OpenShift Dedicated 4

チュートリアル

OpenShift Dedicated チュートリアル

Last Updated: 2025-12-06

OpenShift Dedicated 4 チュートリアル

OpenShift Dedicated チュートリアル

Legal Notice

Abstract

OpenShift Dedicated クラスター管理に関するチュートリアル。

Table of Contents

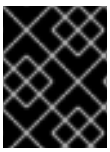
第1章 チュートリアルの概要 3

第2章 チュートリアル: カスタムドメインと TLS 証明書を使用してコンポーネントルートを更新する 4
2.1. 前提条件 4
2.2. 環境の設定 4
2.3. 現在のルートを確認する 4
2.4. 各コンポーネントルートに有効な TLS 証明書を作成する 6
2.5. 証明書をシークレットとしてクラスターに追加する 6
2.6. クラスター内のロードバランサーのロードバランサー IP を見つける 7
2.7. ホスティングプロバイダーにコンポーネントルート DNS レコードを追加する 7
2.8. OCM CLI を使用してコンポーネントルートと TLS シークレットを更新する 7
2.9. OCM CLI を使用してコンポーネントルートをデフォルトにリセットする 8

第3章 チュートリアル: GOOGLE CLOUD NEXT GENERATION FIREWALL で EGRESS を制限する 10
3.1. 前提条件の確認 10
3.2. 環境の設定 10
3.3. VPC とサブネットの作成 11
3.4. グローバルネットワークファイアウォールポリシーのデプロイ 11
3.5. CLOUD ROUTER と CLOUD NETWORK ADDRESS TRANSLATION ゲートウェイの作成 12
3.6. プライベート GOOGLE アクセス用のプライベート DOMAIN NAME SYSTEM レコードの作成 12
3.7. ファイアウォールルールの作成 13
3.8. クラスターの作成 14
3.9. クラスターの削除 14
3.10. リソースのクリーンアップ 14

第1章 チュートリアルの概要

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



重要

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

第2章 チュートリアル: カスタムドメインと TLS 証明書を使用してコンポーネントルートを更新する

このガイドでは、Google Cloud バージョン 4.14 以降上の OpenShift Dedicated で、Web コンソール、OAuth サーバー、およびダウンロードコンポーネントルートのホスト名と TLS 証明書を変更する方法を説明します。[1]

コンポーネントルートに加える変更[2] このガイドで説明されている内容の詳細は、OpenShift Dedicated ドキュメントの 内部 OAuth サーバー URL のカスタマイズ、コンソールルートのカスタマイズ、および ダウンロードルートのカスタマイズ を参照してください。

2.1. 前提条件

- OCM CLI (**ocm**) バージョン 1.0.5 以上
- gcloud CLI (**gcloud**)
- Google Cloud クラスター上で稼働しているバージョン 4.14 以上の OpenShift Dedicated
- OpenShift CLI (**oc**)
- **jq** CLI
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenSSL (デモ用の SSL/TLS 証明書を生成するため)

2.2. 環境の設定

1. **cluster-admin** 権限を持つアカウントを使用してクラスターにログインします。
2. クラスター名の環境変数を設定します。

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

3. 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}"
```

出力例

```
Cluster: my-osd-cluster
```

2.3. 現在のルートを確認する

1. デフォルトのホスト名でコンポーネントルートに到達できることを確認します。**openshift-console** および **openshift-authentication** プロジェクトでルートのリストをクエリーすることで、ホスト名を確認できます。

```
$ oc get routes -n openshift-console
$ oc get routes -n openshift-authentication
```

出力例

