

1. Oracle Cloud Infrastructureへの移行について

既存のOracle Application Container Cloud ServiceアプリケーションをOracle Cloud Infrastructureに移行し、移行プロセスの概要を取得する利点について学習します。

トピックス:

- Oracle Cloud Infrastructureに移行する理由
 - 移行スコープについて
 - Oracle Cloud Infrastructureについて
 - Oracle Cloud Infrastructure Container Engine for Kubernetesについて
 - Oracle Cloud Infrastructureユーザーおよびグループについて
 - 移行タスク・フローについて
-

Oracle Cloud Infrastructureに移行する理由

Oracleは、Oracle Cloud Infrastructure Classicリージョンから既存のクラウド・リソースを移行することをお勧めします。その方法により、複数の利点を得ることができます。

Oracle Cloudでは、地理的なロケーションにローカライズされた特定のリージョンのリソースをプロビジョニングします。リージョンは、Oracle Cloud Infrastructure ClassicまたはOracle Cloud Infrastructureプラットフォームをサポートします。

Oracle Cloud Infrastructureは、最新のクラウド・テクノロジーおよび標準に基づいた、Oracleの最新インフラストラクチャ・プラットフォームです。通常は、Oracle Cloud Infrastructure Classicよりもパフォーマンスが優れています。Oracle Cloud Infrastructureでは、1時間当たりOracle Compute Units (OCPU)の観点から、より予測可能な価格設定とコストも低くなります。リージョン、サービスおよび機能の追加を含め、OracleがOracle Cloud Infrastructureの投資を続行することが最も重要です。[プラットフォームおよびインフラストラクチャ・サービスのデータ・リージョン](#)を参照してください。

Oracle Cloud Infrastructure Classicのクラウド・リソースを移行する際に、Oracle Cloud Infrastructureの次の追加の管理機能からメリットを得ることができます:

- クラウド・リソースを論理コンパートメントの階層に編成します。
 - 各コンパートメントのファイングレイン・アクセス・ポリシーを作成します。
-

移行スコープについて

Oracle Application Container Cloud ServiceアプリケーションをOracle Cloud Infrastructure Container Engine for Kubernetesに移行する前に、このプロセスのスコープと制約を考慮します。

Oracleでは、次のいずれかの条件を満たすアプリケーションの移行はサポートされていません:

- アプリケーションは、Oracle Identity Cloud Serviceを認証に使用するように構成されています。
- アプリケーションはキャッシュ・サービスを使用します。

ほとんどのアプリケーションは、ほかのOracle Cloudサービスに接続して使用します。アプリケーションをOracle Cloud Infrastructure Container Engine for Kubernetesに移行し、Oracle Cloud Infrastructure Classicの他のOracle Cloudサービスへの接続を維持したり、他のサービスをOracle Cloud Infrastructureに移行したりできます。このガイドには、これらのサービスをOracle Cloud Infrastructure ClassicからOracle Cloud Infrastructureに移行するための詳細な手順は含まれていません。

移行プロセスの後、アプリケーションURLが変更されることを考慮する必要があります。

Oracle Cloud Infrastructureについて

Oracle Cloud Infrastructureの基本的なセキュリティ、ネットワーク、およびストレージの概念、およびOracle Cloud Infrastructure Classicに相当する概念について理解します。

Oracle Cloud Infrastructureのクラウド・リソースは論理コンパートメントで作成されます。コンパートメント内のリソースへのアクセスを制御するための、ファイングレイン・ポリシーも作成できます。

インスタンスはOracle Cloud Infrastructureリージョンで作成します。また、選択したリージョンでサポートされている場合、可用性ドメイン(AD)を指定することもできます。Oracle Cloud Infrastructure Classicは可用性ドメインを使用しません。

仮想クラウド・ネットワーク(VCN)は1つ以上のサブネットで作成され、インスタンスは特定のサブネットに割り当てられます。Oracle Cloud Infrastructure Classicでは、IPネットワークまたは共有ネットワークにインスタンスを割り当てます。通常、共有ネットワークに1つのサブネットを作成し、Oracle Cloud Infrastructure Classic内のIPネットワークごとに個別のサブネットを作成します。Oracle Cloud Infrastructure Classicとは異なり、Oracle Cloud Infrastructureではプラットフォーム・サービス用のIPアドレスを予約できません。

サブネット・セキュリティ・リストは、特定のIPアドレスおよびポートとのトラフィックを許可およびブロックします。Oracle Cloud Infrastructure Classicでは、インスタンス・アクセス・ルールは同様の機能を提供しますが、セキュリティ・リストはサブネット・レベルで構成されます。

インスタンスはOracle Cloudの外部のリソースと通信するため、Oracle Cloud Infrastructure FastConnectを使用してオンプレミス・ネットワークへの高速で専用接続を実現します。このサービスはOracle Cloud Infrastructure FastConnect Classicと同等です。または、Oracle Cloud InfrastructureのIPSec VPNを、Oracle Cloud Infrastructure ClassicのVPN as a Service (VPNaaS)またはCorenteとして使用します。

Oracle Cloud Infrastructure Object Storageのバケットを使用すると、ファイルを格納し、複数のインスタンスと共有できます。バケットにアクセスするには、ユーザー生成の認証トークン(認証トークン)が必要です。Oracle Cloud Infrastructure Object Storage ClassicはOracle Cloud Infrastructure Classicで同じサービスを提供しますが、認証トークンは使用しません。

詳細は、Oracle Cloud Infrastructureのドキュメントで[主要な概念と用語](#)を参照してください。

Oracle Cloud Infrastructure Container Engine for Kubernetesについて

Oracle Cloud Infrastructure Container Engine for Kubernetesは完全に管理され、スケーラブルかつ可用性の高いサービスであり、これを使用してコンテナ化されたアプリケーションをクラウドにデプロイできます。

開発チームがクラウドではないアプリケーションのビルド、デプロイおよび管理を確実に行う場合は、Oracle Cloud Infrastructure Container Engine for Kubernetesを使用できます。アプリケーションに必要なコンピューティング・リソースを指定し、Oracle Cloud Infrastructure Container Engine for Kubernetesは、既存テナンシにおいてこれをOracle Cloud Infrastructureにプロビジョニングします。

Kubernetesクラスタは、ノードのグループです。ノードはアプリケーションを実行しているマシンです。各ノードは、物理マシンまたは仮想マシンにすることができます。ノード容量(そのCPU数とメモリー量)は、ノードの作成時に定義されます。クラスタを複数の用途の間で分割するために、クラスタを名前空間に編成することができます。

Oracle Cloud Infrastructure Container Engine for Kubernetesの詳細は、[Kubernetesのコンテナ・エンジンの概要](#)を参照してください。

Oracle Cloud Infrastructureユーザーおよびグループについて

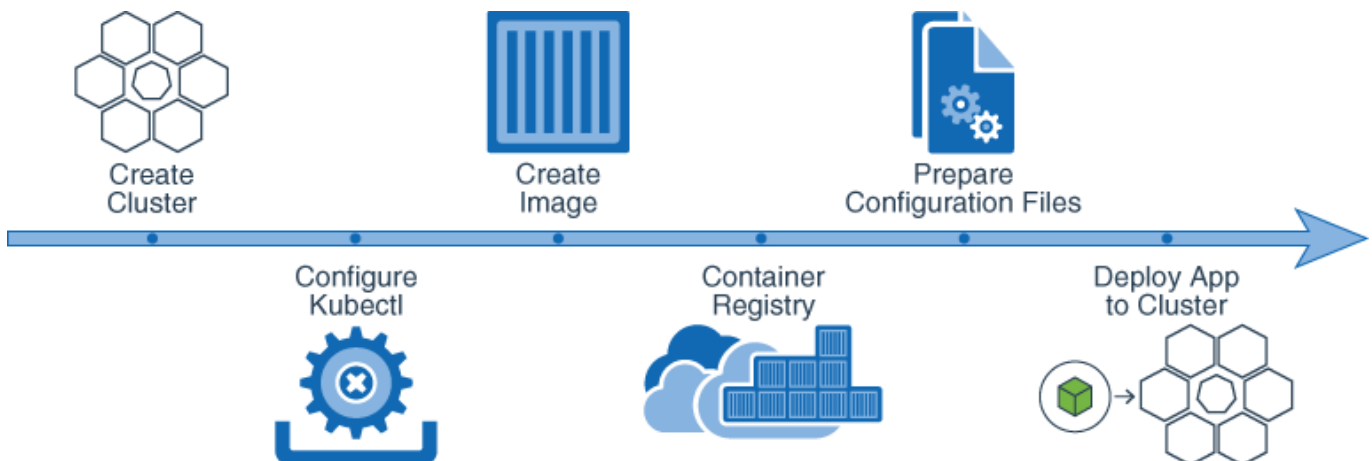
Oracle Cloud InfrastructureのIdentity and Access Management (IAM)システムを使用して、ユーザー、グループおよびポリシーを管理します。

ポリシーとは、会社が所有するOracle Cloud Infrastructureリソースにアクセスできるユーザーとその方法を指定するドキュメントです。Oracle Application Container Cloud ServiceアプリケーションをOracle Cloud Infrastructureに移行するには、Oracle Cloud Infrastructure Container Engine for KubernetesクラスタおよびOracle Cloud Infrastructureレジストリリポジトリで操作を実行するために必要なポリシーを定義する必要があります。[クラスタの作成およびデプロイメントのポリシー構成とリポジトリ・アクセスを制御するポリシー](#)を参照してください。

移行タスク・フローについて

既存のOracle Application Container Cloud ServiceアプリケーションをOracle Cloud Infrastructure Container Engine for Kubernetesに移行するプロセスを理解します。

次の図に、Oracle Application Container Cloud ServiceアプリケーションをOracle Cloud Infrastructure Container Engine for Kubernetesにデプロイするプロセスを示します。



移行プロセスには、次のタスクが含まれます:

1. Kubernetesクラスタを作成します。 Oracle Cloud Infrastructure Container Engine for Kubernetesでクラスタを作成します。
2. kubectlを構成します。 kubectlを使用してkubeconfigファイルをダウンロードし、クラスタにアクセスします。
3. Dockerイメージを作成します。 アプリケーションのランタイムに応じてテンプレートからDockerfileを作成します。
4. DockerイメージをOracle Cloud Infrastructureレジストリにプッシュ
5. Kubernetes構成を作成します。 レジストリの構成やSSLシークレットを含めて、deployment.yaml、services.yaml、およびenv.propertiesファイルを設定します。
6. アプリケーションをKubernetesクラスタにデプロイします。

2. Oracle Application Container Cloud ServiceからOracle Cloud Infrastructure Container Engine for Kubernetesへの移行準備

Oracle Application Container Cloud ServiceアプリケーションのOracle Cloud Infrastructure Container Engine for Kubernetesへの移行を計画します。

トピックス:

- 始める前に
- 停止時間要件の理解
- Oracle Cloud Infrastructureシェイプの選択

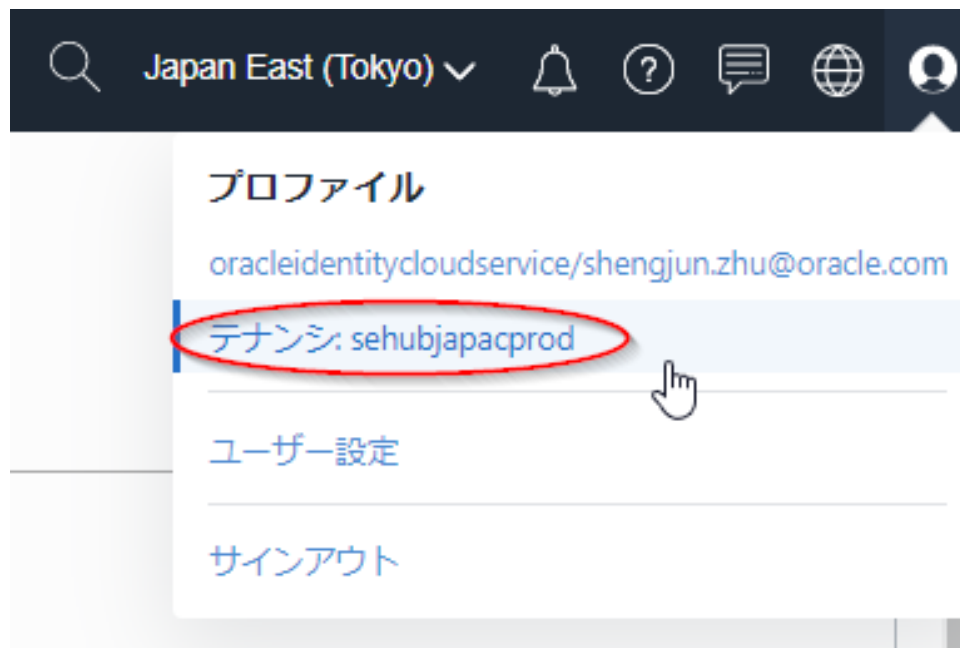
始める前に

Oracle Application Container Cloud Serviceアプリケーションの移行を開始する前に、この項で説明している特定の前提条件を満たす必要があります。

