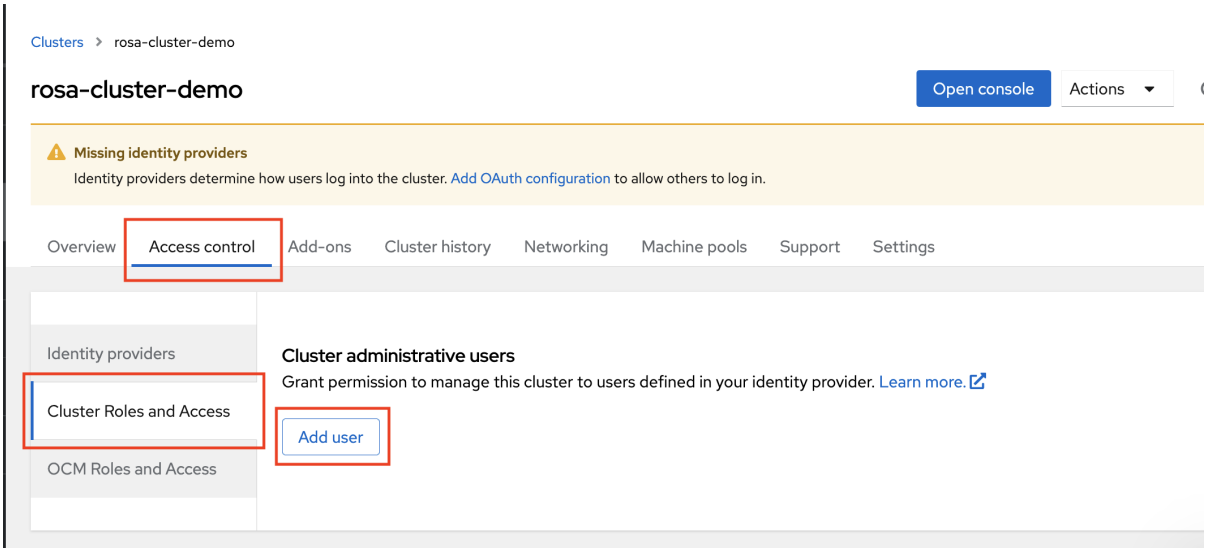


4. 次のコマンドを実行して、アカウントに管理権限が追加されたことをテストすることもできます。このコマンドをエラーなしで実行できるのは、**cluster-admin** ユーザーのみです。

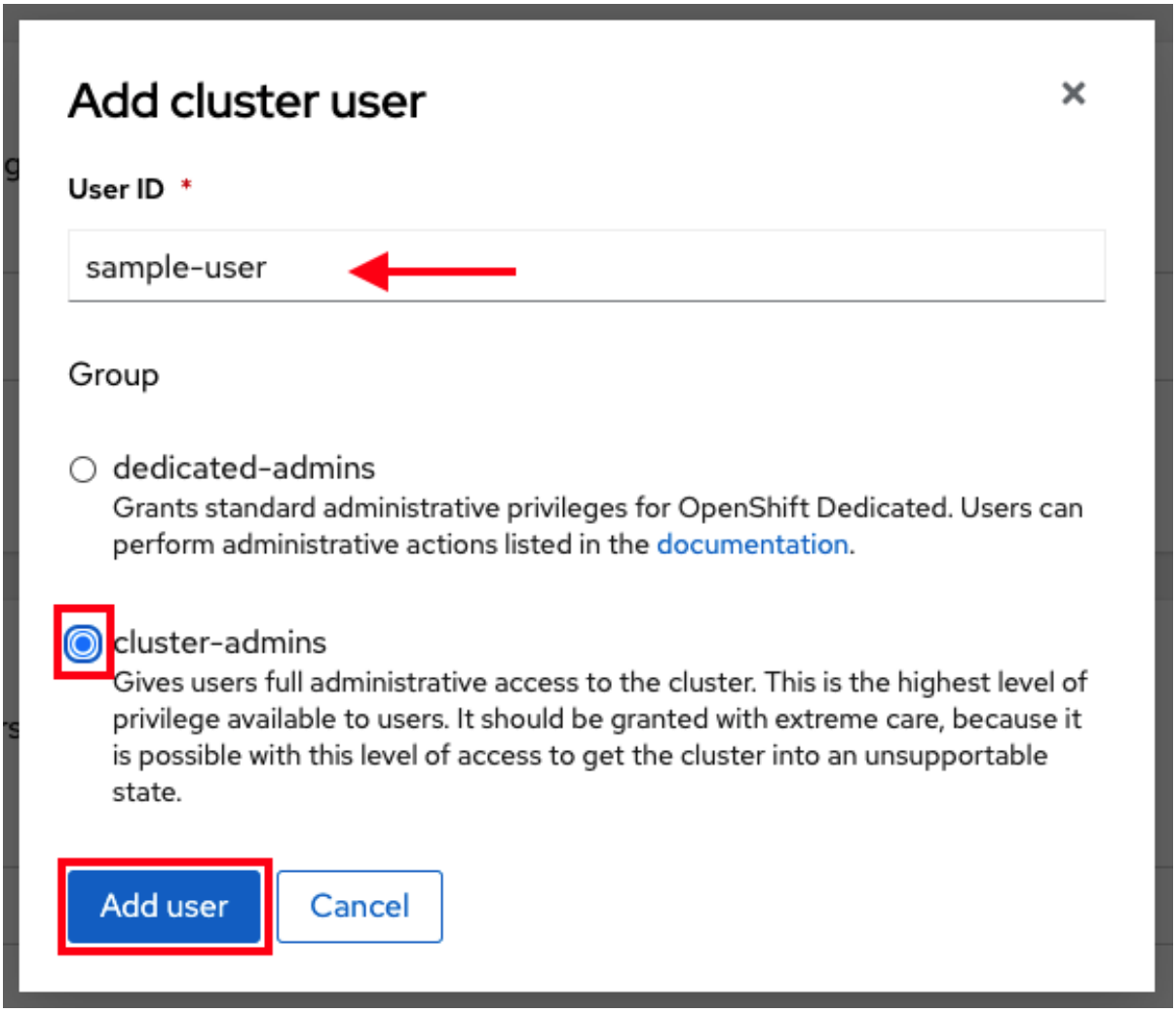
```
$ oc get all -n openshift-apiserver
```

16.6.2. Red Hat OpenShift Cluster Manager UI の使用

1. Red Hat OpenShift Cluster Manager コンソール にログインします。
2. クラスターを選択します。
3. **Access Control** タブをクリックします。
4. サイドバーの **Cluster roles and Access** タブをクリックします。
5. **Add user** をクリックします。



6. ポップアップ画面でユーザー ID を入力します。
7. ユーザーに **cluster-admin** 権限を付与するか、**dedicated-admin** 権限を付与するかを選択します。



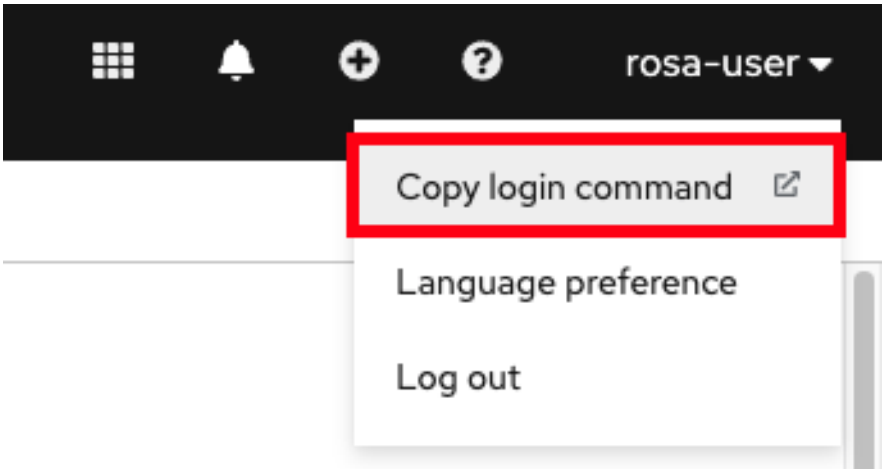
16.7. チュートリアル: クラスターへのアクセス

コマンドラインインターフェイス (CLI) またはコンソールユーザーインターフェイス (UI) を使用してクラスターに接続できます。

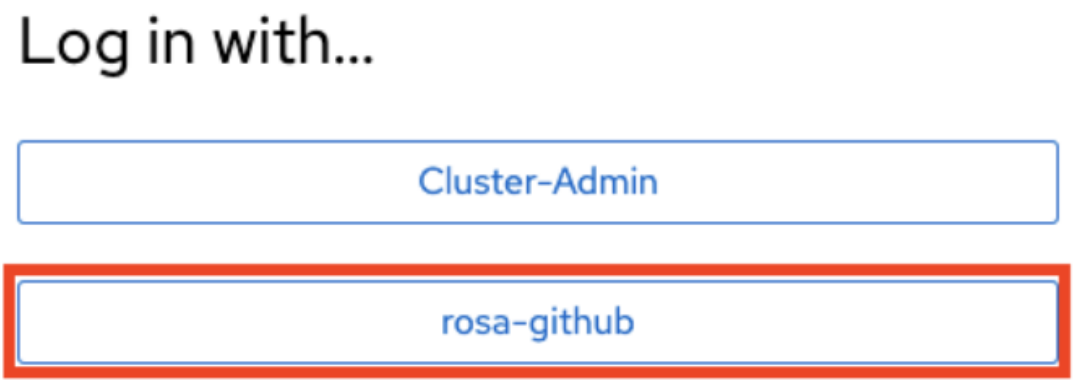
16.7.1. CLI を使用したクラスターへのアクセス

CLI を使用してクラスターにアクセスするには、**oc** CLI がインストールされている必要があります。チュートリアルの手順を実行してきた場合は、**oc** CLI がすでにインストールされています。