```
NAME        HOST/PORT                                                    PATH       SERVICES
PORT    TERMINATION          WILDCARD
console     console-openshift-console.apps.my-example-cluster-
gcp.z9a9.p2.openshiftapps.com    ... 1 more  console    https    reencrypt/Redirect    None
downloads   downloads-openshift-console.apps.my-example-cluster-
gcp.z9a9.p2.openshiftapps.com  ... 1 more  downloads  http    edge/Redirect        None
NAME             HOST/PORT                                                PATH       SERVICES
PORT    TERMINATION          WILDCARD
oauth-openshift   oauth-openshift.apps.my-example-cluster-gcp.z9a9.p2.openshiftapps.com
... 1 more  oauth-openshift   6443    passthrough/Redirect    None
```

この出力から、ベースホスト名が **z9a9.p2.openshiftapps.com** であることを確認できます。

2. 次のコマンドを実行して、デフォルト Ingress の ID を取得します。

```
$ export INGRESS_ID=$(ocm list ingress -c ${CLUSTER_NAME} -o json | jq -r '.[] |
select(.default == true) | .id')
```

3. 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 "Ingress ID: ${INGRESS_ID}"
```

出力例

```
Ingress ID: r3l6
```

これらのコマンドを実行すると、クラスターのデフォルトのコンポーネントルートが次のとおりであることがわかります。

- コンソール: **console-openshift-console.apps.my-example-cluster-gcp.z9a9.p2.openshiftapps.com**
- ダウンロード: **downloads-openshift-console.apps.my-example-cluster-gcp.z9a9.p2.openshiftapps.com**
- OAuth: **oauth-openshift.apps.my-example-cluster-gcp.z9a9.p2.openshiftapps.com**

ocm edit ingress コマンドを使用して、各サービスのホスト名を変更し、すべてのコンポーネントルートに TLS 証明書を追加できます。関連するパラメーターは、以下に示す **ocm edit ingress** コマンドのコマンドラインヘルプの抜粋で確認できます。

```
$ ocm edit ingress -h
Edit a cluster ingress for a cluster. Usage:
  ocm edit ingress ID [flags]
  [...]
  --component-routes string              Component routes settings. Available keys [oauth, console,
```

> downloads]. For each key a pair of hostname and tlsSecretRef is expected to be supplied. Format should be a comma separate list 'oauth: hostname=example-hostname;tlsSecretRef=example-secret-ref,downloads:...'

この例では、次のカスタムコンポーネントルートを使用します。

- Console の場合: **console.my-new-domain.dev**
- Downloads の場合: **downloads.my-new-domain.dev**
- OAuth の場合: **oauth.my-new-domain.dev**

2.4. 各コンポーネントルートに有効な TLS 証明書を作成する

このセクションでは、3 つの個別の自己署名証明書キーペアを作成し、それらを信頼して、実際の Web ブラウザーを使用して新しいコンポーネントルートにアクセスできることを確認します。

警告

これはデモンストレーションのみを目的としており、実稼働環境のワークロードのソリューションとしては推奨されません。実稼働環境のワークロード用に同様の属性を持つ証明書を作成する方法は、認証局に問い合わせてください。

重要

HTTP/2 接続の結合に関する問題を防ぐには、エンドポイントごとに個別の証明書を使用する必要があります。ワイルドカードまたは SAN 証明書の使用はサポートされません。

- 各コンポーネントルートの証明書を生成し、使用するコンポーネントルートのカスタムドメインに、証明書のサブジェクト (**-subj**) を設定するように注意してください。

  例

  ```
  $ openssl req -newkey rsa:2048 -new -nodes -x509 -days 365 -keyout key-console.pem -out cert-console.pem -subj "/CN=console.my-new-domain.dev"
  $ openssl req -newkey rsa:2048 -new -nodes -x509 -days 365 -keyout key-downloads.pem -out cert-downloads.pem -subj "/CN=downloads.my-new-domain.dev"
  $ openssl req -newkey rsa:2048 -new -nodes -x509 -days 365 -keyout key-oauth.pem -out cert-oauth.pem -subj "/CN=oauth.my-new-domain.dev"
  ```

  これにより、**.pem** ファイル、**key-<component>.pem**、**cert-<component>.pem** ペアが 3 つ生成されます。