次のリソースがあることを確認します:

- Oracle Cloud Infrastructure tenancyとユーザー・アカウント
1. OCIRにログインするためにオブジェクト・ストレージ・ネームスペースを確認します。

オブジェクト・ストレージ・ネームスペースは、OCIコンソール画面右上の人型のアイコンをクリックし、展開したプロファイルからテナンシ:<テナンシ名>から確認します。



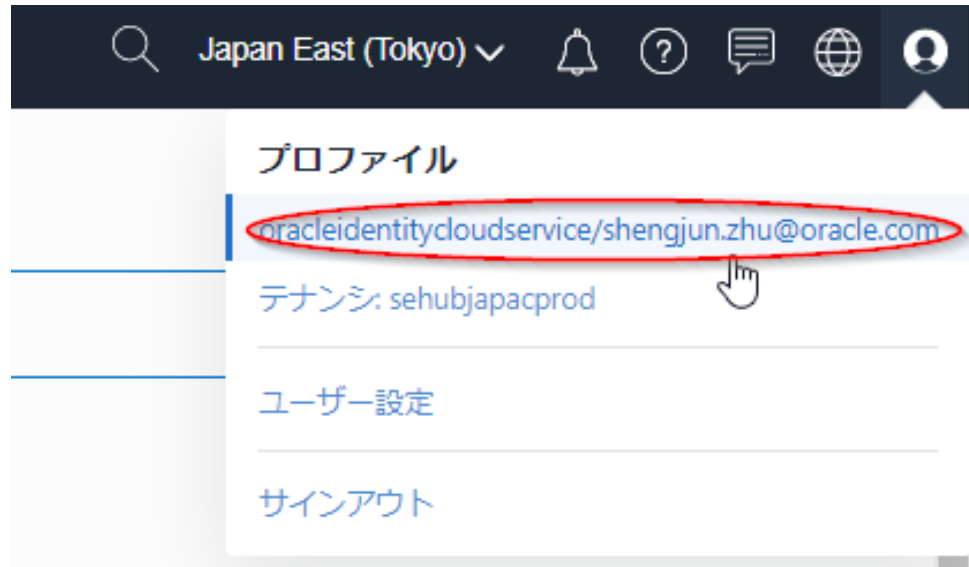
- テナント情報のオブジェクト・ストレージ設定からオブジェクト・ストレージ・ネームスペースの値を確認します。OCIRへのアクセスするために使用するため、値をテキストファイルにコピー＆ペーストするなどして控えておいてください。



注意: オブジェクト・ストレージ・ネームスペースはテナントに対し1つ割り当てられます。リージョン内のすべてのコンパートメントにまたがり使用されます。任意の文字列が設定され、変更することはできません。

- OCIRにログインするためにユーザー名を確認します。

ユーザー名は、OCIコンソール画面右上の人型のアイコンをクリックし、展開したプロフィールからユーザー名から確認します。



4. ユーザーの詳細情報からユーザー名の値を確認します。OCIRへのアクセスする際に使用するため、値をテキストファイルにコピー＆ペーストするなどして控えておいてください。



注意: IDCSでログインしたユーザー名は`oracleidentitycloudservice/<username>`の形になります。

- OCIRにログインするためには、ログイン先のレジストリを指定するにあたり、ホストされているデータセンターリージョンに合わせて適切なリージョンコードを指定する必要があります。ご自身の環境に合わせて、下表から適切なリージョンコードを見つけてください。

リージョン	リージョンコード
ap-tokyo-1	nrt
us-ashburn-1	iad
us-phoenix-1	phx
ap-mumbai-1	bom
ap-seoul-1	icn
ap-sydney-1	syd
ca-toronto-1	yyz
eu-frankfurt-1	fra
eu-zurich-1	zrh

リージョン	リージョンコード
sa-saopaulo-1	gru
uk-london-1	lhr

- 既存のコンパートメントにアクセスします。 [コンパートメントの管理](#)を参照してください
- Oracle Cloud Infrastructure Container Engine for KubernetesクラスタおよびOracle Cloud Infrastructure レジストリリポジトリで操作を実行するユーザー権限。 [クラスタの作成およびデプロイメントのポリシー構成とリポジトリ・アクセスを制御するポリシー](#)を参照してください

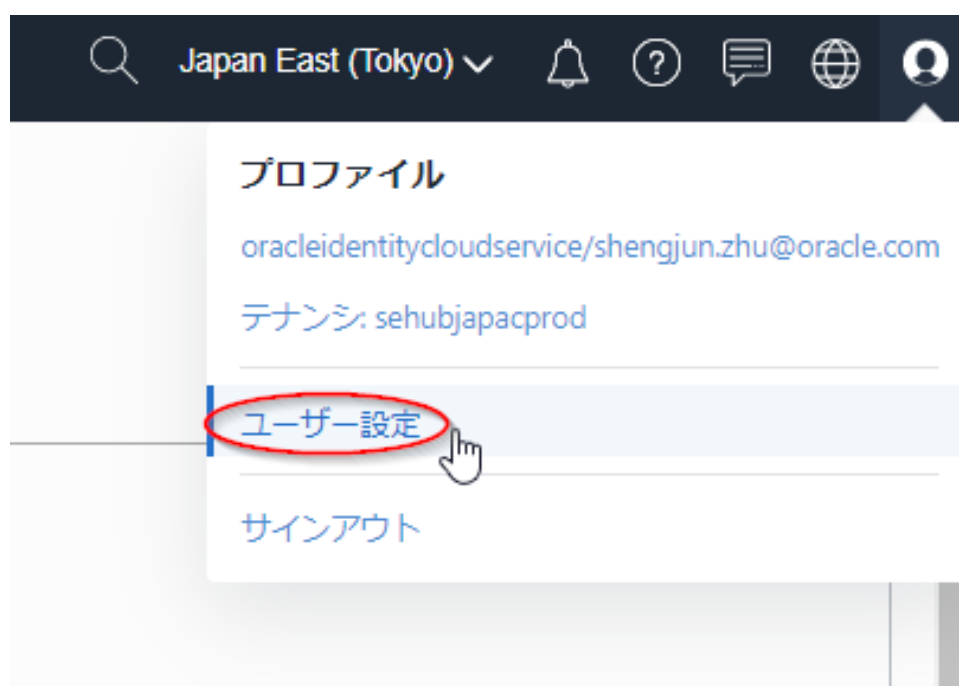
注意: スクリプト・ツールを利用する場合、OCIの管理者権限を付与する必要があります。

`ALLOW GROUP Administrators to manage all-resources IN TENANCY`

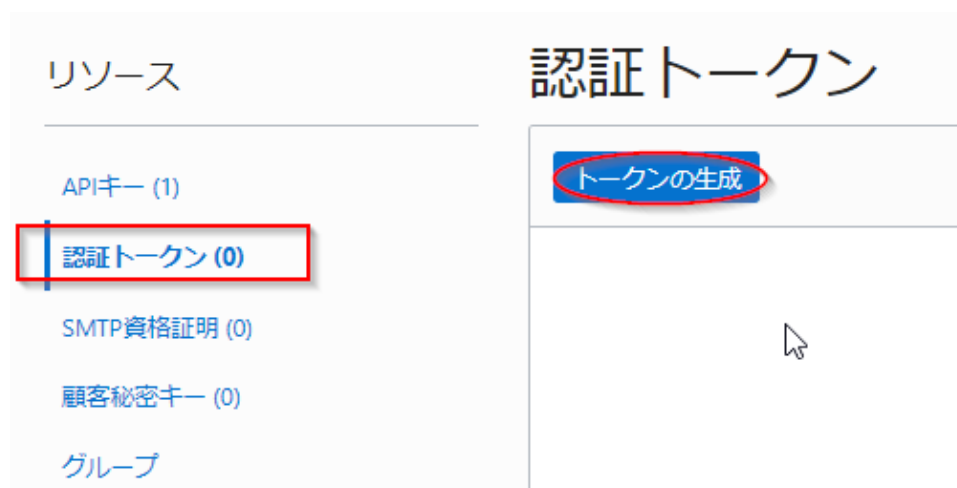
- 認証トークンの生成 [認証トークンの取得](#)を参照してください

1. OCIRにログインするために認証トークンを作成します。

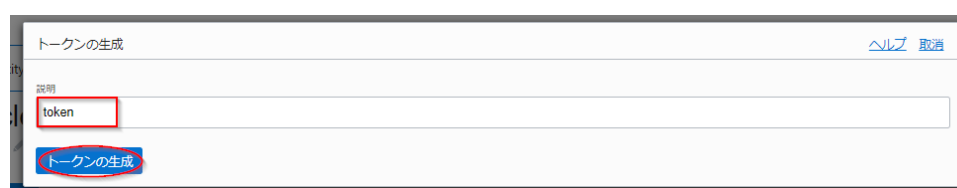
OCIコンソール画面右上の人型のアイコンをクリックし、展開したプロフィールからユーザー設定をクリックします。



2. 左側の「認証トークン」をクリックして、トークンの作成画面に遷移します。そこで「トークンの生成」ボタンをクリックします。



3. トークンの生成ダイアログで、トークンの用途を説明する情報（任意の文字列）を入力し、「トークンの生成」ボタンをクリックします。



4. ダイアログに生成したトークンが表示されます。Copyという文字列をクリックするとクリップボードにこのトークンがコピーされます。そして閉じるをクリックします。

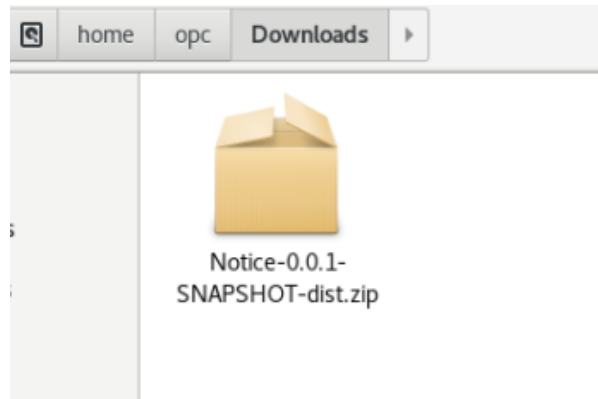


このトークンはあとの手順で利用するため、テキストエディタ等にペーストするなどして控えておいてください。

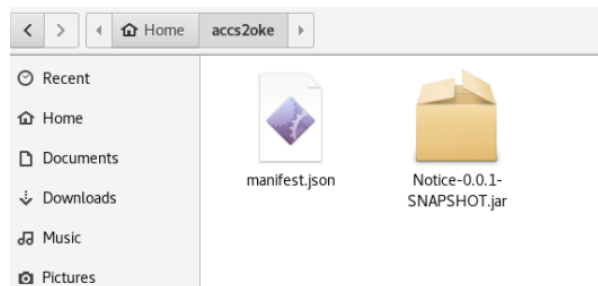
- ローカル・ディスク上のmanifest.jsonおよびdeployment.jsonファイルを含むアプリケーション・アーカイブ。移行スクリプト・ツールを使用すると、Oracle Cloud Infrastructure Object Storageのmanifest.jsonおよびdeployment.jsonファイルなどのアプリケーション・アーカイブをアップロードして、オブジェクトの認証前リクエストを生成できます。 [オブジェクトの管理](#)を参照してください。

1. 手動移行の場合、アプリケーション・アーカイブのコンテンツを抽出します。

例: 次はOracle Application Container Cloud Service用のアプリケーション・アーカイブ (JavaSE)



zipファイルを解凍し、Oracle Cloud Infrastructure Container Engine for Kubernetes用のアプリケーション・アーカイブ（manifest.jsonとJarファイル）が得られます。



2. 移行スクリプト・ツールを使用する場合、アプリケーション・アーカイブをオブジェクト・ストレージにアップロードして、オブジェクトの認証前リクエストを生成します。例: 次はOracle Application Container Cloud Service用のアプリケーション・アーカイブ（JavaSE）

ORACLE Cloud

オブジェクト・ストレージ › パケットの詳細 › 事前認証済リクエスト

ACCS2OKE

可視性の編集 リソースの移動 再暗号化 タグの追加 削除

バケット情報 タグ

可視性: プライベート
 ネームスペース: sehubjapacprod
 ストレージ層: 標準
 近似カウント: 1オブジェクト ⓘ
 ETag: 379a0032-4927-4dab-befc-6eec3eedb7e6