1. Red Hat コンソール にログインします。
2. 右上隅にあるユーザー名をクリックします。
3. **Copy Login Command** をクリックします。



4. アイデンティティープロバイダー (IDP) を選択できる新しいタブが開きます。使用する IDP をクリックします。たとえば、"rosa-github" です。



5. 新しいタブが開きます。**Display token** をクリックします。
6. ターミナルで次のコマンドを実行します。

```
$ oc login --token=sha256~GBAfS4JQ0t1UTKYHbWAK6OUWGUkdMGz000000000000 --
server=https://api.my-rosa-cluster.abcd.p1.openshiftapps.com:6443
```

出力例

```
Logged into "https://api.my-rosa-cluster.abcd.p1.openshiftapps.com:6443" as "rosa-user"
using the token provided.

You have access to 79 projects, the list has been suppressed. You can list all projects with '
projects'

Using project "default".
```

7. 次のコマンドを実行して、ログインしていることを確認します。

```
$ oc whoami
```

出力例

```
rosa-user
```

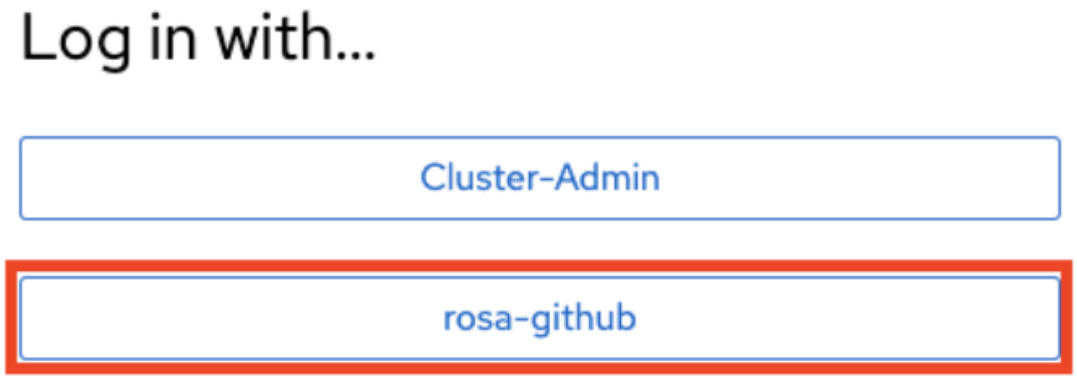
8. これで、クラスターにアクセスできます。

16.7.2. Web コンソールを使用したクラスターへのアクセス

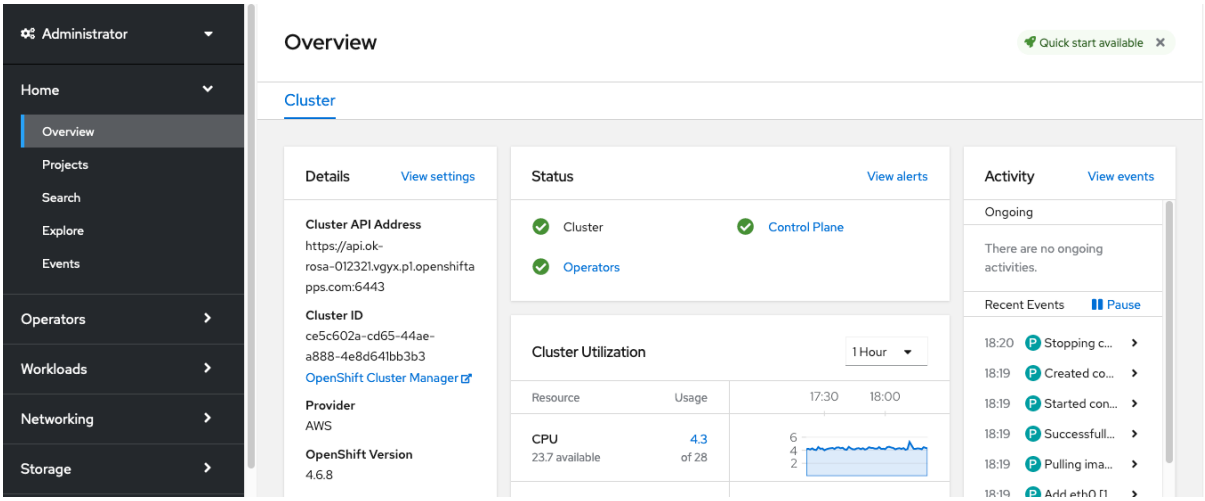
1. Red Hat コンソール にログインします。
   a. コンソールの URL を取得するには、次のコマンドを実行します。

```
rosa describe cluster -c <cluster-name> | grep Console
```

2. IDP をクリックします。たとえば、"rosa-github" です。



3. ユーザー認証情報を入力します。
4. ログインが完了します。チュートリアルの手順を実行してきた場合は、cluster-admin となり、**Administrator** パネルがある Web コンソールが表示されるはずです。



16.8. チュートリアル: ワーカーノードの管理

Red Hat OpenShift Service on AWS (ROSA) では、ワーカーノードの変更はマシンプールを使用して実行します。マシンプールを使用すると、ユーザーは多数のマシンを 1 つのエンティティーとして管理できます。すべての ROSA クラスターに、クラスターの作成時に作成されるデフォルトのマシンプールがあります。詳細は、マシンプール のドキュメントを参照してください。

16.8.1. マシンセットの作成

マシンプールは、コマンドラインインターフェイス (CLI) またはユーザーインターフェイス (UI) のいずれかを使用して作成できます。

16.8.1.1. CLI を使用したマシンプールの作成

1. 以下のコマンドを実行します。

```
rosa create machinepool --cluster=<cluster-name> --name=<machinepool-name> --replicas=
<number-nodes>
```

入力の例

```
 $ rosa create machinepool --cluster=my-rosa-cluster --name=new-mp
 --replicas=2
```

出力例

```
I: Machine pool 'new-mp' created successfully on cluster 'my-rosa-cluster'
I: To view all machine pools, run 'rosa list machinepools -c my-rosa-cluster'
```

2. **オプション:** 次のコマンドを実行して、新しいマシンプール内の特定のノードにノードラベルまたはテイントを追加します。

```
rosa create machinepool --cluster=<cluster-name> --name=<machinepool-name> --replicas=
<number-nodes> --labels=`<key=pair>`
```

入力の例

```
$ rosa create machinepool --cluster=my-rosa-cluster --name=db-nodes-mp --replicas=2 --
labels='app=db','tier=backend'
```

出力例

```
I: Machine pool 'db-nodes-mp' created successfully on cluster 'my-rosa-cluster'
```

これにより、1 つのユニットとして管理できる追加の 2 つのノードが作成されます。また、表示されるラベルがこのノードに割り当てられます。

3. 次のコマンドを実行して、マシンプールの作成と割り当てられたラベルを確認します。

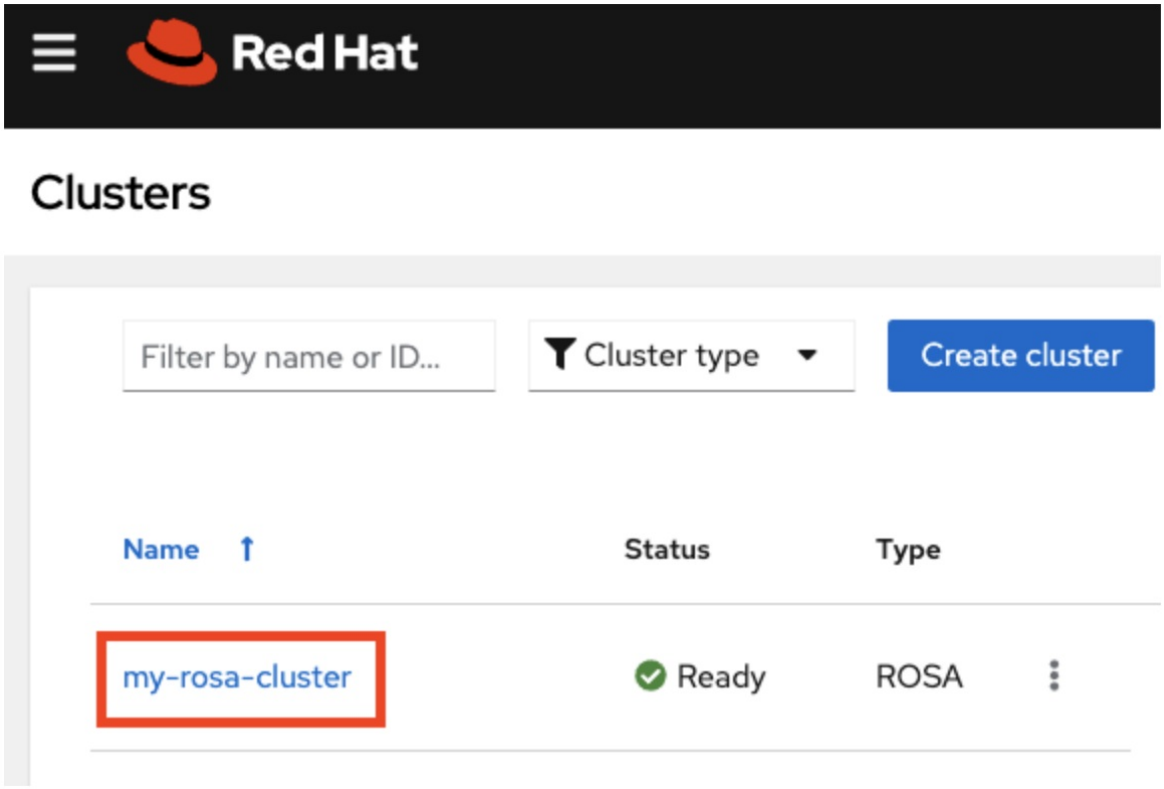
```
rosa list machinepools --cluster=<cluster-name>
```