2.5. 証明書をシークレットとしてクラスターに追加する

- **openshift-config** namespace に 3 つの TLS シークレットを作成します。

これは、このガイドで後にコンポーネントルートを更新するときのシークレット参照になります。

```
$ oc create secret tls console-tls --cert=cert-console.pem --key=key-console.pem -n
openshift-config
$ oc create secret tls downloads-tls --cert=cert-downloads.pem --key=key-downloads.pem -n
openshift-config
$ oc create secret tls oauth-tls --cert=cert-oauth.pem --key=key-oauth.pem -n openshift-
config
```

2.6. クラスター内のロードバランサーのロードバランサー IP を見つける

クラスターを作成すると、サービスによってロードバランサーが作成され、そのロードバランサーのロードバランサー IP が生成されます。クラスターの DNS レコードを作成するために、ロードバランサーの IP を確認する必要があります。

ロードバランサー IP を確認するには、**openshift-ingress** namespace に対して **oc get svc** コマンドを実行します。ロードバランサーの IP は、**openshift-ingress** namespace の **router-default** サービスに関連付けられた **EXTERNAL-IP** です。

```
$ oc get svc -n openshift-ingress
NAME           TYPE           CLUSTER-IP     EXTERNAL-IP      PORT(S)                      AGE
router-default  LoadBalancer  172.30.237.88  34.85.169.230     80:31175/TCP,443:31554/TCP  76d
```

この場合、ロードバランサーの IP は **34.85.169.230** です。

この値を後の手順のために保存します。新しいコンポーネントルートのホスト名に DNS レコードを設定する際に必要になります。

2.7. ホスティングプロバイダーにコンポーネントルート DNS レコードを追加する

DNS 設定で A レコードを作成し、ドメインをルーターのデフォルトのロードバランサーの IP アドレスを参照します。

2.8. OCM CLI を使用してコンポーネントルートと TLS シークレットを更新する

DNS レコードが更新されたら、OCM CLI を使用してコンポーネントルートを変更できます。

1. **ocm edit ingress** コマンドを使用して、デフォルトの Ingress ルートを新しいベースドメインとそれに関連付けられたシークレット参照で更新します。その際、各コンポーネントルートのホスト名を更新するように注意してください。

```
$ ocm edit ingress -c ${CLUSTER_NAME} ${INGRESS_ID} --component-routes 'console:
hostname=console.my-new-domain.dev;tlsSecretRef=console-tls,downloads:
hostname=downloads.my-new-domain.dev;tlsSecretRef=downloads-tls,oauth:
hostname=oauth.my-new-domain.dev;tlsSecretRef=oauth-tls'
```

注記

変更しないコンポーネントルートを空の文字列に設定したままにして、コンポーネントルートのサブセットのみを編集することもできます。たとえば、コンソールおよび OAuth サーバーのホスト名および TLS 証明書のみを変更する場合は、以下のコマンドを実行します。

```
$ ocm edit ingress -c ${CLUSTER_NAME} ${INGRESS_ID} --component-routes 'console: hostname=console.my-new-domain.dev;tlsSecretRef=console-tls,downloads: hostname="";tlsSecretRef="", oauth: hostname=oauth.my-new-domain.dev;tlsSecretRef=oauth-tls'
```

2. **ocm list ingress** コマンドを実行して、変更が正常に行われたことを確認します。

```
$ ocm list ingress -c ${CLUSTER_NAME} -ojson | jq ".[] | select(.id == \"${INGRESS_ID}\") | .component_routes"
```

出力例

```
{
  "console": {
    "kind": "ComponentRoute",
    "hostname": "console.my-new-domain.dev",
    "tls_secret_ref": "console-tls"
  },
  "downloads": {
    "kind": "ComponentRoute",
    "hostname": "downloads.my-new-domain.dev",
    "tls_secret_ref": "downloads-tls"
  },
  "oauth": {
    "kind": "ComponentRoute",
    "hostname": "oauth.my-new-domain.dev",
    "tls_secret_ref": "oauth-tls"
  }
}
```