リソース

オブジェクト
 メトリック
 事前認証済リクエスト
 作業リクエスト
 ライフサイクル・ポリシー・ルール

事前認証済リクエスト

事前認証済リクエストの作成

名前	ステータス	アクセス・タイプ	オブジェクト名
Notice-0.0.1-SNAPSHOT-dist.zip	● アクティブ	ObjectRead	Notice-0.0.1-SNAPSHOT-dist.zip

- (オプション)移行されるアプリケーションで使用するパブリック・ドメイン名(myapp.example.com)
- (オプション)アプリケーションがSSLエンドポイントを必要とする場合、アプリケーション用のSSL証明書と秘密キーを取得する必要があります。

また、次のソフトウェアもインストールおよび構成済みであることを確認してください:

- [Dockerエンジン17.03](#)以上(ユーザーをDockerグループに追加)。 [非ルート・ユーザーとしてのDockerの管理](#)を参照)
 - [Kubectrl 1.7.4](#)以降
 - [cURL](#)または[Wget](#)
 - [Python 2.7.5, 3.5](#)以上(移行スクリプト・ツールを使用している場合はPythonが必要です)。
 - [Oracle Cloud Infrastructure CLI 2.4](#)以降
-

停止時間要件の理解

このガイドの移行プロセスは、Oracle Cloud Infrastructure Classicの既存のOracle Application Container Cloud Serviceインスタンスの可用性には影響しません。 このインスタンスは継続して実行され、このプロセス中にクライアント・リクエストを処理できます。

プロセスでは、デプロイ済のアプリケーションを変更したり、そのパフォーマンスに大幅に影響を与えることはありません。

アプリケーションが正常に移行されると、Oracle Cloud Infrastructureで実行されているアプリケーションにクライアントをルート変更できます。

Oracle Cloud Infrastructureシェイプの選択

Oracle Cloud Infrastructure Classic上のサービス・インスタンスに現在使用しているシェイプに、Oracle Cloud Infrastructure内の同様のIaaSリソースを提供するコンピュート・シェイプを特定します。

コンピュート・シェイプは、サービス・インスタンスの特定のノードで使用可能なOCPUやメモリーなどのIaaSリソースを定義します。 Oracle Cloud InfrastructureおよびOracle Cloud Infrastructure Classicのそれぞれに標準コンピュート・シェイプの独自のセットがあります。 参照:

- Oracle Cloud Infrastructure Compute Classicの使用の[シェイプについて](#)
- Oracle Cloud Infrastructureドキュメントの[シェイプのコンピュート](#)

移行したサービス・インスタンスのパフォーマンス特性が元のインスタンスと同じで、それと同等のワークロードをサポートできるようにするには、Oracle Cloud Infrastructure図形を選択します(これは、インスタンスの作成時に指定したOracle Cloud Infrastructure Classicシェイプに最も近いマップです)。

選択したシェイプがOracle Cloudテナンシで使用可能であることも確認する必要があります。 Oracleは、Oracle Cloud Infrastructureリージョン、またはリージョン内の特定の可用性ドメインのシェイプ制限を構成します。 コンソールを使用して、テナンシの現在のシェイプ制限を確認し、必要に応じて制限の増加をリクエストできます。 Oracle Cloud Infrastructureのドキュメントの[サービス制限](#)を参照してください。

3. Oracle Application Container Cloud ServiceアプリケーションのOracle Cloud Infrastructure Container Engine for Kubernetesへの移行

次のいずれかのメソッドを使用して、Oracle Application Container Cloud ServiceアプリケーションをOracle Cloud Infrastructure Container Engine for Kubernetesに移行できます:

- 移行スクリプト・ツールの使用: このスクリプトは、移行プロセスの自動化に役立ちます。アプリケーションをOracle Cloud Infrastructure Container Engine for Kubernetesに移行するために必要なリソースが自動的に作成されます。Kubernetesの経験がない場合は、可能性のあるエラーを最小限に抑えます。移行スクリプト・ツールはLinuxのみをサポートしています。
- 手動: 移行プロセスのステップを追って説明します。このメソッドを使用すると、Oracle Cloud Infrastructure Container Engine for Kubernetesでのアプリケーションのデプロイに必要なリソースをより適切に把握できます。

トピックス:

- スクリプト・ツールを使用したアプリケーションの移行
- アプリケーションの手動移行
- Kubernetes Clusterからのアプリケーション・ログの取得
- 移行問題のトラブルシューティング

スクリプト・ツールを使用したアプリケーションの移行

移行スクリプト・ツールを使用して、Oracle Application Container Cloud ServiceアプリケーションをOracle Cloud Infrastructure Container Engine for Kubernetesに移行します。移行スクリプト・ツールを使用すると、Kubernetesクラスタの作成とアプリケーションのデプロイに必要なOracle Cloud Infrastructureリソースを作成できます。また、既存のKubernetesクラスタにアプリケーションをデプロイすることもできます。

移行スクリプト・ツールを使用したアプリケーションの移行には、次のステップが含まれます:

アプリケーションの作成

1. Oracle Cloud Infrastructure Container Engine for Kubernetes内の指定のKubernetesクラスタにKubectlを構成します。Kubernetesクラスタが存在しない場合は、次の詳細でKubernetesクラスタを作成します:
 - 次を含むVCN:
 - インターネット・ゲートウェイ
 - NATゲートウェイ
 - 2つのロード・バランサ・サブネット。2つの可用性ドメインの可用性ドメインごとに1つ
 - 3人のワーカーノード・サブネット。3つの可用性ドメインの可用性ドメインごとに1つ
 - ロード・バランシングおよびワーカーノード・サブネットに対するセキュリティ・リストおよびルーティング・ルール
 - 次のようなワーカー・ノード・プール:
 - 3つのワーカーノード。3つの可用性ドメインの1つの可用性ドメイン。
 - ノード・イメージ: Oracle-Linux-7.6
 - ノード・シェイプ: VM.Standard2.1
2. アプリケーション・イメージをビルド
 - URLを指定した場合は、アプリケーション・イメージをローカル・ディスクにダウンロード
 - オプションのLinuxパッケージを含むローカルDockerイメージをビルド

- ローカルDockerイメージを「Oracle Cloud Infrastructureレジストリ」にプッシュ

3. Kubernetesクラスタ内にアプリケーションを作成

- 環境変数のConfigMapを作成
- SSLが必要な場合に、TLS証明書およびキーのシークレットを作成
- Kubernetesデプロイメントおよびサービスyaml構成を作成
- Yaml構成を使用して、アプリケーションのデプロイメントおよびサービスを作成
- HTTPからHTTPSリダイレクトまたはIP_HASHロード・バランシング・ポリシーがアプリケーションのマニフェストに構成されている場合に、Nginx ingressコントローラを設定

4. (オプション)カスタムURLを設定

- DNSゾーンを作成します(存在しない場合)。
- DNSゾーンでアプリケーションのDNSレコードを追加

Delete Application

1. DNSレコードを削除
2. Kubernetesリソース(サービス、デプロイメント、ConfigMapおよびシークレット)を削除
3. (オプション)一時ファイル(アプリケーション・アーカイブおよびコンテンツ、Dockerfile、Kubernetes構成ファイルおよびログ)を削除

トピックス:

- 移行スクリプト・ツールのダウンロードおよびインストール
- 構成ファイルの作成
- 環境変数の設定
- Java EE Systemおよびサービス・バインディング・プロパティの構成
- Kubernetes ClusterとOracle Cloud Services間の接続の有効化
- アプリケーションの作成
- アプリケーションの削除

移行スクリプト・ツールのダウンロードおよびインストール

移行スクリプト・ツールは、Oracle Application Container Cloud ServiceアプリケーションをOracle Cloud Infrastructure Container Engine for Kubernetesに移行するのに役立ちます。

移行スクリプト・ツールをダウンロードしてインストールするには、次の手順を実行します: Oracle Application Container Cloud Serviceコンソールで、「ヘルプ」をクリックして「ダウンロード・センター」をクリックし、移行スクリプト・ツールをダウンロードします。 Zipファイルの内容をディレクトリに抽出します。

The screenshot shows the Oracle Cloud Infrastructure console. The top navigation bar includes the Oracle Cloud logo and the 'Infrastructure Classic' tab. The main header is 'Oracle Application Container Cloud'. Below it, there are tabs for 'アプリケーション' (Applications) and 'アクティビティ' (Activities). The 'アプリケーション' tab is active, showing a search bar and a 'Create Application' button. A 'Profile' dropdown menu is open, showing options like 'ヘルプ' (Help), 'アクセシビリティ' (Accessibility), '情報' (Information), and 'サイン・アウト' (Sign Out). The 'ヘルプ' option is highlighted. Below the search bar, there is a 'Notice' section with details about the application, including version, deployment date, and URL. The 'Download Center' modal is open, showing a list of download links for various tools. The 'accs-migration-1.0.0' folder is highlighted in the file explorer. The file explorer shows the following files: 'accs-migration-1.0.0', 'Notice-0.0.1-SNAPSHOT-dist.zip', and 'oracle.cloud.apaas.v2.migration.accs-cli-19.2.2-1904221550.zip'.

Oracle Cloud Infrastructure Console

Infrastructure Classic

Oracle Application Container Cloud

アプリケーション アクティビティ

アプリケーション

アプリケーション名またはタグ... 失敗したアプリケーションの非表示

Create Application

Notice

バージョン: 1.0 最終デプロイ日: Sep 25, 2019 6:15:16 AM UTC 1つ以上の更新が使用可能です

ランタイム: Java SE 8u171 作成日: Sep 25, 2019 6:15:16 AM UTC メモリー: 2 GB

URL: <https://Notice-jptest01.apaas.ap5.oraclecloud.com> インスタンス: 2

Download Center

This is a download page for tools required by some of the Oracle Cloud Platform services. Each tool indicates the applicable services. Click the download image on the right and follow the instructions.

Oracle offers a PaaS Service Manager (PSM) Command Line Interface (CLI) that enables users of Oracle Cloud Platform Services to create, monitor and manage their service instances from a command shell or script. For more details click [here](#).

Applies to: All Oracle Cloud Platform Services File Size: 46K

AppToCloud allows users to export an existing Oracle WebLogic Server domain configuration, including resources and deployed Java applications, and to then provision a new Oracle Java Cloud Service instance. This allows you to migrate a WebLogic domain running anywhere to Java Cloud Service. For more details click [here](#).

Applies to: Oracle Java Cloud Service

This utility allows users to migrate an application from OCI-C to OCI. This will create an instance of the application in an OKE cluster in OCI. After migration, you will need to clean up the instance running in OCI-C manually. For more details click [here](#).

Applies to: Application Container Service

accs-migration-1.0.0

Recent Home Documents Downloads Music Pictures Videos Trash

accs-migration-1.0.0 Notice-0.0.1-SNAPSHOT-dist.zip oracle.cloud.apaas.v2.migration.accs-cli-19.2.2-1904221550.zip

コマンドライン・ウィンドウを開き、コンテンツを抽出したディレクトリに移動します。 ヘルプ・コマンドを実行します。

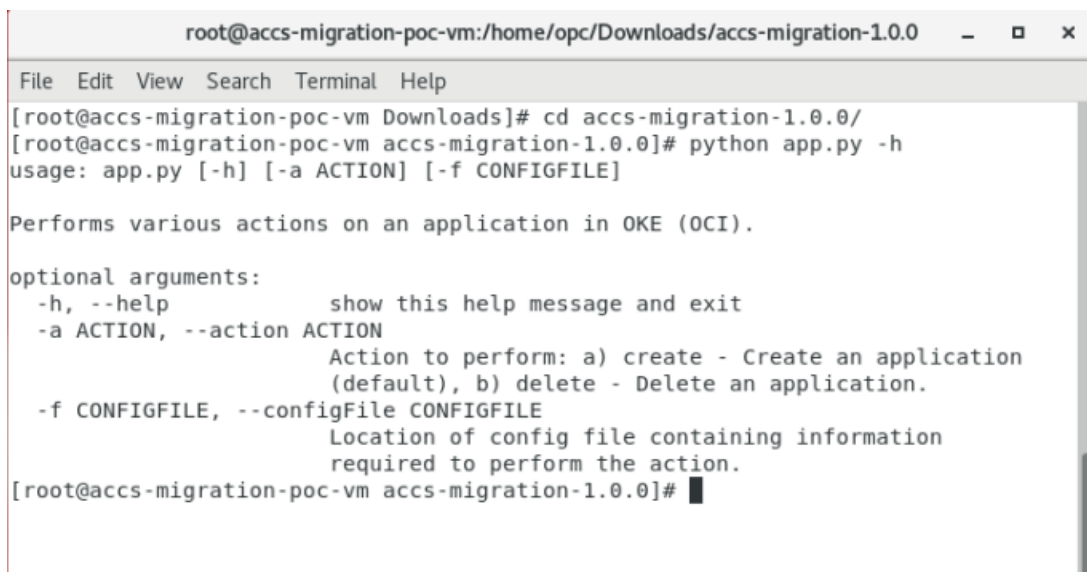
```
python app.py -h
```

例:

```
$ python app.py -h
usage: app.py [-h] [-a ACTION] [-f CONFIGFILE]

Performs various actions on an application in OKE (OCI).

optional arguments:
  -h, --help            show this help message and exit
  -a ACTION, --action ACTION
                        Action to perform: a) create - Create an application
                        (default), b) delete - Delete an application.
  -f CONFIGFILE, --configFile CONFIGFILE
                        Location of config file containing information
                        required to perform the action.
```