出力例

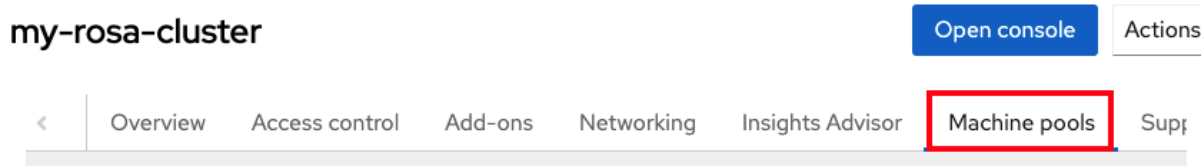
```
ID          AUTOSCALING  REPLICAS  INSTANCE TYPE  LABELS          TAINTS
AVAILABILITY ZONES
Default     No           2         m5.xlarge                      us-east-1a
```

16.8.1.2. UI を使用したマシンプールの作成

1. Red Hat コンソール にログインし、クラスターをクリックします。



2. **Machine pools** をクリックします。

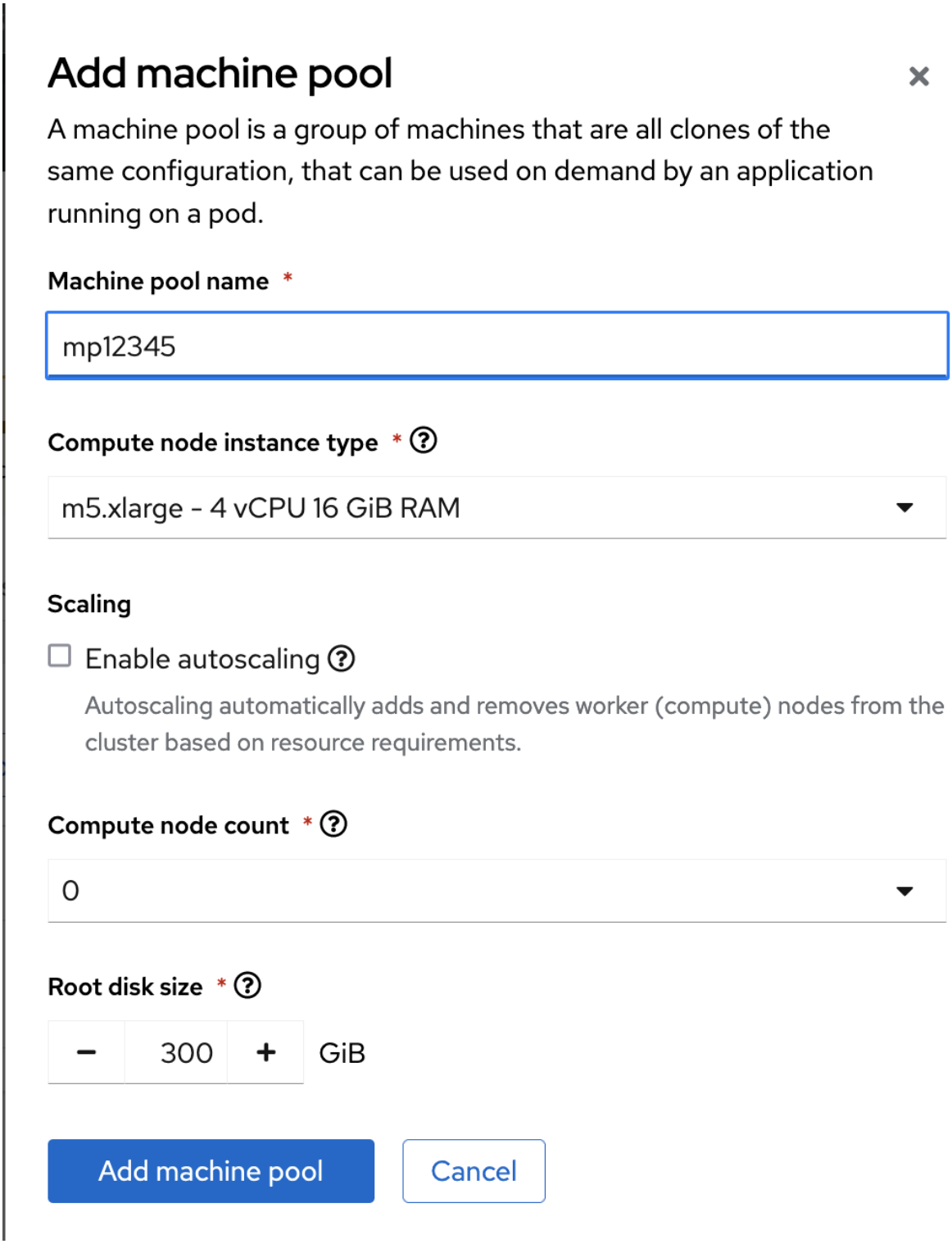


3. **Add machine pool** をクリックします。

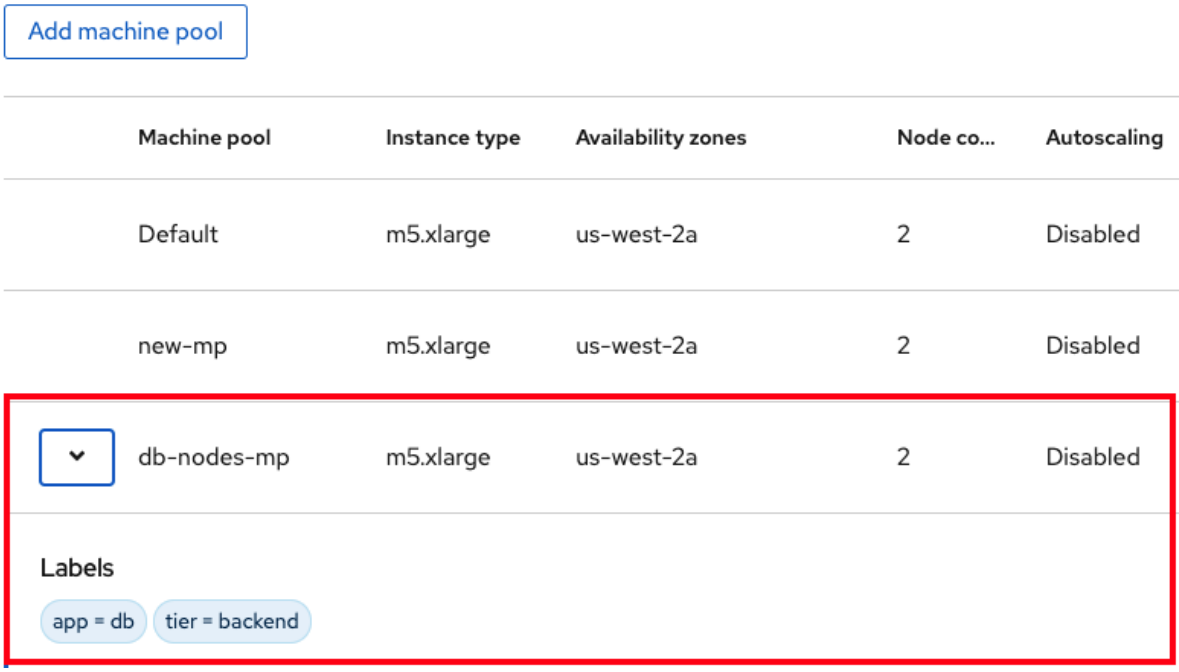
4. 必要な設定を入力します。

ヒント

また、**Edit node labels and taints** セクションを展開して、ノードラベルとテイントをマシンプール内のノードに追加することもできます。



5. 作成した新しいマシンプールが表示されます。



16.8.2. ワーカーノードのスケーリング

マシンプールを編集して、その特定のマシンプール内のワーカーノードの数をスケーリングします。CLI または UI を使用してワーカーノードをスケーリングできます。

16.8.2.1. CLI を使用したワーカーノードのスケーリング

1. 次のコマンドを実行して、各クラスターで作成されたデフォルトのマシンプールを確認します。

```
rosa list machinepools --cluster=<cluster-name>
```

出力例

```
ID          AUTOSCALING  REPLICAS  INSTANCE TYPE  LABELS          TAINTS
AVAILABILITY ZONES
Default     No           2         m5.xlarge                      us-east-1a
```

2. デフォルトのマシンプールを異なるノード数にスケールアウトするには、次のコマンドを実行します。

```
rosa edit machinepool --cluster=<cluster-name> --replicas=<number-nodes> <machinepool-name>
```

入力の例

```
rosa edit machinepool --cluster=my-rosa-cluster --replicas 3 Default
```

3. 次のコマンドを実行して、マシンプールがスケーリングされたことを確認します。

```
rosa describe cluster --cluster=<cluster-name> | grep Compute
```