3. ローカルシステムのトラストストアに証明書を追加し、ローカル Web ブラウザーを使用して新しいルートでコンポーネントにアクセスできることを確認します。

2.9. OCM CLI を使用してコンポーネントルートをデフォルトにリセットする

コンポーネントルートをデフォルト設定にリセットする場合は、次の **ocm edit ingress** コマンドを実行します。

```
$ ocm edit ingress -c ${CLUSTER_NAME} ${INGRESS_ID} --component-routes 'console: hostname="";tlsSecretRef="",downloads: hostname="";tlsSecretRef="", oauth: hostname="";tlsSecretRef=""'
```

[1] OpenShift Dedicated OCM バージョン 4.14 より前のバージョンでは、これらのルートの変更は通常サポートされていません。ただし、バージョン 4.13 を使用しているクラスターがある場合は、サポートケースを作成 して、バージョン 4.13 クラスターでこの機能のサポートを有効にするように Red Hat サポートにリクエストできます。

[2] OCM が最初にインストールされたときに提供される OAuth、コンソール、およびダウンロードルートを指すために、「コンポーネントルート」という用語を使用します。

第3章 チュートリアル: GOOGLE CLOUD NEXT GENERATION FIREWALL で EGRESS を制限する

このガイドは、Google Cloud's Next Generation Firewall (NGFW) を使用して、Google Cloud 上の OpenShift Dedicated で Egress 制限を実装するために使用できます。NGFW は、ファイアウォールポリシールールで完全修飾ドメイン名 (FQDN) オブジェクトを許可する、完全に分散されたファイアウォールサービスです。これは、OpenShift Dedicated が依存する多くの外部エンドポイントで必要です。

重要

ファイアウォールやその他のネットワークデバイスを使用して Egress トラフィックを制限する機能は、Private Service Connect (PSC) を使用してデプロイされた OpenShift Dedicated クラスターでのみサポートされます。PSC を使用しないクラスターでこの機能を使用するには、サポート例外が必要です。さらにサポートが必要な場合は、サポートケース を作成してください。

3.1. 前提条件の確認

- Google Cloud コマンドラインインターフェイス (**gcloud**) がインストールされている。
- Google Cloud CLI にログインしており、OpenShift Dedicated をデプロイする予定の Google Cloud プロジェクトを選択した。
- Google Cloud で、以下を含む必要な最小限の権限がある。
  - **Compute Network Admin**
  - **DNS Administrator**
- ターミナルで次のコマンドを実行して、特定のサービスを有効にした。

```
$ gcloud services enable networksecurity.googleapis.com
```

```
$ gcloud services enable networkservices.googleapis.com
```

```
$ gcloud services enable servicenetworking.googleapis.com
```

3.2. 環境の設定

ターミナルで、次の環境変数を設定します。

```
export project_id=$(gcloud config list --format="value(core.project)")
export region=us-east1
export prefix=osd-ngfw
export service_cidr="172.30.0.0/16"
export machine_cidr="10.0.0.0/22"
export pod_cidr="10.128.0.0/14"
```

この例では、デプロイ先のリージョンとして **us-east1** を使用し、クラスターのリソースの接頭辞として **osd-ngfw** を使用します。サービスおよび Pod ネットワークにはデフォルトの CIDR 範囲が割り当てられます。マシンの CIDR は、このチュートリアルの後半で設定するサブネット範囲に基づいていま

す。ニーズに合わせてパラメーターを変更します。

3.3. VPC とサブネットの作成

Google Cloud NGFW をデプロイする前に、まず OpenShift Dedicated に使用する Virtual Private Cloud (VPC) とサブネットを作成する必要があります。