A screenshot of a terminal window titled 'root@accs-migration-poc-vm:/home/opc/Downloads/accs-migration-1.0.0'. The terminal shows the command 'cd accs-migration-1.0.0/' followed by 'python app.py -h'. The output is the same as the previous block, showing the usage and optional arguments for the application. The terminal has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'.

```
root@accs-migration-poc-vm:/home/opc/Downloads/accs-migration-1.0.0
File Edit View Search Terminal Help
[root@accs-migration-poc-vm Downloads]# cd accs-migration-1.0.0/
[root@accs-migration-poc-vm accs-migration-1.0.0]# python app.py -h
usage: app.py [-h] [-a ACTION] [-f CONFIGFILE]

Performs various actions on an application in OKE (OCI).

optional arguments:
  -h, --help            show this help message and exit
  -a ACTION, --action ACTION
                        Action to perform: a) create - Create an application
                        (default), b) delete - Delete an application.
  -f CONFIGFILE, --configFile CONFIGFILE
                        Location of config file containing information
                        required to perform the action.
[root@accs-migration-poc-vm accs-migration-1.0.0]#
```

構成ファイルの作成

移行スクリプト・ツールを使用して、アプリケーションを移行するための構成ファイルを作成する必要があります。このファイルは、Oracle Cloud Infrastructure Container Engine for Kubernetesでアプリケーションを作成するために必要であり、JSON形式の情報が含まれます。

構成ファイルを作成するには、次のテンプレートを使用します:

```
{
  "account": {
    "authToken": "<auth-token>",
    "profile": "<oci-profile-name>",
    "OCIConfig": "<oci-config-path>"
  },
}
```

```

"application": {
  "name": "<application-name>",
  "runtime": "<application-runtime>",
  "source": "<application-archive-location>",
  "manifest": "<manifest-file>",
  "deployment": "<deployment-file>",
  "ssl": {
    "tlsKey": "<TLS-key>",
    "tlsCert": "<TLS-certificate>"
  },
  "environment-variables":{
    "<env-var-key1>": "<env-var-value1>",
    "<env-var-key2>": "<env-var-value2>"
  }
},
"cluster": {
  "compartmentOCID": "<compartmentOCID>",
  "name": "<oke-cluster-name>",
  "sshPublicKey": "<sshPublicKey>"
},
"dnszone": {
  "name": "<dnszone-name>",
  "compartmentOCID": "<compartmentOCID-zone>"
}
}

```

テンプレートには次のセクションが含まれています:

- アカウント: Oracle Cloud Infrastructure アカウントのユーザー情報が含まれています。
- アプリケーション: アプリケーションの詳細が含まれています。
- クラスタ: Oracle Cloud Infrastructure Container Engine for Kubernetes クラスタの詳細が格納されます。既存のクラスタを使用することも、クラスタが存在しない場合に作成することもできます。
- Dnszone: このセクションはオプションで、カスタムURLのDNSゾーンの詳細を含みます。
dnszone.name プロパティで指定されているDNSゾーンが存在しない場合は作成されます。アプリケーションURLは次のように生成されます: `http[s]://<application.name>.<dnszone.name>`。

次の表を使用して、構成ファイルのプレースホルダーを置換します:

プレースホルダー	説明	必須
auth-token	Oracle Cloud Infrastructure 認証トークン。	Yes
oci-config-path	Oracle Cloud Infrastructure 構成ファイルのパス。 デフォルト値: <code>~/oci/config</code> 。	No
oci-profile-name	Oracle Cloud Infrastructure 構成プロファイル名。 デフォルト値: <code>DEFAULT</code> 。	No
application-name	アプリケーションの名前。	Yes
application-runtime	アプリケーション・ランタイム: java, node, php, javaee, dotnet, ruby, python, または golang。	Yes

プレースホルダー	説明	必須
application-archive-location	Oracle Cloud Infrastructure Object Storageのアプリケーション・アーカイブの認証前リクエストURLまたはローカル・ディスクのアプリケーション・アーカイブのロケーション。	Yes
manifest-file	manifest.jsonファイルのパス。 manifest.jsonファイルのパスを指定した場合、アプリケーション・アーカイブのmanifest.jsonファイルは無視されます。	No
deployment-file	deployment.jsonファイルのパス。 deployment.jsonファイルのパスが指定されている場合、アプリケーション・アーカイブのdeployment.jsonファイルは無視されます。注意: "instances"を数字として定義する必要があります。例: {"instances": 2}	No
env-var-keyNとenv-var-valueN	アプリケーションによって使用されるカスタム環境変数です。	No
TLS-certificate	ワーカーノードのSSH公開キー・ファイルのパス。 このSSH公開キーを持つワーカーノードにアクセスするには、ベース・ホストを設定する必要があります。 ホスト を参照してください。	application.sslセクションを指定した場合は必須です。
TLS-key	TLSキー・ファイルのパス。	application.sslセクションを指定した場合は必須です。
compartmentOCID	クラスタが存在するか、作成する必要があるコンパートメントのOCID。	Yes
oke-cluster-name	クラスタの名前（15文字以内）。	Yes
sshPublicKey	ワーカーノードのSSH公開キー・ファイルのパス。	No
compartmentOCID-zone	DNSゾーンが存在するコンパートメントのOCID、または作成する必要があります。	dnszoneセクションを指定した場合は必須です。
dnszone-name	DNSゾーンの名前。	dnszoneセクションを指定した場合は必須です。

例:(JAVA)

```
{
  "account": {
    "authToken": "6h.}82DrE4+k#sfDa)3<",
    "profile": "DEFAULT",
    "OCIConfig": "~/.oci/config"
```



```

    },
    "application": {
      "name": "notice2",
      "runtime": "java",
      "source": "https://objectstorage.us-ashburn-
1.oraclecloud.com/p/RVQB9JpnLHUCtgHj0SC90SZqWQSP5qbeQf1NeT_zBk/n/sehubjapacprod/b
/ACCS20KE/o/Notice-0.0.1-SNAPSHOT-dist.zip",
      "deployment": "/home/opc/Downloads/accs-migration-
1.0.0/deployment.json",
      "environment-variables": {
        "DBAAS_DEFAULT_CONNECT_DESCRIPTOR":
"146.56.2.52:1521/PDB1.jpctest01.oraclecloud.internal",
        "DBAAS_USER_NAME": "oracleusr2",
        "DBAAS_USER_PASSWORD": "*****",
        "DBAAS_LISTENER_HOST_NAME": "146.56.2.52",
        "DBAAS_LISTENER_PORT": "1521",
        "DBAAS_DEFAULT_SID": "ORCL",
        "DBAAS_DEFAULT_SERVICE_NAME": "PDB1.jpctest01.oraclecloud.internal"
      }
    },
    "cluster": {
      "compartmentOCID":
"ocid1.compartment.oc1..aaaaaaaajgkfq6r4xflex534ouhycro3rxflwcnwp4aenvrvjygg62y2ar
6a",
      "name": "mycluster2",
      "sshPublicKey": "/home/opc/Downloads/accs-migration-
1.0.0/public.pub"
    },
    "dnszone": {
      "name": "taosheng.tk",
      "compartmentOCID":
"ocid1.compartment.oc1..aaaaaaaajgkfq6r4xflex534ouhycro3rxflwcnwp4aenvrvjygg62y2ar
6a"
    }
  }
}

```

例:(JAVAEE+SSL証明書)

```

{
  "account": {
    "authToken": "6h.}82DrE4+k#sfDa)3<",
    "profile": "DEFAULT",
    "OCIConfig": "~/oci/config"
  },
  "application": {
    "name": "employees",
    "runtime": "javaee",
    "source": "https://objectstorage.us-ashburn-
1.oraclecloud.com/p/gm9AuY1Hr-
WgsQI6dgfqSzna0kbzL_WIa7b0JJFGZ_g/n/sehubjapacprod/b/ACCS20KE/o/employees-
app.war",

```

```

        "deployment": "/home/opc/Downloads/accs-migration-
1.0.0/deployment.json",
        "ssl": {
            "tlsKey": "/home/opc/Downloads/JAVAEE/key/server2.key",
            "tlsCert": "/home/opc/Downloads/JAVAEE/key/server.crt"
        },
        "environment-variables": {
            "DBAAS_DEFAULT_CONNECT_DESCRIPTOR":
"146.56.2.52:1521/PDB1.jpctest01.oraclecloud.internal",
            "DBAAS_USER_NAME": "oracleusr2",
            "DBAAS_USER_PASSWORD": "*****",
            "DBAAS_LISTENER_HOST_NAME": "146.56.2.52",
            "DBAAS_LISTENER_PORT": "1521",
            "DBAAS_DEFAULT_SID": "ORCL",
            "DBAAS_DEFAULT_SERVICE_NAME": "PDB1.jpctest01.oraclecloud.internal",
            "EXTRA_JAVA_PROPERTIES": "-DconfigPath=/u01/app/conf/ -
Dlogfile=/u01/app/logs/app.log",
            "DBAAS_SERVICE_BINDING_NAME": "DBCSDemo",
            "DBAAS_PROPERTIES": "jndi-name:jdbc/testds|max-capacity:5|min-
capacity:1|"
        },
        "cluster": {
            "compartmentOCID":
"ocid1.compartment.oc1..aaaaaaaajgkf6r4xflex534ouhycro3rxflwcnwp4aenvrvjygg62y2ar
6a",
            "name": "employees",
            "sshPublicKey": "/home/opc/Downloads/accs-migration-
1.0.0/public.pub"
        },
        "dnszone": {
            "name": "taosheng.tk",
            "compartmentOCID":
"ocid1.compartment.oc1..aaaaaaaajgkf6r4xflex534ouhycro3rxflwcnwp4aenvrvjygg62y2ar
6a"
        }
    }
}

```

スクリプト・ツールのデフォルト設定の変更

移行スクリプト・ツールの配下にある設定ファイル(source/resources/cluster_default_inputs.json)を必要に応じて変更する必要があります。

注意: `cidr_block_worker` (ワーカーノードのサブネット)、`cidr_block_loadbalancer` (ロードバランシングのサブネット)、`nodes_per_subnet` (サブネットごとのワーカーノード数) は、リージョン内の可用性ドメイン全体 (または、東京リージョンなど単一可用性ドメインの場合、その可用性ドメイン内の障害ドメイン全体) に可能な限り均等に分散されます。実運用の際は可用性を考慮し、適切な値を指定してください。

注意: 最新版(2019/10/25)OCIでは、イメージのバージョン `Oracle-Linux-7.5` が利用不可になりましたので、利用可能なバージョンに修正してください。

cluster_default_inputs.jsonの修正例（東京リージョン）：

```
{
  "cidr_block_vcn": "10.0.0.0/16",
  "cidr_block_worker": [
    "10.0.10.0/24"
  ],
  "cidr_block_loadbalancer": [
    "10.0.20.0/24"
  ],
  "work_req_polling_time_seconds": 60,
  "work_req_polling_count": 60,
  "nodepool_name": "pool1",
  "node_image_name": "Oracle-Linux-7.6",
  "node_shape": "VM.Standard2.1",
  "nodes_per_subnet": 3,
  "worker_egress_rules_file": "source/resources/worker-egress-rule.json",
  "worker_ingress_rules_file": "source/resources/worker-ingress-rule.json",
  "lb_egress_rules_file": "source/resources/lb-egress-rule.json",
  "lb_ingress_rules_file": "source/resources/lb-ingress-rule.json",
  "worker_subnet_display_name": "oke-subnet-<cluster_name>-<ad>",
  "lb_subnet_display_name": "oke-svclbsubnet-<cluster_name>-<ad>",
  "worker_sec_list_display_name": "oke-wkr-seclist-<cluster_name>",
  "lb_sec_list_display_name": "oke-lb-seclist-<cluster_name>",
  "kubeconfig_file": "~/kube/config"
}
```

環境変数の設定

Oracle Application Container Cloud Serviceアプリケーションに必要な環境変数をKubernetes構成に移行します。

カスタム環境変数

アプリケーションのこれらの変数は、deployment.jsonファイルで定義しました。移行スクリプトによってdeployment.jsonファイルが読み取られ、その変数がKubernetes構成に自動的に追加されます。