入力の例

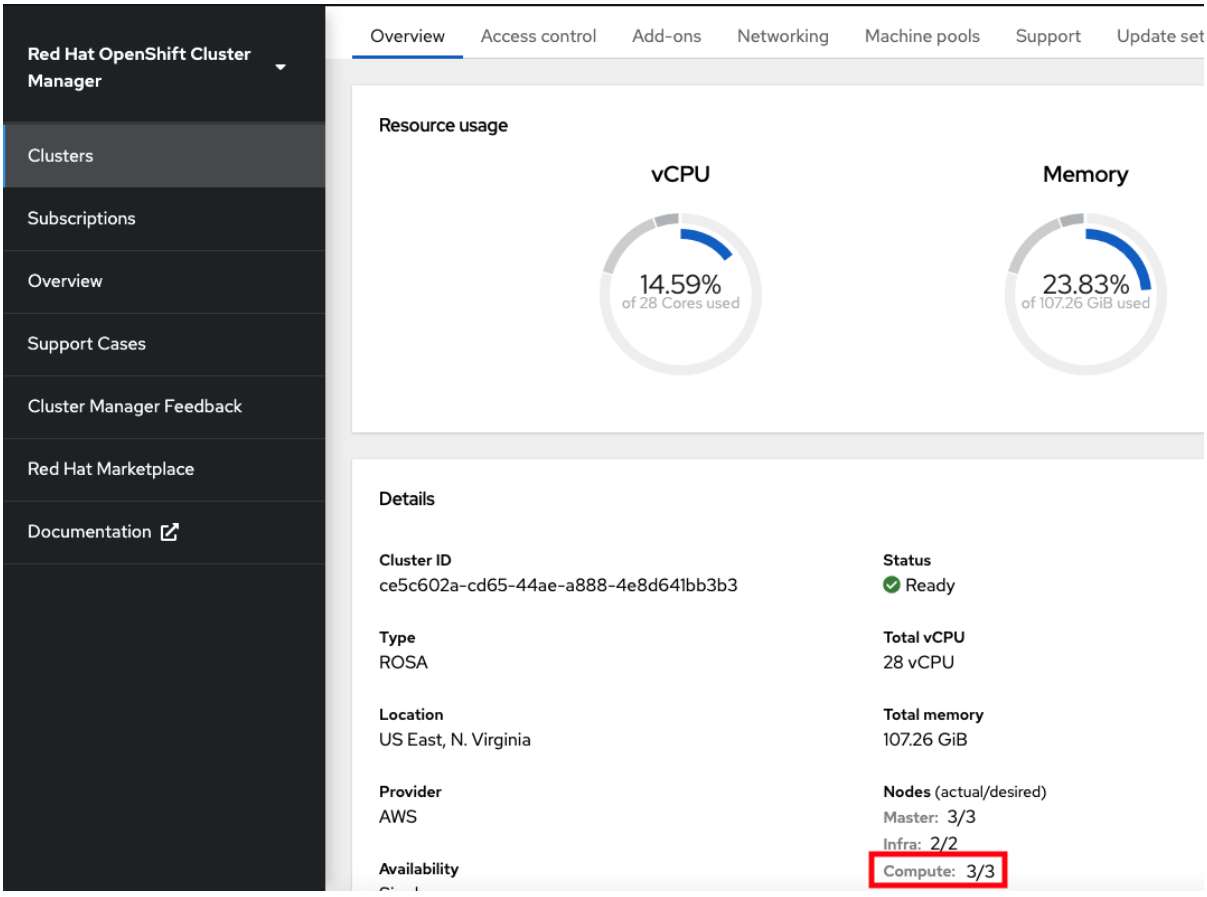
```
$ rosa describe cluster --cluster=my-rosa-cluster | grep Compute
```

出力例

```
- Compute:                3 (m5.xlarge)
```

16.8.2.2. UI を使用したワーカーノードのスケーリング

1. 編集するマシンプールの右側にある 3 つの点をクリックします。
2. **Edit** をクリックします。
3. 必要なノード数を入力し、**Save** をクリックします。
4. クラスターを選択し、**Overview** タブをクリックします。**Compute listing** までスクロールして、クラスターがスケーリングされたことを確認します。Compute listing はスケーリングされたノードと同じであるはずです。たとえば、3/3 のようになります。



16.8.2.3. ノードラベルの追加

1. 次のコマンドを使用してノードラベルを追加します。

```
rosa edit machinepool --cluster=<cluster-name> --replicas=<number-nodes> --
labels='key=value' <machinepool-name>
```

入力の例

```
rosa edit machinepool --cluster=my-rosa-cluster --replicas=2 --labels 'foo=bar','baz=one'
new-mp
```

これにより、新しいマシンプールに 2 つのラベルが追加されます。

重要

このコマンドは、すべてのマシンプール設定を新しく定義した設定に置き換えます。別のラベルを追加し、**かつ** 古いラベルを保持する場合は、新しいラベルと既存のラベルの両方を指定する必要があります。指定しないと、既存のすべてのラベルが追加するラベルに置き換えられます。同様に、ラベルを削除する場合は、削除するラベルを除いて必要なラベルを指定し、コマンドを実行します。

16.8.3. ノードタイプの混在

新しいマシンプールを使用して、同じクラスター内で異なるワーカーノードマシンタイプを混在させることもできます。マシンプールの作成後にマシンプールのノードタイプを変更することはできません。しかし、**--instance-type** フラグを追加することで、異なるノードを持つ新しいマシンプールを作成できます。

1. たとえば、データベースノードを別のノードタイプに変更するには、次のコマンドを実行します。

```
rosa create machinepool --cluster=<cluster-name> --name=<mp-name> --replicas=<number-nodes> --labels='<key=pair>' --instance-type=<type>
```

入力の例

```
rosa create machinepool --cluster=my-rosa-cluster --name=db-nodes-large-mp --replicas=2 --labels='app=db','tier=backend' --instance-type=m5.2xlarge
```

2. 利用可能なすべてのインスタンスタイプ を表示するには、次のコマンドを実行します。

```
rosa list instance-types
```

3. ステップバイステップで変更するには、**--interactive** フラグを使用します。

```
rosa create machinepool -c <cluster-name> --interactive
```

```
[? Machine pool name: large-nodes-pool
[? Enable autoscaling (optional): No
[? Replicas: 3
 ? Instance type:  [Use arrows to move, type to filter, ? for more help]
 > m5.xlarge
   r5.xlarge
   r5.2xlarge
   m5.2xlarge
   c5.2xlarge
   r5.4xlarge
   m5.4xlarge
```

4. 次のコマンドを実行してマシンプールをリストし、より大きな新しいインスタンスタイプを確認します。

```
rosa list machinepools -c <cluster-name>
```

```
ID                  AUTOSCALING  REPLICAS  INSTANCE TYPE  LABELS                  TAINTS    AVAILABILITY ZONES
default             No           3         m5.xlarge                                        us-east-1a
db-nodes-large--mp  No           2         m5.2xlarge     app=db, tier=backend              us-east-1a
db-nodes-mp         No_          2         m5.xlarge      app=db, tier=backend              us-east-1a
```

16.9. チュートリアル: 自動スケーリング

クラスターオートスケーラー は、Pod リソースに基づいてクラスターにワーカーノードを追加または削除します。

クラスターオートスケーラーは、次の場合にクラスターのサイズを増やします。

- リソースが不足しているため、Pod を現在のノードでスケジュールできない場合。
- デプロイメントのニーズを満たすために、別のノードが必要な場合。

クラスターオートスケーラーは、指定の制限を超えてクラスターリソースを拡大することはありません。

クラスターオートスケーラーは、次の場合にクラスターのサイズを減らします。

- 一部のノードが長期間続けて必要とされない場合。たとえば、ノードのリソース使用量が低く、そのノードの重要な Pod がすべて他のノードに収まる場合です。

16.9.1. CLI を使用した既存マシンプールの自動スケーリングの有効化

注記

クラスターの自動スケーリングは、クラスターの作成時、および新しいマシンプールを作成するときに、**--enable-autoscaling** オプションを使用して有効にできます。

1. 自動スケーリングは、マシンプールの可用性に基づいて設定されます。どのマシンプールが自動スケーリングに使用できるかを確認するには、次のコマンドを実行します。

```
$ rosa list machinepools -c <cluster-name>
```

出力例

```
ID       AUTOSCALING  REPLICAS  INSTANCE TYPE  LABELS    TAINTS
AVAILABILITY ZONES
Default  No           2         m5.xlarge                  us-east-1a
```

2. 次のコマンドを実行して、利用可能なマシンプールに自動スケーリングを追加します。

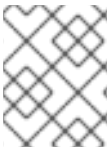
```
$ rosa edit machinepool -c <cluster-name> --enable-autoscaling <machinepool-name> --min-replicas=<num> --max-replicas=<num>
```

入力の例

```
$ rosa edit machinepool -c my-rosa-cluster --enable-autoscaling Default --min-replicas=2 --max-replicas=4
```

上記のコマンドは、リソースに応じて 2 - 4 ノードの間でスケーリングするワーカーノードのオートスケーラーを作成します。

16.9.2. UI を使用した既存マシンプールの自動スケーリングの有効化



注記

マシンプールの作成時に **Enable autoscaling** チェックボックスをオンにすることで、クラスターの作成時にクラスターの自動スケーリングを有効にできます。

1. **Machine pools** タブに移動し、右側の 3 つの点をクリックします。
2. **Scale** をクリックし、**Enable autoscaling** をクリックします。
3. 次のコマンドを実行して、自動スケーリングが追加されたことを確認します。

```
$ rosa list machinepools -c <cluster-name>
```

出力例

```
ID        AUTOSCALING  REPLICAS  INSTANCE TYPE  LABELS     TAINTS
AVAILABILITY ZONES
Default    Yes         2-4       m5.xlarge                 us-east-1a
```

16.10. チュートリアル: クラスターのアップグレード

Red Hat OpenShift Service on AWS (ROSA) では、クラスターのアップグレードはすべてマネージドサービスの一環として実行されます。ユーザーがコマンドを実行したり、クラスターに変更を加えたりする必要はありません。アップグレードは都合の良い時刻にスケジュールできます。