1. 次のコマンドを実行して VPC を作成します。

```
$ gcloud compute networks create ${prefix}-vpc --subnet-mode=custom
```

2. 次のコマンドを実行してワーカーサブネットを作成します。

```
$ gcloud compute networks subnets create ${prefix}-worker \
    --range=10.0.2.0/23 \
    --network=${prefix}-vpc \
    --region=${region} \
    --enable-private-ip-google-access
```

3. 次のコマンドを実行してコントロールプレーンサブネットを作成します。

```
$ gcloud compute networks subnets create ${prefix}-control-plane \
    --range=10.0.0.0/25 \
    --network=${prefix}-vpc \
    --region=${region} \
    --enable-private-ip-google-access
```

4. 次のコマンドを実行して PSC サブネットを作成します。

```
$ gcloud compute networks subnets create ${prefix}-psc \
    --network=${prefix}-vpc \
    --region=${region} \
    --stack-type=IPV4_ONLY \
    --range=10.0.0.128/29 \
    --purpose=PRIVATE_SERVICE_CONNECT
```

これらの例では、サブネット範囲としてワーカーサブネットに 10.0.2.0/23、コントロールプレーンサブネットに 10.0.0.0/25、PSC サブネットに 10.0.0.128/29 を使用します。ニーズに合わせてパラメーターを変更します。パラメーター値が、ここまでのチュートリアルで設定したマシン CIDR 内に含まれていることを確認してください。

3.4. グローバルネットワークファイアウォールポリシーのデプロイ

1. 次のコマンドを実行して、グローバルネットワークファイアウォールポリシーを作成します。

```
$ gcloud compute network-firewall-policies create \
    ${prefix} \
    --description "OpenShift Dedicated Egress Firewall" \
    --global
```

2. 次のコマンドを実行して、新しく作成したグローバルネットワークファイアウォールポリシーを上記で作成した VPC に関連付けます。

```
$ gcloud compute network-firewall-policies associations create \
    --firewall-policy ${prefix} \
    --network ${prefix}-vpc \
    --global-firewall-policy
```

3.5. CLOUD ROUTER と CLOUD NETWORK ADDRESS TRANSLATION ゲートウェイの作成

Network Address Translation (NAT) ゲートウェイは、すべてのトラフィックを単一のパブリック IP アドレスの下にマスカレードすることで、プライベート仮想マシンのインターネット接続を可能にします。指定された出口ポイントとして、更新の取得などのすべての送信要求に対して内部 IP を変換します。このプロセスにより、プライベートアドレスを公開することなく、インターネットへのアクセスが効果的に許可されます。

1. 次のコマンドを実行して、Cloud NAT の IP アドレスを予約します。

```
$ gcloud compute addresses create ${prefix}-${region}-cloudnatip \
    --region=${region}
```

2. 次のコマンドを実行して Cloud Router を作成します。

```
$ gcloud compute routers create ${prefix}-router \
    --region=${region} \
    --network=${prefix}-vpc
```

3. 次のコマンドを実行して Cloud NAT を作成します。

```
$ gcloud compute routers nats create ${prefix}-cloudnat-${region} \
    --router=${prefix}-router --router-region ${region} \
    --nat-all-subnet-ip-ranges \
    --nat-external-ip-pool=${prefix}-${region}-cloudnatip
```

3.6. プライベート GOOGLE アクセス用のプライベート DOMAIN NAME SYSTEM レコードの作成

プライベート Domain Name System (DNS) ゾーンは、トラフィックがパブリックインターネットを経由しないようにすることで、リソースが Google API に接続する方法を最適化します。これは、Google サービスに対する DNS 要求を傍受してプライベート IP アドレスに解決し、Google の内部ネットワークへの接続を強制することで、より高速で安全なデータ交換を実現します。