サービス・バインド環境変数

Oracle Application Container Cloud Serviceでは、アプリケーションでサービス・バインディングを追加すると、これらの変数が自動的に作成されます。アプリケーションで1つ以上のサービス・バインディングが構成されている場合は、各サービス・バインディングの環境変数を特定します。

サービス・バインディング環境変数を構成する手順は、次のとおりです：

Oracle Application Container Cloud Serviceコンソールでアプリケーションにアクセスします。「デプロイメント」タブをクリックし、「環境変数」セクションでサービス・バインディング変数を識別します。構成ファイルを編集し、各変数を"key": "value"として別の行に追加します。例：

```

    "environment-variables":{
      "DBAAS_DEFAULT_CONNECT_DESCRIPTOR":
"146.56.2.52:1521/PDB1.jpctest01.oraclecloud.internal",
      "DBAAS_USER_NAME": "oracleusr2",
      "DBAAS_USER_PASSWORD": "*****",
      "DBAAS_LISTENER_HOST_NAME": "146.56.2.52",
      "DBAAS_LISTENER_PORT": "1521",
      "DBAAS_DEFAULT_SID": "ORCL",
      "DBAAS_DEFAULT_SERVICE_NAME": "PDB1.jpctest01.oraclecloud.internal"
    }

```

Java EE Systemおよびサービス・バインディング・プロパティの構成

Java EEアプリケーションで、システムおよびJNDIサービス・バインディング・プロパティを使用する場合、構成ファイルの環境変数セクションでこれらのプロパティを指定できます。

システム・プロパティ

Java EEアプリケーションでは、構成ファイルでEXTRA_JAVA_PROPERTIES環境変数を使用してシステム・プロパティを構成できます。

1. Java EEアプリケーションのシステム・プロパティを指定します。
2. 構成ファイルに、EXTRA_JAVA_PROPERTIESという名前の環境変数を追加します。この変数の値内で、-Dフラグを使用して各システム・プロパティを指定します。各-Dフラグは空白で区切ります。例:

```

"EXTRA_JAVA_PROPERTIES": "-DconfigPath=/u01/app/conf/ -
Dlogfile=/u01/app/logs/app.log"

```

サービス・バインディング・プロパティ

Java EEアプリケーションで1つ以上のJNDIサービス・バインディング・プロパティ、つまりjndi-name, max-capacity, min-capacity,またはdriver-properties,を使用している場合、それらを構成ファイルに追加する必要があります。

1. アプリケーションで使用するサービス・バインディングを指定してください。
2. 構成ファイルを編集します。
3. キーを使用してサービスの環境変数を追加します:

```

"<service-type>_SERVICE_BINDING_NAME": "<service-name>",

```

説明:

- はサービス・タイプです。たとえば、DBAAS, MYSQLCSなどです。
- は、サービスの名前です。例:

```
"DBAAS_SERVICE_BINDING_NAME": "testDb"
```

4. キーを使用して、サービスのJNDIサービス・バインディング・プロパティを指定します:

```
"<service-type>_PROPERTIES": "jndi-name:<jndi-name>|max-capacity:<max-capacity>|min-capacity:<min-capacity>|driver-properties:<driver-properties>|"
```

説明:

- はサービス・タイプです。たとえば、DBAAS, MYSQLCSなどです。
- は、サービスのJNDI名です。"jdbc/"形式である必要があります。例: "jdbc/dbcs".
- は、接続プールの最大容量です。
- は、接続プールの最小容量です。
- は、JDBCドライバに渡されるプロパティのセミコロン区切りリストです。

例:

```
"environment-variables": {
  "MYSQLCS_CONNECT_STRING": "10.x.x.x:3306/mydb",
  "MYSQLCS_MYSQL_PORT": "3306",
  "MYSQLCS_USER_PASSWORD": "<your_password>",
  "MYSQLCS_USER_NAME": "TestUser",
  "DBAAS_DEFAULT_CONNECT_DESCRIPTOR": "10.x.x.x:1521/mydb",
  "DBAAS_USER_NAME": "TestUser",
  "DBAAS_USER_PASSWORD": "<your_password>",
  "DBAAS_LISTENER_HOST_NAME": "10.x.x.x",
  "DBAAS_LISTENER_PORT": "1521",
  "DBAAS_DEFAULT_SID": "ORCL",
  "DBAAS_DEFAULT_SERVICE_NAME": "mydb",
  "EXTRA_JAVA_PROPERTIES": "-DconfigPath=/u01/app/conf/ -
DlogFile=/u01/app/logs/app.log",
  "DBAAS_SERVICE_BINDING_NAME": "dbaasDb",
  "DBAAS_PROPERTIES": "jndi-name:jdbc/dbcs|max-capacity:5|min-capacity:1|driver-
properties:user=admin;database=test|",
  "MYSQLCS_SERVICE_BINDING_NAME": "mysqlDb",
  "MYSQLCS_PROPERTIES": "jndi-name:jdbc/mysqlcs|max-capacity:10|min-
capacity:1|driver-properties:user=oci;database=app|"
}
```

Kubernetes ClusterとOracle Cloud Services間の接続の有効化

Oracle Application Container Cloud Serviceのアプリケーションがサービス・バインディングを使用して、他のOracle Cloudサービスとの通信を有効にする場合、アプリケーションの移行後にそれらのサービスと通信できるようにする必要があります。

次の2つのシナリオがあります:

- 使用しているサービスがOracle Cloud Infrastructure Classicにある場合は、Oracle Cloud Infrastructure Classicサービスで、Kubernetesクラスタの就業者ノードにアタッチされているNAT GatewayのパブリックIPアドレスをサービスに接続できるようにするアクセス・ルールを作成します。たとえば、アプリケーションでOracle Database Cloud Serviceを使用する場合は、「Database Cloud Serviceへのネットワーク・アクセスの管理」を参照してください。

注意: NAT GatewayのパブリックIPアドレスは、メニューからOracle Cloud Infrastructureコンソールに配置できます: **開発者サービス、コンテナ・クラスタ(OKE)、クラスタ詳細、ノード・プール・セクション、ノード・インスタンスの詳細、仮想クラウド・ネットワーク詳細、NATゲートウェイ、パブリックIPアドレス。**

- サービスがOracle Cloud Infrastructureにある場合、サービスがデプロイされているVCNおよびサブネットを探します。Kubernetesクラスタからサービスへのトラフィックを可能にするために、インGRESS・セキュリティ・ルールが存在することを確認してください。Oracle Cloud Infrastructureのドキュメントの[セキュリティ・リスト](#)を参照してください。

アプリケーションの作成

移行スクリプト・ツールで作成アクションを使用して、Oracle Application Container Cloud ServiceアプリケーションをOracle Cloud Infrastructure Container Engine for Kubernetesに移行します。

1. アプリケーションを作成するには、コマンドライン・ウィンドウを開き、移行スクリプトを実行します。構成ファイルへのパスを指定します。

```
python app.py -f <path-of-config-file>
```

2. コマンドライン・ウィンドウで、YESと入力してクラスタを作成します。既存のクラスタを使用している場合、このメッセージは表示されません。
3. DNSゾーンを作成するには、YESを入力します。既存のDNSゾーンを使用している場合、または指定していない場合、このメッセージは表示されません。