クラスターのアップグレードをスケジュールする方法には、次のものがあります。

- **コマンドラインインターフェイス (CLI) を使用した手動アップグレード**: 1 回限りの即時アップグレードを開始するか、将来の日時に 1 回限りのアップグレードをスケジュールします。
- **Red Hat OpenShift Cluster Manager ユーザーインターフェイス (UI) を使用した手動アップグレード**: 1 回限りの即時アップグレードを開始するか、将来の日時に 1 回限りのアップグレードをスケジュールします。
- **自動アップグレード**: 手動でスケジュールを設定することなく、新しいバージョンが利用可能になるたびに、定期的な y-stream アップグレードのアップグレード時間を設定します。マイナーバージョンは手動でスケジュールする必要があります。

クラスターのアップグレードの詳細は、次のコマンドを実行してください。

```
$ rosa upgrade cluster --help
```

16.10.1. CLI を使用したクラスターの手動アップグレード

1. 次のコマンドを実行して、利用可能なアップグレードがあるかどうかを確認します。

```
$ rosa list upgrade -c <cluster-name>
```

出力例

```
$ rosa list upgrade -c <cluster-name>
VERSION  NOTES
4.14.7   recommended
4.14.6
...
```

上の例では、バージョン 4.14.7 と 4.14.6 の両方が利用可能です。

2. 次のコマンドを実行して、1 時間以内にクラスターをアップグレードするようにスケジュールします。

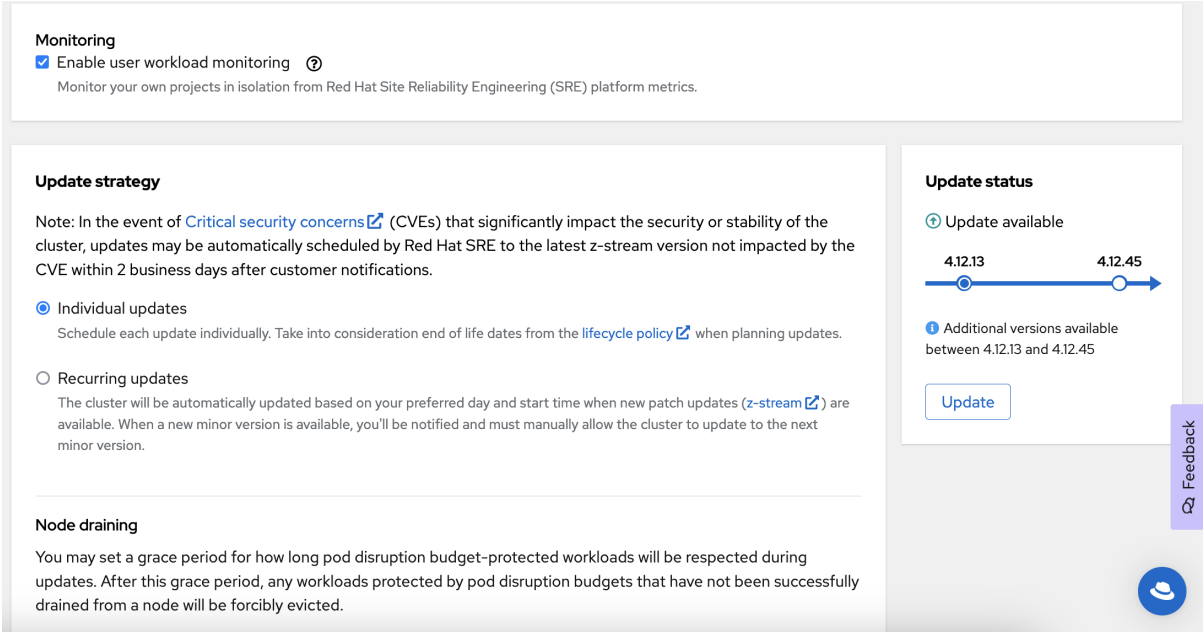
```
$ rosa upgrade cluster -c <cluster-name> --version <desired-version>
```

3. **オプション:** 次のコマンドを実行して、将来の日時にクラスターをアップグレードするようにスケジュールします。

```
$ rosa upgrade cluster -c <cluster-name> --version <desired-version> --schedule-date
<future-date-for-update> --schedule-time <future-time-for-update>
```

16.10.2. UI を使用したクラスターの手動アップグレード

1. OpenShift Cluster Manager にログインし、アップグレードするクラスターを選択します。
2. **Settings** をクリックします。
3. アップグレードが利用可能な場合は、**Update** をクリックします。



4. 新しいウィンドウでアップグレードするバージョンを選択します。
5. アップグレードの時間をスケジュールするか、すぐに開始します。

16.10.3. 自動定期アップグレードの設定

1. OpenShift Cluster Manager にログインし、アップグレードするクラスターを選択します。
2. **Settings** をクリックします。

1. **Update Strategy** で、**Recurring updates** をクリックします。

3. アップグレードを実行する日時を設定します。

4. **Node draining** で、Pod のエビクション前にノードをドレインできるようにする猶予期間を選択します。

5. **Save** をクリックします。

16.11. チュートリアル: クラスターの削除

コマンドラインインターフェイス (CLI) またはユーザーインターフェイス (UI) を使用して、Red Hat OpenShift Service on AWS (ROSA) クラスターを削除できます。

16.11.1. CLI を使用した ROSA クラスターの削除

1. **オプション:** 次のコマンドを実行して、クラスターをリストし、削除する正しいクラスターを確認します。

```
$ rosa list clusters
```

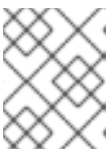
2. 次のコマンドを実行してクラスターを削除します。

```
$ rosa delete cluster --cluster <cluster-name>
```

警告

このコマンドは元に戻せません。

3. CLI に、クラスターを削除するかどうかを確認するプロンプトが表示されます。**y** を押してから **Enter** を押します。クラスターとそれに関連するすべてのインフラストラクチャーが削除されます。



注記

AWS STS および IAM のロールとポリシーはすべての残るため、クラスターの削除が完了したら、以下の手順に従って手動で削除する必要があります。

4. CLI に、作成した OpenID Connect (OIDC) プロバイダーおよび Operator IAM ロールのリソースを削除するコマンドが出力します。クラスターの削除が完了するまで待ってから、これらのリソースを削除します。次のコマンドを実行して、簡単なステータスチェックを実行します。

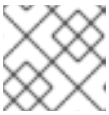
```
$ rosa list clusters
```

5. クラスターが削除されたら、次のコマンドを実行して OIDC プロバイダーを削除します。

```
$ rosa delete oidc-provider -c <clusterID> --mode auto --yes
```

6. 次のコマンドを実行して、Operator IAM ロールを削除します。

```
$ rosa delete operator-roles -c <clusterID> --mode auto --yes
```



注記

このコマンドには、クラスター名ではなくクラスター ID が必要です。

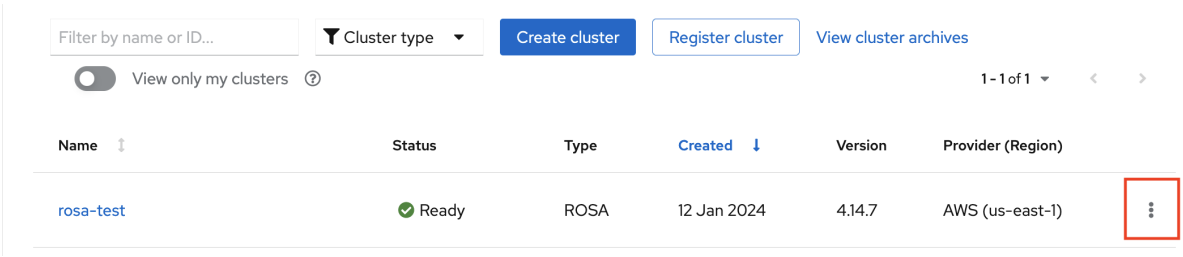
7. 残りのアカウントロールは、同じアカウント内の他のクラスターで必要なくなった場合にのみ削除してください。このアカウントで他の ROSA クラスターを作成する場合は、この手順を実行しないでください。
アカウントロールを削除するには、アカウントロールの作成時に使用した接頭辞を確認する必要があります。特に指定しなかった場合、デフォルトは "ManatedOpenShift" です。

   次のコマンドを実行して、アカウントのロールを削除します。

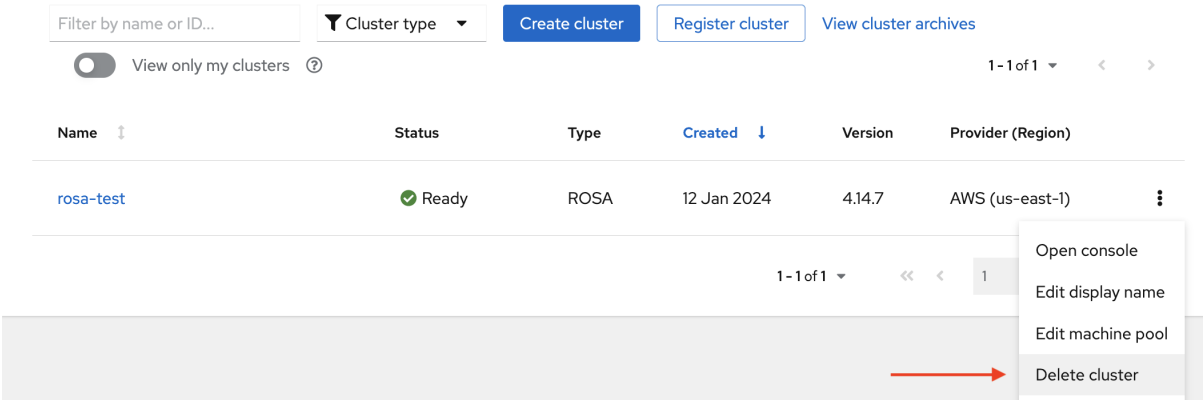
```
$ rosa delete account-roles --prefix <prefix> --mode auto --yes
```