1. 次のコマンドを実行して、googleapis.com ドメインのプライベート DNS ゾーンを作成します。

```
$ gcloud dns managed-zones create ${prefix}-googleapis \
    --visibility=private \
    --
networks=https://www.googleapis.com/compute/v1/projects/${project_id}/global/networks/${prefix}-vpc \
    --description="Private Google Access" \
    --dns-name=googleapis.com
```

2. 次のコマンドを実行して、レコードセットトランザクションを開始します。

```
$ gcloud dns record-sets transaction start \
    --zone=${prefix}-googleapis
```

3. 次のコマンドを実行して、googleapis.com ドメインの配下にある Google API の DNS レコードをステージングします。

```
$ gcloud dns record-sets transaction add --name="*.googleapis.com." \
    --type=CNAME restricted.googleapis.com. \
    --zone=${prefix}-googleapis \
    --ttl=300
```

```
$ gcloud dns record-sets transaction add 199.36.153.4 199.36.153.5 199.36.153.6
199.36.153.7 \
    --name=restricted.googleapis.com. \
    --type=A \
    --zone=${prefix}-googleapis \
    --ttl=300
```

4. 次のコマンドを実行して、上記で開始したステージングされたレコードセットトランザクションを適用します。

```
$ gcloud dns record-sets transaction execute \
    --zone=$prefix-googleapis
```

3.7. ファイアウォールルールの作成

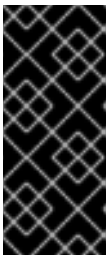
1. 次のコマンドを実行して、プライベート IP (RFC 1918) アドレス空間の一括許可ルールを作成します。

```
$ gcloud compute network-firewall-policies rules create 500 \
    --description "Allow egress to private IP ranges" \
    --action=allow \
    --firewall-policy=${prefix} \
    --global-firewall-policy \
    --direction=EGRESS \
    --layer4-configs all \
    --dest-ip-ranges=10.0.0.0/8,172.16.0.0/12,192.168.0.0/16
```

2. 次のコマンドを実行して、OpenShift Dedicated に必要な HTTPS (tcp/443) ドメインの許可ルールを作成します。

```
$ gcloud compute network-firewall-policies rules create 600 \
    --description "Allow egress to OpenShift Dedicated required domains (tcp/443)" \
    --action=allow \
    --firewall-policy=${prefix} \
    --global-firewall-policy \
    --direction=EGRESS \
    --layer4-configs tcp:443 \
    --dest-fqdns accounts.google.com,pull.q1w2.quay.rhcloud.com,http-inputs-
osdsecuritylogs.splunkcloud.com,nosnch.in,api.deadmanssnitch.com,events.pagerduty.com,api
pagerduty.com,api.openshift.com,mirror.openshift.com,observatorium.api.openshift.com,observ
atorium-
mst.api.openshift.com,console.redhat.com,infogw.api.openshift.com,api.access.redhat.com,cert
```

```
api.access.redhat.com,catalog.redhat.com,sso.redhat.com,registry.connect.redhat.com,registry.access.redhat.com,cdn01.quay.io,cdn02.quay.io,cdn03.quay.io,cdn04.quay.io,cdn05.quay.io,cdn06.quay.io,cdn.quay.io,quay.io,registry.redhat.io,quayio-production-s3.s3.amazonaws.com
```



重要

トラフィックを許可するための一致するルールがない場合、ファイアウォールによってブロックされます。内部ネットワークやその他の外部エンドポイントなど、他のリソースへのアクセスを許可するには、優先度が 1000 未満の追加のルールを作成します。ファイアウォールルールを作成する方法の詳細は、Use global network firewall policies and rules を参照してください。

3.8. クラスターの作成

これで、Google Cloud 上に OpenShift Dedicated クラスターを作成する準備が整いました。詳細は、Workload Identity Federation 認証を使用して Google Cloud 上にクラスターを作成する を参照してください。