例:

```
[root@accs2oke accs-migration-1.0.0]# python app.py -f ./config.json
2019-10-25 05:54:01,328 root INFO Using default action: create
2019-10-25 05:54:01,328 root INFO Starting application creation.
2019-10-25 05:54:01,329 root INFO URL setup is enabled. Checking if zone name and
compartment OCID is given.
2019-10-25 05:54:01,330 root INFO Fetching OCI details....
2019-10-25 05:54:05,205 root INFO Checking OKE cluster existence in OCI with
provided name: mycluster4
The OKE Cluster with name : mycluster4 does not exist in OCI and it needs to be
created to proceed further. Do you want to continue ? Yes/No: Yes
2019-10-25 05:54:14,554 root INFO Creating VCN...
2019-10-25 05:54:16,008 root INFO VCN successfully created with id :
ocid1.vcn.oc1.ap-tokyo-
```

```
1.aaaaaaaawbz5stwpzpv26wepzu6ss5bue7564qbdugfjwncksxx5mgssbga
2019-10-25 05:54:16,009 root INFO Creating internet gateway...
2019-10-25 05:54:17,553 root INFO Internet Gateway successfully created with id
ocid1.internetgateway.oc1.ap-tokyo-
1.aaaaaaaar552twpc4hyjerqwlbieyqvm7ufрту4rdxww7v6mxmdwbip4k3jq
2019-10-25 05:54:17,553 root INFO Creating nat gateway...
2019-10-25 05:54:19,693 root INFO NAT Gateway successfully created with id
ocid1.natgateway.oc1.ap-tokyo-
1.aaaaaaaajfdhbc7bs24lcjpnbc34f5t42giitlflawq34lvc1545m4etgq
2019-10-25 05:54:19,694 root INFO Creating Route table...
2019-10-25 05:54:19,694 root INFO Creating Route rules...
2019-10-25 05:54:19,694 root INFO Successfully created route rules: [{"cidrBlock":
"0.0.0.0/0", "networkEntityId": "ocid1.internetgateway.oc1.ap-tokyo-
1.aaaaaaaar552twpc4hyjerqwlbieyqvm7ufрту4rdxww7v6mxmdwbip4k3jq"}]]
2019-10-25 05:54:21,201 root INFO Route table successfully created with id :
ocid1.routetable.oc1.ap-tokyo-
1.aaaaaaa3dbxnjmqhgnkb6rbqhytwrrsndgiceycoyemshsh3ni2qfa6xuq
2019-10-25 05:54:21,201 root INFO Creating Route table...
2019-10-25 05:54:21,201 root INFO Creating Route rules...
2019-10-25 05:54:21,202 root INFO Successfully created route rules: [{"cidrBlock":
"0.0.0.0/0", "networkEntityId": "ocid1.natgateway.oc1.ap-tokyo-
1.aaaaaaaajfdhbc7bs24lcjpnbc34f5t42giitlflawq34lvc1545m4etgq"}]]
2019-10-25 05:54:22,598 root INFO Route table successfully created with id :
ocid1.routetable.oc1.ap-tokyo-
1.aaaaaaaajq5b5h6f7wykzqydszaith2vsunpusrmdicnr552daiw6oeb5x5a
2019-10-25 05:54:22,598 root INFO Creating security list for worker nodes...
2019-10-25 05:54:23,949 root INFO Security list successfully created with id
ocid1.securitylist.oc1.ap-tokyo-
1.aaaaaaaagowwsj3nbmchliyz5c2ve7e274pramkaictng27ww7nxk4gqwjq
2019-10-25 05:54:23,949 root INFO Creating security list for loadbalancer...
2019-10-25 05:54:25,305 root INFO Security list successfully created with id
ocid1.securitylist.oc1.ap-tokyo-
1.aaaaaaaag5efx444nmp2leipygbhppmh4osbl4g2iyv4c6oi3en5c5353v7q
2019-10-25 05:54:25,305 root INFO Getting availability domains for the compartment
2019-10-25 05:54:26,589 root INFO Successfully fetched availability domains for
the compartment.
2019-10-25 05:54:26,590 root INFO Number of subnets to be created for worker nodes
: 1
2019-10-25 05:54:26,590 root INFO Creating worker subnet 1
2019-10-25 05:54:26,590 root INFO Creating subnet...
2019-10-25 05:54:26,590 root INFO Creating subnet with security list:
["ocid1.securitylist.oc1.ap-tokyo-
1.aaaaaaaagowwsj3nbmchliyz5c2ve7e274pramkaictng27ww7nxk4gqwjq"]
2019-10-25 05:54:28,059 root INFO Subnet id is ocid1.subnet.oc1.ap-tokyo-
1.aaaaaaa34pkplxkjddoxw5v3v2jki24xi5uiu4jlugsaetz4zxxvnycpaq
2019-10-25 05:54:28,059 root INFO Number of subnets to be created for loadbalancer
: 1
2019-10-25 05:54:28,059 root INFO Creating loadbalancer subnet 1
2019-10-25 05:54:28,060 root INFO Creating subnet...
2019-10-25 05:54:28,060 root INFO Creating subnet with security list:
["ocid1.securitylist.oc1.ap-tokyo-
1.aaaaaaaag5efx444nmp2leipygbhppmh4osbl4g2iyv4c6oi3en5c5353v7q"]
2019-10-25 05:54:29,741 root INFO Subnet id is ocid1.subnet.oc1.ap-tokyo-
1.aaaaaaaawgzqzyfesbrtrn7jn43hsgnqzift6zxjymupsag3ftqlqja2ipzq
```



```
2019-10-25 05:54:29,741 root INFO Fetching latest kubernetes version...
2019-10-25 05:54:31,005 root INFO The latest Kubernetes version available is
v1.13.5
2019-10-25 05:54:31,005 root INFO Creating cluster...
2019-10-25 05:54:32,745 root INFO Work request ID to create cluster is:
ocid1.clustersworkrequest.oc1.ap-tokyo-
1.aaaaaaaaafswiobug5rdimbqgqzdomlegzsweszjyga3dqzbugwzdsyzsmezt
2019-10-25 05:54:32,746 root INFO Monitoring work request
ocid1.clustersworkrequest.oc1.ap-tokyo-
1.aaaaaaaaafswiobug5rdimbqgqzdomlegzsweszjyga3dqzbugwzdsyzsmezt
2019-10-25 05:54:34,022 root INFO The status is IN_PROGRESS
2019-10-25 05:54:34,022 root INFO Checking the status again in 60 seconds.
2019-10-25 05:55:35,353 root INFO The status is IN_PROGRESS
2019-10-25 05:55:35,354 root INFO Checking the status again in 60 seconds.
2019-10-25 05:56:36,659 root INFO The status is IN_PROGRESS
2019-10-25 05:56:36,660 root INFO Checking the status again in 60 seconds.
2019-10-25 05:57:38,029 root INFO The status is IN_PROGRESS
2019-10-25 05:57:38,030 root INFO Checking the status again in 60 seconds.
2019-10-25 05:58:39,329 root INFO The status is SUCCEEDED
2019-10-25 05:58:39,330 root INFO Cluster successfully created with id :
ocid1.cluster.oc1.ap-tokyo-
1.aaaaaaaaafrdnbgzsdoolfgm3genlbmizdntggrqtondgmccqtamjqgjsjg
2019-10-25 05:58:42,695 root INFO Work request ID to create node pool is:
ocid1.clustersworkrequest.oc1.ap-tokyo-
1.aaaaaaaaae2dkzjwga3wkobxgy4wiojwhfsdenjvme3dcnldgw3tomdegq4t
2019-10-25 05:58:42,695 root INFO Monitoring work request
ocid1.clustersworkrequest.oc1.ap-tokyo-
1.aaaaaaaaae2dkzjwga3wkobxgy4wiojwhfsdenjvme3dcnldgw3tomdegq4t
2019-10-25 05:58:43,966 root INFO The status is SUCCEEDED
2019-10-25 05:58:43,967 root INFO Node pool successfully created with id:
ocid1.nodepool.oc1.ap-tokyo-
1.aaaaaaaaaeywgm3dhaytcylgdm2gczzjmyytanteg4ytsyzwmn4dqzjxg5td
2019-10-25 05:58:43,967 root INFO Checking worker nodes status for ACTIVE state
...
2019-10-25 05:58:45,341 root INFO Nodes data is not available yet, will retry in
60 seconds.
2019-10-25 05:59:46,651 root INFO Checking for worker node: oke-cqtamjqgjsjg-
n4dqzjxg5td-szxxvnycpaq-2
2019-10-25 05:59:46,651 root INFO Checking for worker node: oke-cqtamjqgjsjg-
n4dqzjxg5td-szxxvnycpaq-1
2019-10-25 05:59:46,652 root INFO Checking for worker node: oke-cqtamjqgjsjg-
n4dqzjxg5td-szxxvnycpaq-0
2019-10-25 05:59:46,652 root INFO Nodes are not in ACTIVE state, will retry in 60
seconds.
2019-10-25 06:00:47,934 root INFO Checking for worker node: oke-cqtamjqgjsjg-
n4dqzjxg5td-szxxvnycpaq-2
2019-10-25 06:00:47,934 root INFO Checking for worker node: oke-cqtamjqgjsjg-
n4dqzjxg5td-szxxvnycpaq-1
2019-10-25 06:00:47,935 root INFO Checking for worker node: oke-cqtamjqgjsjg-
n4dqzjxg5td-szxxvnycpaq-0
2019-10-25 06:00:47,935 root INFO Nodes are not in ACTIVE state, will retry in 60
seconds.
2019-10-25 06:01:49,211 root INFO Checking for worker node: oke-cqtamjqgjsjg-
n4dqzjxg5td-szxxvnycpaq-2
```



```
2019-10-25 06:01:49,212 root INFO Checking for worker node: oke-cqtamjqgjs-g-
n4dqzjxg5td-szxxvnycpaq-1
2019-10-25 06:01:49,212 root INFO Checking for worker node: oke-cqtamjqgjs-g-
n4dqzjxg5td-szxxvnycpaq-0
2019-10-25 06:01:49,212 root INFO Nodes are not in ACTIVE state, will retry in 60
seconds.
2019-10-25 06:02:50,550 root INFO Checking for worker node: oke-cqtamjqgjs-g-
n4dqzjxg5td-szxxvnycpaq-1
2019-10-25 06:02:50,551 root INFO Checking for worker node: oke-cqtamjqgjs-g-
n4dqzjxg5td-szxxvnycpaq-0
2019-10-25 06:02:50,551 root INFO Checking for worker node: oke-cqtamjqgjs-g-
n4dqzjxg5td-szxxvnycpaq-2
2019-10-25 06:02:50,551 root INFO Nodes are not in ACTIVE state, will retry in 60
seconds.
2019-10-25 06:03:51,870 root INFO Checking for worker node: oke-cqtamjqgjs-g-
n4dqzjxg5td-szxxvnycpaq-2
2019-10-25 06:03:51,870 root INFO Worker node is in ACTIVE state now, proceeding
further.
2019-10-25 06:03:51,871 root INFO Kubeconfig file not readable or missing at
location: /root/.kube/config
2019-10-25 06:03:51,871 root INFO Creating kubeconfig at default location in
user's home.
2019-10-25 06:03:53,339 root INFO New config written to the Kubeconfig file
/root/.kube/config
```

```
2019-10-25 06:03:54,842 root INFO Kubernetes master is running at
https://cqtamjqgjs-g.ap-tokyo-1.clusters.oci.oraclecloud.com:6443
KubeDNS is running at https://cqtamjqgjs-g.ap-tokyo-
1.clusters.oci.oraclecloud.com:6443/api/v1/namespaces/kube-system/services/kube-
dns:dns/proxy
```

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

```
2019-10-25 06:03:54,843 root INFO Kubectl configured properly.
2019-10-25 06:03:58,727 root INFO Updated the deployment file with user provided
input.
2019-10-25 06:03:58,728 root INFO Using baseimage:
iad.ocir.io/psmsvc3/accs/java8:latest
2019-10-25 06:03:58,829 root INFO Pulling base image :
iad.ocir.io/psmsvc3/accs/java8:latest
2019-10-25 06:04:30,145 root INFO Building application image. This can take
several minutes...
2019-10-25 06:05:04,228 root INFO Image tag is:
NRT.ocir.io/sehubjapacprod/accs/oracleidentitycloudservice/shengjun.zhu/notice4:la
test
2019-10-25 06:05:04,309 root INFO Pushing application image to OCI registry. This
can take several minutes...
2019-10-25 06:05:43,770 root INFO Application image successfully pushed to OCI
registry.
2019-10-25 06:05:52,017 root INFO Config Map created successfully
2019-10-25 06:05:52,017 root INFO Creating docker-registry secret for pulling
images from ocir.
2019-10-25 06:05:53,382 root INFO Docker-registry secret for pulling images from
ocir has been created successfully.
```

```
2019-10-25 06:05:53,382 root INFO Creating deployment for application. This can
take several minutes...
Waiting for deployment "notice4-deployment" rollout to finish: 0 of 2 updated
replicas are available...
Waiting for deployment "notice4-deployment" rollout to finish: 1 of 2 updated
replicas are available...
deployment "notice4-deployment" successfully rolled out
2019-10-25 06:06:42,141 root INFO Deployment created successfully.
2019-10-25 06:06:42,142 root INFO Creating service for application. This can take
several minutes...
2019-10-25 06:06:43,626 root INFO Kubernetes service created successfully.
2019-10-25 06:06:45,075 root INFO Checking if load balancer is accessible. This
can take several minutes...
2019-10-25 06:06:46,463 root INFO Load balancer is not accessible. Trying again in
seconds: 30
2019-10-25 06:07:17,799 root INFO Load balancer is accessible. Public IP address
of load balancer is: 140.238.57.90
2019-10-25 06:07:20,440 root INFO Zone already exists: taosheng.tk
2019-10-25 06:07:22,702 root INFO Adding DNS record: notice4.taosheng.tk
2019-10-25 06:07:32,917 root INFO DNS record added.
2019-10-25 06:07:32,918 root INFO Application URL is: http://notice4.taosheng.tk
2019-10-25 06:07:32,919 root INFO Application creation completed.
```

この例では、クラスタ及び関連のすべてのリソースを作成し、Javaアプリケーションを作成し、カスタムURLおよび新しいDNSゾーンを使用します。

アプリケーションの削除

作成中に問題が発生した場合は、Oracle Cloud Infrastructure Container Engine for Kubernetesクラスタからアプリケーションを削除して再起動できます。構成ファイルを読み取り、DNSレコード、Kubernetesクラスタ内のアプリケーション・リソースおよびアプリケーション・ディレクトリ(migration_tool/work/<application-name>)を削除するアプリケーション削除アクションを使用できます。

1. アプリケーションを削除するには、コマンドライン・ウィンドウを開き、移行スクリプトを実行します。構成ファイルへのパスを指定します。

```
python app.py -a delete -f <path-of-config-file>
```

2. コマンドライン・ウィンドウで、アプリケーション・ディレクトリを削除するYESを入力します。

注意: 削除オプションでは、クラスタ、Vcn、サブネット、DNSゾーンなどのOracle Cloud Infrastructureリソースは削除されません。リソースを削除する必要がある場合は、Oracle Cloud InfrastructureコンソールまたはREST APIを使用して、リソースを手動で削除する必要があります。

例:

```
[root@accs2oke accs-migration-1.0.0]# python app.py -a delete -f ./config.json
2019-10-25 07:04:29,113 root INFO Starting application deletion.
2019-10-25 07:04:29,114 root INFO Verifying input config file...
2019-10-25 07:04:29,114 root INFO URL setup is enabled. Checking if zone name and
compartment OCID is given.
2019-10-25 07:04:29,114 root INFO Fetching OCI details....
2019-10-25 07:04:33,142 root INFO Checking if any Kubernetes resources exist for
the application...
Error from server (NotFound): secrets "notice4-tls-certificate" not found
2019-10-25 07:04:40,218 root INFO Some Kubernetes resources found for the
application. Cleaning up...
2019-10-25 07:04:45,659 root INFO Kubernetes resources cleaned up successfully.
2019-10-25 07:04:45,659 root INFO Checking if application data exists in local
directory...
The application data exists at location /home/opc/Downloads/accs-migration-
1.0.0/work/notice4 . Do you want to remove it ? Yes/No: Yes
2019-10-25 07:05:48,396 root INFO Successfully removed application data from local
directory.
2019-10-25 07:05:48,398 root INFO Checking if DNS zone record exists...
2019-10-25 07:05:52,395 root INFO DNS zone record exists. Removing it...
2019-10-25 07:05:55,563 root INFO DNS zone record removed successfully.
2019-10-25 07:05:55,564 root INFO Application deletion completed. Check the logs
for further details.
```

スクリプト・ログの取得

スクリプト・ログの詳細を取得できます: `$SCRIPT_ROOT_FOLDER/work///logs/accs-migration.log`.