16.11.2. UI を使用した ROSA クラスターの削除

1. Red Hat OpenShift Cluster Manager にログインし、削除するクラスターを見つけます。
2. クラスターの右側にある 3 つの点をクリックします。



3. ドロップダウンメニューで、**Delete cluster** をクリックします。



4. 削除を確認するクラスターの名前を入力し、**Delete** をクリックします。

16.12. チュートリアル: サポートの利用

必要なときに適切なサポートを受けることが重要です。ここでは、サポートが必要な場合に利用できるリソースをいくつか紹介します。

16.12.1. サポート連絡先の追加

クラスターに関する連絡用のメールアドレスを追加できます。

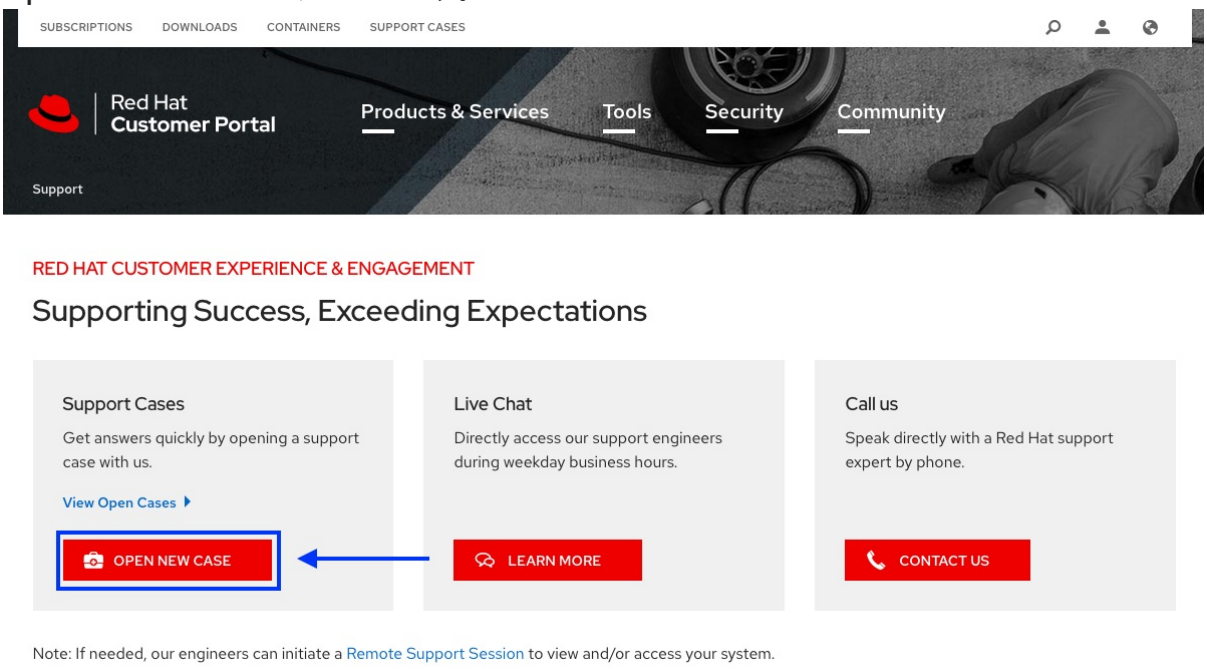
1. Red Hat OpenShift Cluster Manager ユーザーインターフェイス (UI) で、**select cluster** をクリックします。
2. **Support** タブをクリックします。
3. **Add notification contact** をクリックし、追加のメールアドレスを入力します。

16.12.2. UI を使用して Red Hat にサポートについて問い合わせる

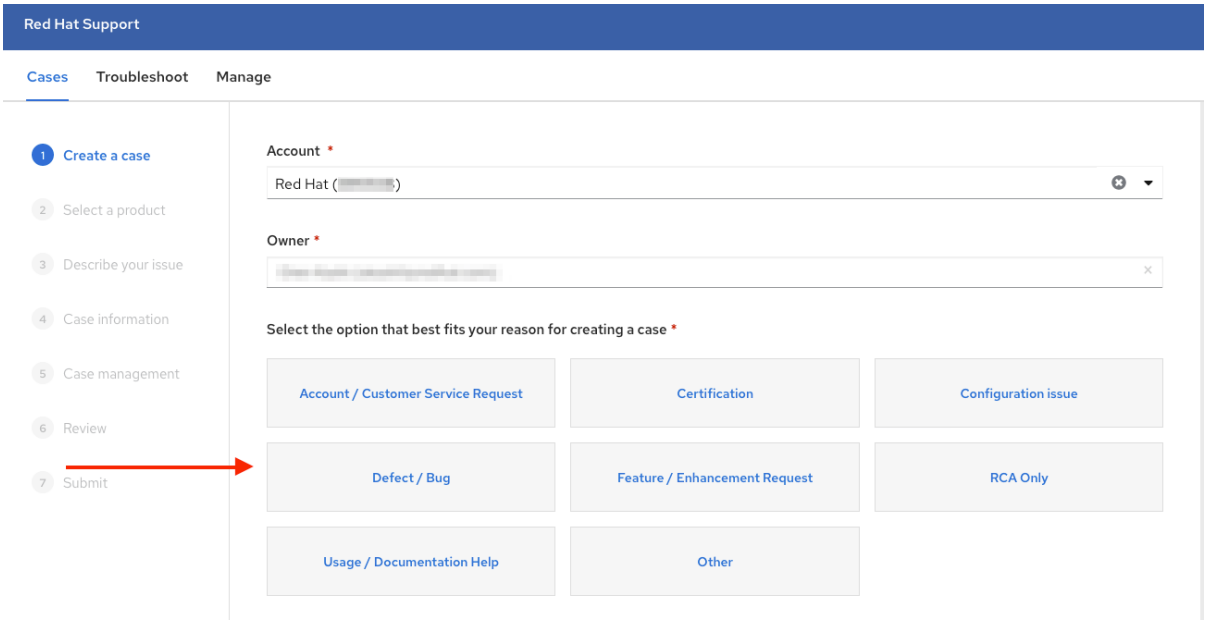
1. OpenShift Cluster Manager UI で、**Support** タブをクリックします。
2. **Open support case** をクリックします。

16.12.3. サポートページを使用して Red Hat にサポートについて問い合わせる

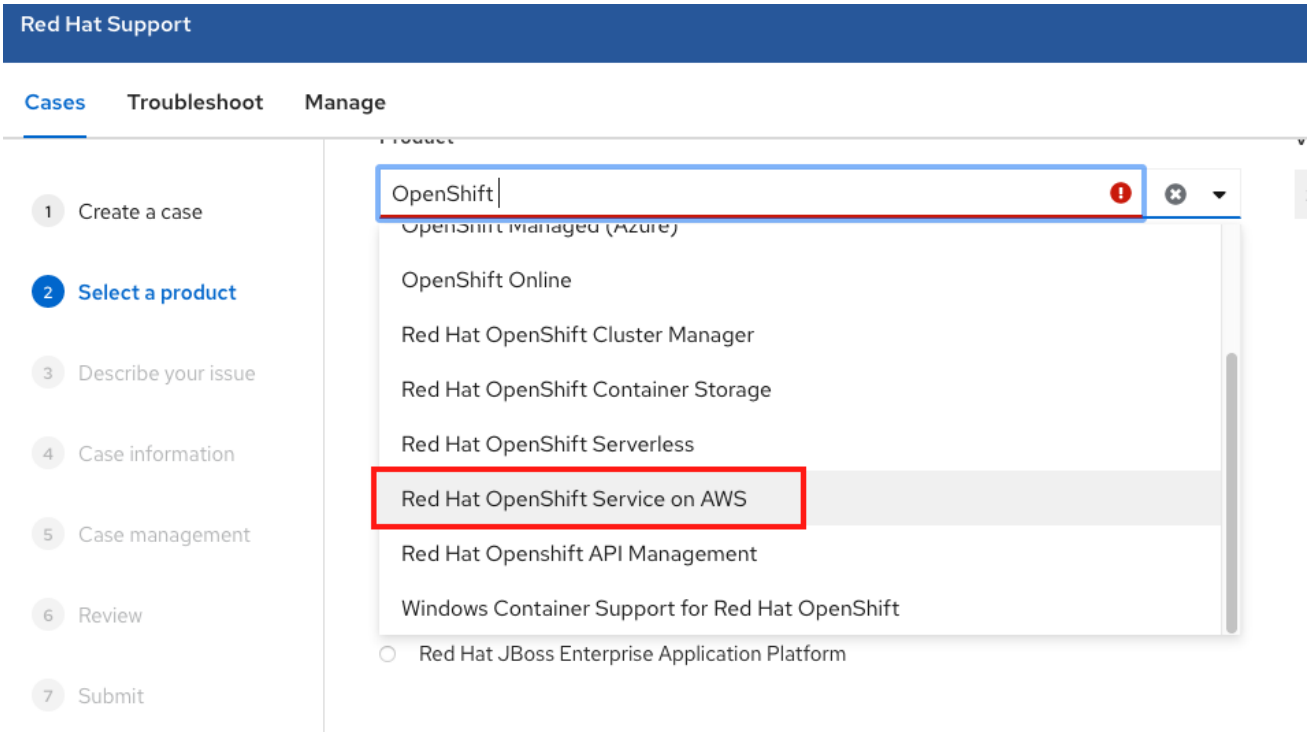
1. Red Hat サポートページ に移動します。
2. **Open a new Case** をクリックします。