3.9. クラスターの削除

クラスターを削除するには、Google Cloud 上の OpenShift Dedicated クラスターの削除 を参照してください。

3.10. リソースのクリーンアップ

継続的な課金を防ぐには、クラスターを削除した後、このチュートリアルで作成した Google Cloud ネットワークインフラストラクチャーを手動で削除する必要があります。クラスターを削除しても、これらの基盤となるリソースは自動的に削除されません。これらのリソースは、gcloud CLI コマンドと Google Cloud コンソール内のアクションを組み合わせてクリーンアップできます。

このチュートリアル用に作成したリソースをクリーンアップするプロセスを開始する前に、次のコマンドを実行し、プロンプトをすべて完了します。

1. 次のコマンドを実行して、アイデンティティーを認証します。

```
$ gcloud init
```

2. 次のコマンドを実行して、Google Cloud アカウントにログインします。

```
$ gcloud auth application-default login
```

3. 次のコマンドを実行して、OpenShift Cluster Manager CLI ツールにログインします。

```
$ ocm login --use-auth-code
```

これで、このチュートリアルで作成したリソースをクリーンアップする準備が整いました。リソースの依存関係を考慮して、作成と逆の順序で削除します。

1. 次のコマンドを実行して、ファイアウォールポリシーと VPC の関連付けを削除します。

```
$ gcloud compute network-firewall-policies associations delete \
    --firewall-policy=${prefix} \
    --network=${prefix}-vpc \
```

```
    --global-firewall-policy
```

2. 次のコマンドを実行して、グローバルネットワークファイアウォールポリシーを削除します。

```
$ gcloud compute network-firewall-policies delete ${prefix} --global
```

3. すべてのユーザー定義レコードセットが削除されるまで、Google Cloud のマネージド DNS ゾーンは削除できません。次のコマンドを実行して、クリーンアップする Google Cloud プロジェクトとマネージド DNS ゾーンをクリーンアップを対象とする変数を定義します。

```
$ cat /tmp/delete_records.sh
PROJECT_ID=<your-project-id>
ZONE_NAME=<your-managed-zone-name>
```

4. 次のコマンドを実行して、プライベート DNS ゾーン内に含まれるレコードセットをリスト表示します。

```
$ gcloud \
    dns record-sets list \
    --project=$PROJECT_ID \
    --zone=$ZONE_NAME \
    --filter="type!=NS AND type!=SOA" \
    --format="value(name,type)" | while read name type;
```

5. 次のコマンドを実行して、そのプライベート DNS ゾーン内に含まれるレコードセットを削除します。

```
$ gcloud --project=$PROJECT_ID dns record-sets delete "$name" --zone=$ZONE_NAME --
type="$type"
```

6. 次のコマンドを実行して、プライベート DNS ゾーンを削除します。

```
$ gcloud dns managed-zones delete ${prefix}-googleapis
```

7. Cloud NAT ゲートウェイを削除します。

```
$ gcloud compute routers nats delete ${prefix}-cloudnat-${region} \
    --router=${prefix}-router \
    --router-region=${region}
```

8. 次のコマンドを実行して、Cloud Router を削除します。

```
$ gcloud compute routers delete ${prefix}-router --region=${region}
```

9. 次のコマンドを実行して、予約済みの IP アドレスを削除します。

```
$ gcloud compute addresses delete ${prefix}-${region}-cloudnatip --region=${region}
```

10. 次のコマンドを実行して、ワーカーサブネットを削除します。

```
$ gcloud compute networks subnets delete ${prefix}-worker --region=${region}
```

11. 次のコマンドを実行して、コントロールプレーンサブネットを削除します。

```
$ gcloud compute networks subnets delete ${prefix}-control-plane --region=${region}
```

12. 次のコマンドを実行して、PSC サブネットを削除します。

```
$ gcloud compute networks subnets delete ${prefix}-psc --region=${region}
```

13. 次のコマンドを実行して、VPC を削除します。

```
$ gcloud compute networks delete ${prefix}-vpc
```