説明:

1. `$SCRIPT_ROOT_FOLDER`は移行スクリプト・ツール(`app.py`)のディレクトリです。
2. `<app-name>`は、構成ファイルで指定したアプリケーション名です。 例:

```
[root@accs2oke accs-migration-1.0.0]# tail -10 /home/opc/Downloads/accs-migration-
1.0.0/work/notice4/create/logs/accs-migration.log
2019-10-25 06:06:45,074 root DEBUG Service type is: LoadBalancer
2019-10-25 06:06:45,075 root INFO Checking if load balancer is accessible. This
can take several minutes...
2019-10-25 06:06:46,463 root INFO Load balancer is not accessible. Trying again in
seconds: 30
2019-10-25 06:07:17,799 root INFO Load balancer is accessible. Public IP address
of load balancer is: 140.238.57.90
2019-10-25 06:07:20,440 root INFO Zone already exists: taosheng.tk
2019-10-25 06:07:20,441 root DEBUG DNS record: notice4.taosheng.tk
2019-10-25 06:07:22,702 root INFO Adding DNS record: notice4.taosheng.tk
2019-10-25 06:07:32,917 root INFO DNS record added.
2019-10-25 06:07:32,918 root INFO Application URL is: http://notice4.taosheng.tk
2019-10-25 06:07:32,919 root INFO Application creation completed.
```

アプリケーションの手動移行

Oracle Application Container Cloud Serviceアプリケーションに基づいてDockerイメージを作成し、Oracle Cloud Infrastructure Container Engine for Kubernetesにデプロイします。

トピックス:

- Kubernetesクラスタを作成
- Kubectlの構成
- Dockerイメージのビルド
- Linuxパッケージを追加インストール
- Docker ImageをOracle Cloud Infrastructure Registryにプッシュ
- 環境変数の設定
- Java EE Systemおよびサービス・バインディング・プロパティの構成
- Kubernetes ClusterとOracle Cloud Services間の接続の有効化
- Kubernetes構成ファイルの作成
- Docker Registry SecretおよびSSL証明書を設定
- アプリケーションのデプロイ
- カスタムURLの設定
- イングレス・コントローラを設置

Kubernetesクラスタを作成

Oracle Application Container Cloud ServiceアプリケーションをOracle Cloud Infrastructureに移行するには、Kubernetesクラスタを作成するか、既存のクラスタを使用できます。

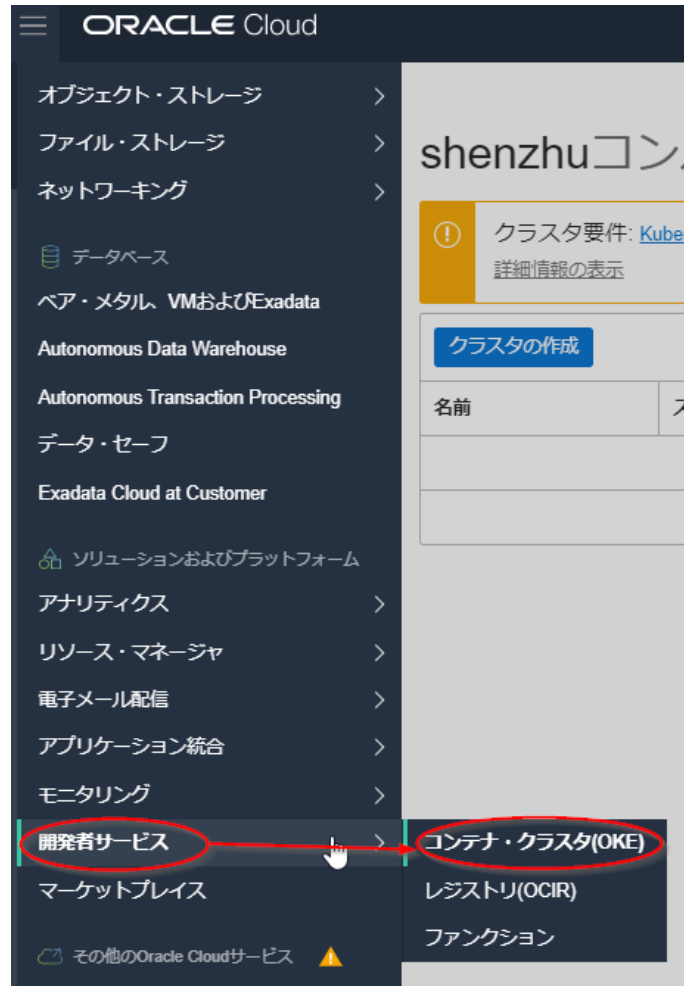
既存のKubernetesクラスタには、次のような特性が必要です:

- クラスタが使用するVirtual Cloud Network (VCN)には、リモート・アクセスを可能にするためにインターネット・ゲートウェイが含まれている必要があります。
- クラスタ内のロード・バランシング(LB)サブネットには、インターネットからポート80または443 (SSLが有効な場合)上のサブネットに受信トラフィックを許可するイングレス・セキュリティ・ルールが必要です。
- クラスタ内のノードのシェイプと数量は、このクラスタに移行するすべてのアプリケーションのCPUとメモリーの容量要件を満たすほど十分に大きくする必要があります。

Oracle Cloud Infrastructureコンソールを使用して、新しいKubernetesクラスタを作成できます。 マスター・ノードにインストールするクラスタ名やKubernetesバージョンなどの詳細を指定する必要があります。

OKEクラスターのプロビジョニングもご参照ください。

1. Oracle Cloud Infrastructureコンソールから、ナビゲーション・メニューを開きます。「ソリューション、プラットフォームおよびエッジ」の下で「開発者サービス、」に移動し、「コンテナ・クラスタ (OKE)」をクリック



2. 「クラスタ」ページでコンパートメントを選択し、「クラスタの作成」をクリック



3. クラスタの作成ダイアログで、次の値を入力および選択します:

- 名前:デフォルト名をそのまま使用するか、任意の名前を入力します。
- KUBERNETESバージョン:マスター・ノードおよびクラスタのワーカー・ノードで実行する Kubernetesのバージョンを選択します。
- クイック作成: 選択済
- シェイプ:ノード・プールの各ノードに使用するシェイプを選択します。 このシェイプによって、CPUの数と各ノードに割り当てられるメモリー量が決まります。

- サブネットごとの数量:各サブネット内のノード・プールに作成するワーカー・ノードの数を入力します。
- 公開SSHキー: (オプション)ノード・プールの各ノードへのSSHアクセスに使用するキー・ペアの公開キーを入力します。公開キーは、クラスタ内のすべてのワーカー・ノードにインストールされます。公開SSHキーを指定しない場合は、Oracle Cloud Infrastructure Container Engine for Kubernetesによって提供されます。ただし、対応する秘密キーがないため、ワーカー・ノードへのSSHアクセス権を持つことはできません。

4. 作成をクリックします。

ヘルプ

閉じる

クラスタの作成

クラスタ・コンパートメント

shenzhu

名前

accs_migration_poc_cluster

KUBERNETESバージョン

v1.13.5

マスター・ノードとワーカー・ノードにインストールされたKubernetesバージョン

クイック作成

デフォルト設定を使用してクラスタを迅速に作成し、専用ネットワークも作成します

カスタム作成

カスタム設定を使用してクラスタを作成し、既存のネットワークを引き継ぎます

仮想クラウド・ネットワークの作成

機能するクラスタを確保するために、新しいVCNネットワークが作成されます

コンパートメント: shenzhu リソース作成: 1つのVCN、1つのサービスLBサブネットおよび1つのワーカー・ノード・サブネット

プライベート

作成されたKubernetesワーカー・ノードは、プライベート・サブネットでホストされます

パブリック

作成されたKubernetesワーカー・ノードは、パブリック・サブネットでホストされます

ノード・プールの作成

名前: pool1 コンパートメント: shenzhu バージョン: v1.13.5 イメージ: Oracle-Linux-7.6

シェイプ

VM.Standard2.1

ノードの数

3

ノード・プールのすべてのノードのシェイプ。

ノード・プールのノードの数。

SSH公開キー オプション

ssh-rsa
AAAAAB3NzaC1yc2EAAAABJQAAAAQEAOTXNBaS2YoZGvU0Tan3pVYPXNAEzrEHMyo1ZQsHdmuRT1btFUoul8Q0kRSMt6elssgYFmy2ylKtD29EWHd0K
hbglFk5arSVE4xoYEK5FCBDCsXfuYsXbuZte3dvZjrfNp4YdXNlmsEp8oj4tYAF9q36VaqA9rRTJ0HbO9suVDem1t5kyWhk8eqhsf2mDHZNJneCmptpe9Vglzdy7
FjraRlyNxd7wVjkXLgOwFyQ9wjne4dlbICg0PbE4PQdm4lugXaTaNXiVPs6NdBNPXXm8i+NJKNNHPnnvWNhUvzfQ5TqGYa1g1N4EEba5XEEQqlZ+cbhB9i
Q/Ux03MJHgQ== rsa-key-20181121

SSH公開キーでプライベート・ノードにアクセスするには、秘密キースト(別名ジャンプ・ボックス)を設定する必要があります。 [秘密キーストの設定の詳細](#)を参照してください

Kubernetesラベル

キー

name

値

pool1

このノード・プールに追加されたノードには、1つ以上のKubernetesラベルが自動的に適用されるため、ユーザーは、特定のプールのKubernetesワークロードをターゲットにできません

+ 別のペア

追加アドオン

KUBERNETESダッシュボード有効

TILLER (HELM)有効

このクラスタのリクエスト後に詳細ページを表示

作成

取消

クラスタの作成には数分かかることがあります。クラスタが作成されたあと、そのステータスはアクティブに変わります。

クラスタと関連ネットワーク・リソースの作成

仮想クラウド・ネットワークの作成

仮想クラウド・ネットワークが作成されました: [oke-vcn-quick-accs_migration_poc_cluster-20190926021143](#)

インターネット・ゲートウェイの作成

インターネット・ゲートウェイが作成されました: [oke-igw-quick-accs_migration_poc_cluster-20190926021143](#)

作成 ルート表

ルート表が作成されました: [oke-routetable-accs_migration_poc_cluster-20190926021143](#)

セキュリティ・リストの作成

セキュリティ・リストが作成されました: [oke-svclbseclist-quick-accs_migration_poc_cluster-20190926021143](#)
セキュリティ・リストが作成されました: [oke-seclist-quick-accs_migration_poc_cluster-20190926021143](#)

サブネットの作成

サブネットが作成されました: [oke-svclbsubnet-quick-accs_migration_poc_cluster-20190926021143-regional](#)
サブネットが作成されました: [oke-subnet-quick-accs_migration_poc_cluster-20190926021143-regional](#)

クラスタの作成

クラスタのリクエスト中: [accs_migration_poc_cluster](#)

ノード・プールの作成

ノード・プールのリクエスト中: [pool1](#)

✔ クラスタと関連ネットワーク・リソースが作成されました。

Close

Kubectlの構成

kubectlを使用してクラスタにアクセスできるように、kubeconfigファイルをダウンロードする必要があります。

CLIのセットアップもご参照ください。

- 作成したクラスタのページで「Kubeconfigへのアクセス」をクリックします。

ORACLE Cloud

Japan East (Tokyo)

コンテナ > クラスタ > accs_migration_poc_cluster

accs_migration_poc_cluster

Kubeconfigへのアクセス クラスタの監視

クラスタ詳細

クラスタ情報

クラスタ・ステータス: アクティブ

ノード・プール: 1

クラスタID: [cypgysgft](#) 表示 コピー

コンパートメント: [sehuajpacprod \(ルート\)JAPAC_PaaSOracleDemo/shenzhu](#)

ネットワーク情報

VCN名: [oke-vcn-quick-accs_migration_poc_cluster-20190926021143](#)

VCN ID: [m5kac3a](#) 表示 コピー

コンパートメント: [sehuajpacprod \(ルート\)JAPAC_PaaSOracleDemo/shenzhu](#)

ポッドCIDR: 10.244.0.0/16

サービスCIDR: 10.96.0.0/16

サービスLBサブネット1: [021143-regional](#) 表示 コピー

サービスLBサブネット2: -

Node Pools

ノード・プールの作成

pool1

詳細 ラベル

ノード・プールID: [imdmrpf](#) 表示 コピー

Kubernetesバージョン: v1.13.5

シェイプ: VM.Standard2.1

合計ワーカー・ノード: 3

イメージ: Oracle-Linux-7.6

コンパートメント: [sehuajpacprod \(ルート\)JAPAC_PaaSOracleDemo/shenzhu](#)

アクション

- 「Kubeconfigへのアクセス方法」ダイアログ・ボックスに表示された手順に従います。

Kubeconfigへのアクセス方法 [ヘルプ](#) [閉じる](#)

OCI CLIバージョン2.6.4 (以上)を[ダウンロードしてインストール](#)し、使用するために[構成](#)しておく必要があります。OCI CLIのバージョンが2.6.4より前の場合は、[こちら](#)から新しいバージョンをダウンロードしてインストールしてください。

現在インストールされているOCI CLIのバージョンが不明な場合は、次のコマンドでチェックします。

```
oci -v
```

クラスターのkubeconfigにアクセスするには、次のコマンドを実行します:

```
1. mkdir -p $HOME/.kube
2. oci ce cluster create-kubeconfig --cluster-id ocid1.cluster.oc1.ap-tokyo-1.aaaaaaaafrwkmrug4zdqnddgjstmbogntsnstdgi4wgjygygyeysrg5rt --file $HOME/.kube/config --region ap-tokyo-1 --token-version 2.0.0
```

KUBECONFIG環境変数をこのクラスターのファイルに設定するには、次を使用します:

```
export KUBECONFIG=$HOME/.kube/config
```

新しいkubeconfigを別の場所に保存する場合、前述のCLIコマンドの--file引数を新しい場所のパスに変更してください。場合によっては、KUBECONFIG環境変数もこの新しい場所のパスを使用して設定または更新する必要があります。環境変数を永続化するには、シェル起動スクリプトも更新する必要があります。

kubeconfigファイルの管理の詳細は、公式の[Kubernetesドキュメント](#)を参照してください。Kubernetes CLIのOCIのコンテナ・エンジンで使用できるコマンドの詳細が[ここ](#)に記載されています。

[閉じる](#)