3. Red Hat アカウントにログインします。
4. サポートに問い合わせる理由を選択します。



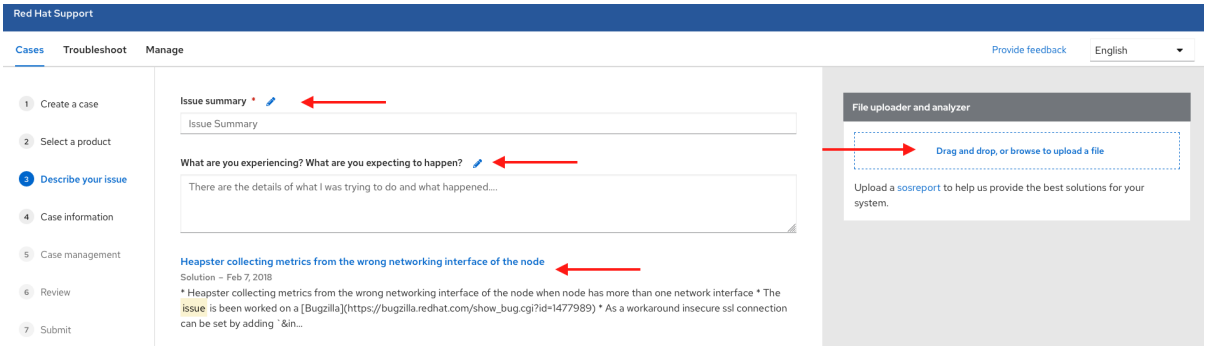
5. **Red Hat OpenShift Service on AWS**を選択します。



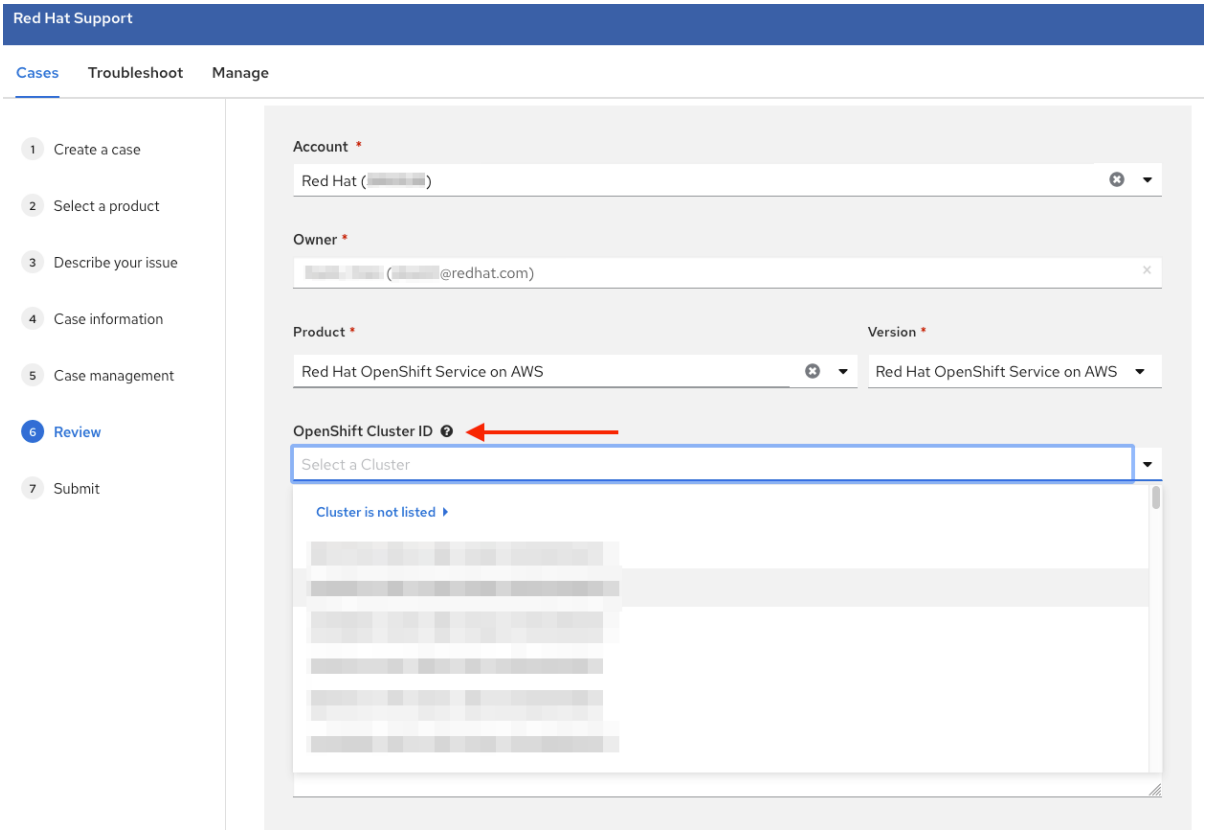
1. **continue** をクリックします。
2. 問題の概要とリクエストの詳細を入力します。ファイル、ログ、スクリーンショットをアップロードします。ご提供いただく情報が多いほど、Red Hat サポートが適切な解決方法を見つけやすくなります。

注記

このページの下部に、問題の解決に役立つ関連性の高い提案が表示されます。



3. **Continue** をクリックします。
4. 新しいフィールドの質問に回答します。
5. **Continue** をクリックします。
6. ケースに関する次の情報を入力します。
    a. **Support level:** Premium
    b. **Severity:** Red Hat サポートの重大度レベルの定義を確認して、正しいものを選択します。
    c. **Group:** このケースが他のいくつかのケースに関連する場合は、対応するグループを選択できます。
    d. **Language**
    e. **Send notifications:** アクティビティーの通知を受け取るためのメールアドレスを追加します。
    f. **Red Hat associates:** Red Hat の従業員とやり取りをしていて、その従業員と情報を共有する場合は、ここに従業員のメールアドレスを入力できます。
    g. **Alternate Case ID:** お客様独自の ID を割り当てる場合は、ここに入力できます。
7. **Continue** をクリックします。
8. 確認画面で、問い合わせに関連する正しいクラスター ID を選択していることを確認します。



9. **Submit** をクリックします。
10. 指定した重大度レベル の約束された応答時間に応じて連絡が届きます。

第17章 アプリケーションのデプロイ

17.1. チュートリアル: アプリケーションのデプロイ

17.1.1. 概要

クラスターのプロビジョニングが正常に完了したら、そのクラスターにアプリケーションをデプロイできます。このアプリケーションを使用すると、Red Hat OpenShift Service on AWS (ROSA) と Kubernetes の機能の一部をさらに詳しく知ることができます。

17.1.1.1. ラボの概要

このラボでは、コンテナーベースのアプリケーションのデプロイと操作の概念を理解するのに役立つ次の一連のタスクを完了します。

- S2I および Kubernetes Deployment オブジェクトを使用して、Node.js ベースのアプリをデプロイします。
- 継続的デリバリー (CD) パイプラインをセットアップして、ソースコードの変更を自動的にプッシュします。
- ロギングを調べます。
- アプリケーションの自己修復を試します。
- configmap、シークレット、環境変数を通じて設定の管理を確認します。
- 永続ストレージを使用して、Pod の再起動後もデータを保持します。
- Kubernetes とアプリケーション内のネットワークを確認します。
- ROSA と Kubernetes の機能についての理解を深めます。
- Horizontal Pod Autoscaler からの負荷に基づいて Pod を自動的にスケーリングします。
- AWS Controllers for Kubernetes (ACK) を使用して、S3 バケットをデプロイして使用します。

このラボでは、ROSA CLI または ROSA Web ユーザーインターフェイス (UI) を使用します。

17.2. チュートリアル: アプリケーションのデプロイ

17.2.1. 前提条件

1. プロビジョニングされた ROSA クラスター
   このラボは、正常にプロビジョニングされた ROSA クラスターにアクセスできることを前提としています。ROSA クラスターをまだ作成していない場合は、Red Hat OpenShift Service on AWS クイックスタートガイド で詳細を参照してください。

2. OpenShift コマンドラインインターフェイス (CLI)
   詳細は、OpenShift CLI の使用を開始する を参照してください。

3. GitHub アカウント
   既存の GitHub アカウントを使用するか、https://github.com/signup で登録します。

17.3. チュートリアル: アプリケーションのデプロイ

17.3.1. ラボの概要

17.3.1.1. ラボのリソース

- OSToy アプリケーションのソースコード
- OSToy フロントエンドコンテナーイメージ
- OSToy マイクロサービスコンテナーイメージ
- デプロイメント定義 YAML ファイル:

ostoy-frontend-deployment.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: ostoy-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ostoy-frontend
  labels:
    app: ostoy
spec:
  selector:
    matchLabels:
      app: ostoy-frontend
  strategy:
    type: Recreate
  replicas: 1
  template:
    metadata:
      labels:
        app: ostoy-frontend
    spec:
      # Uncomment to use with ACK portion of the workshop
      # If you chose a different service account name please replace it.
      # serviceAccount: ostoy-sa
      containers:
      - name: ostoy-frontend
        securityContext:
          allowPrivilegeEscalation: false
          runAsNonRoot: true
          seccompProfile:
            type: RuntimeDefault
```

```
          capabilities:
            drop:
            - ALL
        image: quay.io/ostoylab/ostoy-frontend:1.6.0
        imagePullPolicy: IfNotPresent
        ports:
        - name: ostoy-port
          containerPort: 8080
        resources:
          requests:
            memory: "256Mi"
            cpu: "100m"
          limits:
            memory: "512Mi"
            cpu: "200m"
        volumeMounts:
        - name: configvol
          mountPath: /var/config
        - name: secretvol
          mountPath: /var/secret
        - name: datavol
          mountPath: /var/demo_files
        livenessProbe:
          httpGet:
            path: /health
            port: 8080
          initialDelaySeconds: 10
          periodSeconds: 5
        env:
        - name: ENV_TOY_SECRET
          valueFrom:
            secretKeyRef:
              name: ostoy-secret-env
              key: ENV_TOY_SECRET
        - name: MICROSERVICE_NAME
          value: OSTOY_MICROSERVICE_SVC
        - name: NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
      volumes:
        - name: configvol
          configMap:
            name: ostoy-configmap-files
        - name: secretvol
          secret:
            defaultMode: 420
            secretName: ostoy-secret
        - name: datavol
          persistentVolumeClaim:
            claimName: ostoy-pvc
---
apiVersion: v1
kind: Service
metadata:
  name: ostoy-frontend-svc
```

```
  labels:
    app: ostoy-frontend
spec:
  type: ClusterIP
  ports:
    - port: 8080
      targetPort: ostoy-port
      protocol: TCP
      name: ostoy
  selector:
    app: ostoy-frontend
---
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: ostoy-route
spec:
  to:
    kind: Service
    name: ostoy-frontend-svc
---
apiVersion: v1
kind: Secret
metadata:
  name: ostoy-secret-env
type: Opaque
data:
  ENV_TOY_SECRET: VGhpcyBpcyBhIHRlc3Q=
---
kind: ConfigMap
apiVersion: v1
metadata:
  name: ostoy-configmap-files
data:
  config.json:  '{ "default": "123" }'
---
apiVersion: v1
kind: Secret
metadata:
  name: ostoy-secret
data:
  secret.txt:
VVNFUk5BTUU9bXlfdXNlcgpQQVNTV09SRD1AT3RCbCVYQXAhIzYzMlk1RndDQE1UUWsK
U01UUD1sb2NhbGhvc3QKU01UUF9QT1JUPTI1
type: Opaque
```

ostoy-microservice-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ostoy-microservice
  labels:
    app: ostoy
spec:
```

```
  selector:
    matchLabels:
      app: ostoy-microservice
  replicas: 1
  template:
    metadata:
      labels:
        app: ostoy-microservice
    spec:
      containers:
      - name: ostoy-microservice
        securityContext:
          allowPrivilegeEscalation: false
          runAsNonRoot: true
          seccompProfile:
            type: RuntimeDefault
          capabilities:
            drop:
            - ALL
        image: quay.io/ostoylab/ostoy-microservice:1.5.0
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 8080
          protocol: TCP
        resources:
          requests:
            memory: "128Mi"
            cpu: "50m"
          limits:
            memory: "256Mi"
            cpu: "100m"
---
apiVersion: v1
kind: Service
metadata:
  name: ostoy-microservice-svc
  labels:
    app: ostoy-microservice
spec:
  type: ClusterIP
  ports:
    - port: 8080
      targetPort: 8080
      protocol: TCP
  selector:
    app: ostoy-microservice
```

- ACK S3 の S3 バケットマニフェスト

  s3-bucket.yaml

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: ostoy-bucket
```

```
  namespace: ostoy
spec:
  name: ostoy-bucket
```

注記

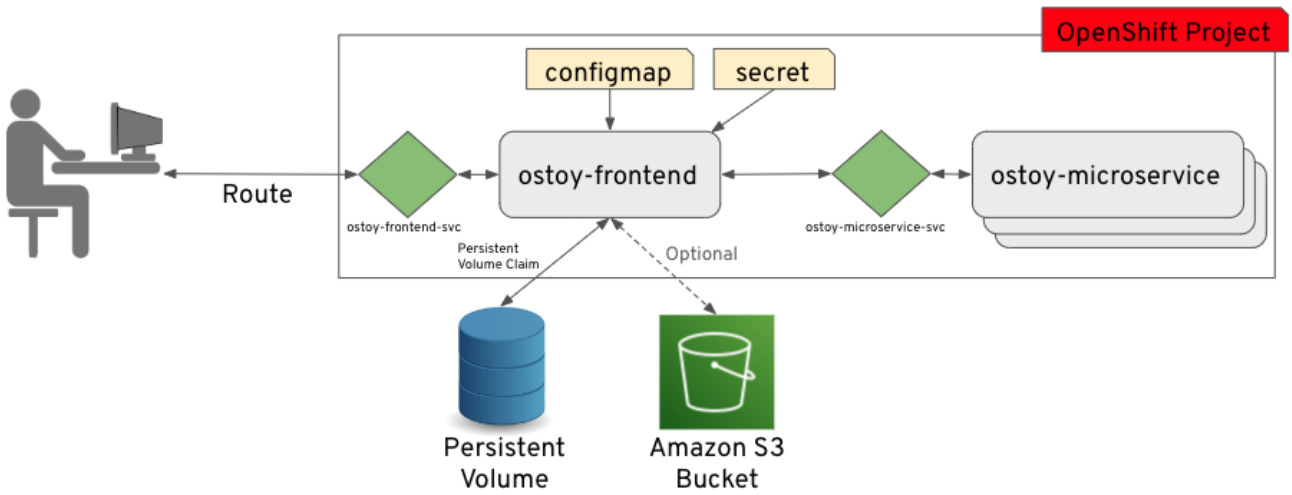
OSToy アプリケーションのデプロイを簡略化するために、上記のデプロイメントマニフェストで必要なすべてのオブジェクトはグループ化されています。一般的なエンタープライズデプロイメントでは、Kubernetes オブジェクトごとに個別のマニフェストファイルを使用することを推奨します。

17.3.1.2. OSToy アプリケーションについて

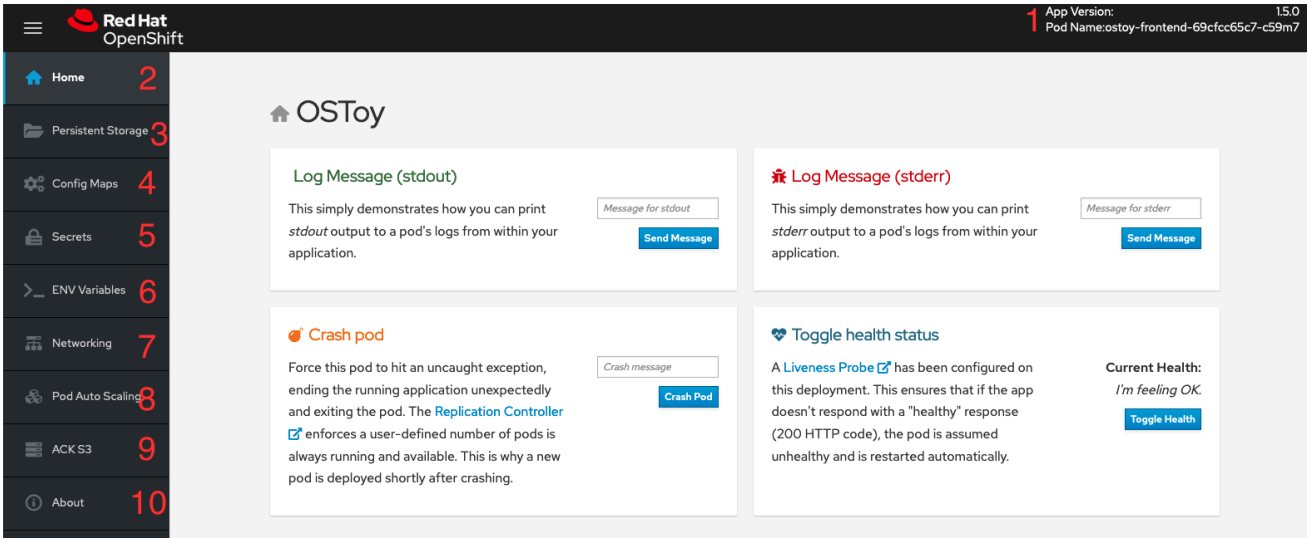
OSToy は、Kubernetes の機能を確認するのに役立つシンプルな Node.js アプリケーションです。このアプリケーションを ROSA クラスターにデプロイします。このアプリケーションのユーザーインターフェイスから、以下を実行できます。

- メッセージをログ (stdout/stderr) に書き込む。
- アプリケーションを意図的にクラッシュして自己修復を確認する。
- liveness プローブを切り替えて、OpenShift の動作を監視する。
- config map、シークレット、および環境変数を読み取る。
- 共有ストレージに接続されている場合にファイルの読み取りと書き込みを行う。
- ネットワーク接続、クラスター内 DNS、および含まれるマイクロサービスとの内部通信を確認する。
- 負荷を増やし、Horizontal Pod Autoscaler を使用して負荷を処理する Pod の自動スケーリングを確認する。
- オプション: AWS S3 バケットに接続して、オブジェクトの読み取りと書き込みを行う。

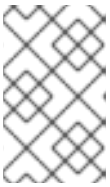
17.3.1.3. OSToy アプリケーションの図



17.3.1.4. OSToy の UI について



1. ブラウザーにページを提供した Pod 名が表示されます。
2. **Home:** アプリケーションのメインページです。このラボで確認する機能の一部を実行できます。
3. **Persistent Storage:** このアプリケーションにバインドされた永続ボリュームにデータを書き込むことができます。
4. **Config Maps:** アプリケーションで使用可能な設定マップの内容とキーと値のペアが表示されます。
5. **Secrets:** アプリケーションで使用可能なシークレットの内容とキーと値のペアが表示されます。
6. **ENV Variables:** アプリケーションで使用可能な環境変数が表示されます。
7. **Networking:** アプリケーション内のネットワークを説明するツール。
8. **Pod Auto Scaling:** Pod の負荷を増やし、HPA をテストするツール。
9. **ACK S3:** (オプション) AWS S3 と統合して、バケットへのオブジェクトの読み取りおよび書き込みを行います。



注記

OSToy の "ACK S3" セクションを表示するには、このワークショップの ACK セクションを完了する必要があります。このセクションを完了しなくても、OSToy アプリケーションは機能します。

10. **About:** アプリケーションに関する詳細情報が表示されます。