3. kubectlを使用してクラスターにアクセスできることを確認します。

```
$ kubectl get services
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP
PORT(S) AGE			
kubernetes	ClusterIP	10.96.0.1	<none>
443/TCP	12d		

Dockerイメージのビルド

Dockerイメージをビルドするには、Dockerfileにそのイメージを組み立てる命令が含まれている必要があります。アプリケーション・ランタイムに基づいて、テンプレートを使用してDockerfileを作成できます。

コンテナイメージの作成もご参照ください。

2. アプリケーションのランタイム・タイプを指定してください。

注意: Java、Node、PHP、Java EE、DotNet、Ruby、PythonまたはGoのいずれかにしてください。

3. manifest.json ファイルを開き、ランタイム・バージョンを特定します。

例:

```
{
  "runtime": {
    "majorVersion": "8"
  },
  ...
}
```

例のランタイム・バージョンは8です

4. プロジェクト・ディレクトリで、**Dockerfile**を作成し、実行時にそのテンプレートを使用します:

Java:

```
FROM iad.ocir.io/psmsvc3/accs/java<runtime-version>:latest
ENV APP_HOME=/u01/app
WORKDIR /u01/app
EXPOSE 8080
USER root
COPY . /u01/app
RUN if [ -f /u01/app/linux-packages.txt ] ; then  chmod 755
/u01/app/package-installer.sh && /u01/app/package-installer.sh ; fi
RUN mkdir -p /u01/scripts /u01/logs/ \
&& chown -R apaas:apaas /u01/
USER apaas
```

Node:

```
FROM iad.ocir.io/psmsvc3/accs/node<runtime-version>:latest
ENV APP_HOME=/u01/app
WORKDIR /u01/app
EXPOSE 8080
USER root
COPY . /u01/app
RUN if [ -f /u01/app/linux-packages.txt ] ; then  chmod 755
/u01/app/package-installer.sh && /u01/app/package-installer.sh ; fi
RUN mkdir -p /u01/scripts /u01/logs/ \
&& chown -R apaas:apaas /u01/
USER apaas
```

Java EE:

```
FROM iad.ocir.io/psmsvc3/accs/javaee<runtime-version>:latest
ENV APP_HOME=/u01/app
WORKDIR /u01/app
EXPOSE 8080
USER root
COPY . /u01/app
RUN if [ -f /u01/app/linux-packages.txt ] ; then  chmod 755
/u01/app/package-installer.sh && /u01/app/package-installer.sh ; fi
RUN mkdir -p /u01/logs/ \
&& chown -R apaas:apaas /u01/
USER apaas
ENTRYPOINT ["sh", "-c", "$SCRIPT_HOME/post-install.sh &&
$DOMAIN_HOME/startServer.sh"]
```

PHP:

```
FROM iad.ocir.io/psmsvc3/accs/php<runtime-version>:latest
ENV APP_HOME=/u01/app
WORKDIR /u01/app
EXPOSE 8080
USER root
COPY . /u01/app
RUN if [ -f /u01/app/linux-packages.txt ] ; then  chmod 755
/u01/app/package-installer.sh && /u01/app/package-installer.sh ; fi
RUN mkdir -p /u01/scripts /u01/logs/ \
&& chown -R apaas:apaas /u01/
USER apaas
ENTRYPOINT ["sh", "-c", "apache2-run"]
```

DotNet:

```
FROM microsoft/dotnet:<runtime-version>-runtime
ENV APP_HOME=/u01/app
WORKDIR /u01/app
EXPOSE 8080
RUN mkdir -p /u01/app/
COPY . /u01/app
RUN mkdir -p /u01/data/ \
&& mkdir -p /u01/logs/ \
&& groupadd apaas \
&& groupadd builds \
&& useradd -m -b /home -g apaas -G builds apaas \
&& chown -R apaas:apaas /u01/ \
&& chgrp -hR builds /usr/local
USER apaas
```

Ruby:

```
FROM ruby:<runtime-version>
ENV APP_HOME=/u01/app
WORKDIR /u01/app
EXPOSE 8080
RUN mkdir -p /u01/app/
COPY . /u01/app
RUN mkdir -p /u01/data/ \
&& mkdir -p /u01/logs/ \
&& groupadd apaas \
&& groupadd builds \
&& useradd -m -b /home -g apaas -G builds apaas \
&& chown -R apaas:apaas /u01/ \
&& chgrp -hR builds /usr/local
USER apaas
```

Python:

```
FROM python:<runtime-version>
ENV APP_HOME=/u01/app
WORKDIR /u01/app
EXPOSE 8080
RUN mkdir -p /u01/app/
COPY . /u01/app
RUN mkdir -p /u01/data/ \
&& mkdir -p /u01/logs/ \
&& groupadd apaas \
&& groupadd builds \
&& useradd -m -b /home -g apaas -G builds apaas \
&& chown -R apaas:apaas /u01/ \
&& chgrp -hR builds /usr/local
USER apaas
```

Go:

```
FROM golang:<runtime-version>
ENV APP_HOME=/u01/app
WORKDIR /u01/app
EXPOSE 8080
RUN mkdir -p /u01/app/
COPY . /u01/app
RUN mkdir -p /u01/data/ \
&& mkdir -p /u01/logs/ \
&& groupadd apaas \
&& groupadd builds \
&& useradd -m -b /home -g apaas -G builds apaas \
&& chown -R apaas:apaas /u01/ \
&& chgrp -hR builds /usr/local
USER apaas
```

5. ファイル内の<runtime-version>プレースホルダーを、アプリケーションの実行時バージョンで置き換えます。

例:

```
FROM iad.ocir.io/psmsvc3/accs/java8:latest
ENV APP_HOME=/u01/app
WORKDIR /u01/app
EXPOSE 8080
USER root
COPY . /u01/app
RUN if [ -f /u01/app/linux-packages.txt ] ; then chmod 755
/u01/app/package-installer.sh && /u01/app/package-installer.sh ; fi
RUN mkdir -p /u01/scripts /u01/logs/ \
```

```
&& chown -R apaas:apaas /u01/  
USER apaas
```

6. アプリケーションで追加のLinuxパッケージが必要な場合は、`package-installer.sh`スクリプトを`linux-packages.txt`ファイルとともに含めます。「Linuxパッケージを追加インストール」を参照してください。
7. コマンド行ウィンドウを開き、プロジェクト・ディレクトリに移動して、Dockerイメージをビルドします。`<local-image-name>`プレースホルダーをイメージの名前に置き換えます。

```
sudo docker build -t <local-image-name>:latest
```

注意: ローカル・イメージの複数のバージョンを保守する場合、最新ではなく適切なタグを使用できます。

例:

```
[root@accs-migration-poc-vm accs2oke]# cd $HOME/accs2oke  
[root@accs-migration-poc-vm accs2oke]# sudo docker build -t notice-  
image:latest .  
Sending build context to Docker daemon 21.14MB  
Step 1/9 : FROM iad.ocir.io/psmsvc3/accs/java8:latest  
---> 0f6c786aee03  
Step 2/9 : ENV APP_HOME=/u01/app  
---> Running in 334f063a1e7e  
Removing intermediate container 334f063a1e7e  
---> 4d533d052bf6  
Step 3/9 : WORKDIR /u01/app  
---> Running in 88ca8685858c  
Removing intermediate container 88ca8685858c  
---> 1214dde2dd67  
Step 4/9 : EXPOSE 8080  
---> Running in cdde3ff367f6  
Removing intermediate container cdde3ff367f6  
---> 8068d29e3ed8  
Step 5/9 : USER root  
---> Running in d3c502a64cb2  
Removing intermediate container d3c502a64cb2  
---> e36609976271  
Step 6/9 : COPY . /u01/app  
---> 98117051006b  
Step 7/9 : RUN if [ -f /u01/app/linux-packages.txt ] ; then chmod 755  
/u01/app/package-installer.sh && /u01/app/package-installer.sh ; fi  
---> Running in ffd1fa7799fb  
Removing intermediate container ffd1fa7799fb  
---> 96b39851c009  
Step 8/9 : RUN mkdir -p /u01/scripts /u01/logs/ && chown -R apaas:apaas  
/u01/  
---> Running in ace9fd561179
```

```
Removing intermediate container ace9fd561179
---> c424be2fc2a5
Step 9/9 : USER apaas
---> Running in c63bc8a23954
Removing intermediate container c63bc8a23954
---> 1aaac123baab
Successfully built 1aaac123baab
Successfully tagged notice-image:latest
```

8. Dockerイメージが作成されたことを確認します。

```
sudo docker image ls | grep <local-image-name>
```

例:

```
[root@accs-migration-poc-vm accs2oke]# sudo docker image ls | grep notice-
image
notice-image                                latest                                1aaac123baab
About a minute ago    776MB
```

9. アプリケーションDockerイメージのコンテンツを検証します。

```
sudo docker run -it <local-image-name>:latest sh
```

例:

```
[root@accs-migration-poc-vm accs2oke]# sudo docker run -it notice-
image:latest sh
sh-4.2$ ls
Dockerfile  Notice-0.0.1-SNAPSHOT.jar  manifest.json
sh-4.2$
```

10. アプリケーション・コンテナを終了します。

```
sh-4.2$ exit
```

Linuxパッケージを追加インストール

linux-packages.txtファイルを作成し、アプリケーションとともにバンドルすることで、追加のLinuxパッケージをインストールできます。

プロジェクト・ディレクトリで、package-installer.shファイルを作成し、次のスクリプトの内容を貼り付けます:

```
#Step 1: Check if the file linux-packages.txt exists at /u01/app
#Step 2: Iterate through the file linux-packages.txt, to verify package exists in
our Oracle repo
#If step 2 success, then go ahead and install all packages. If step 2 fails, then
exit with failure and display specify packages

# The timeout function is called with 20m hard timeout, and within the stipulated
time if all the package are installed
# then return from function will be caught in parent process.
# An appropriate message displays back, depending on the return status.
# Return code for below scenarios:
# Syntax error: 2
# Validation failure : 3
# Success : 4
# Transaction error: 5

#!/bin/bash

timeout_value=20

cust_loc=/u01/app
export cust_loc
cd $cust_loc
if [ ! -s $cust_loc/linux-packages.txt ] || [ ! -f $cust_loc/linux-packages.txt ]
then
    exit 0
fi

export uuid=`date +%Y%m%d%H%M%S`
export LOG_NAME="/tmp/output_${uuid}.log"
export PKG_LOGNAME="/tmp/pkgoutput_${uuid}.log"
export GRP_LOGNAME="/tmp/grpoutput_${uuid}.log"
export ERR_LOG_NAME="/tmp/error_${uuid}.log"
rm -rf $LOG_NAME

function cleanup()
{
/bin/rm -rf $PKG_LOGNAME
/bin/rm -rf $GRP_LOGNAME
/bin/rm -rf /tmp/tmp_log /tmp/tmp_syn /tmp/tmp_succ /tmp/tmp_val /tmp/tmp_err
}

function install_packages()
{
cur_loc=`pwd`
ret_flag=0
syn_flag=0
sucpack_list=""
```

```

sucgroup_list=""
synpack_list=""
errpack_list=""
sucdisp_list=""
grpdisp_list=""
install_pkgs=""
failed_pkgs=""

#Fix for file created in notepad++
sed -i -e '$a\' $cust_loc/linux-packages.txt

echo "VALIDATION_CHECK_START" > $LOG_NAME
while read package_rec
do
    if [[ "$package_rec" != "#"* ]] && [[ ! -z "$package_rec" ]]
    then
        echo "Record Picked: $package_rec" >> $LOG_NAME
        package_name=`echo $package_rec | awk -F':' '{print $2}' | sed -e
's/^[:space:]*//' -e 's/[:space:]*$//'`
        install_type=`echo $package_rec | awk -F':' '{print $1}' | tr -d
'[:space:]`
        if [ "package_install" == "$install_type" ];then
            yum list $package_name 1>>$LOG_NAME 2>&1
            var=$?
            if [ $var -eq 1 ]
            then
                if [[ -z "$errpack_list" ]];then
                    errpack_list=$package_name
                else
                    errpack_list=$errpack_list,"$package_name
                fi
                ret_flag=1
            elif [ $var -eq 0 ]
            then
                if [[ -z "$sucdisp_list" ]];then
                    sucdisp_list=$package_name
                else
                    sucdisp_list=$sucdisp_list,"$package_name
                fi
                sucpack_list=$sucpack_list" "$package_name
            fi
        elif [ "group_install" == "$install_type" ];then
            yum grouplist "$package_name" 1>>$LOG_NAME 2>&1
1>/tmp/tmp_log
            cat /tmp/tmp_log | grep -E -iw -q "Available
Groups|Installed Groups:"
            var=$?
            if [ $var -eq 1 ]
            then
                if [[ -z "$errpack_list" ]];then
                    errpack_list=$package_name
                else
                    errpack_list=$errpack_list,"$package_name
                fi

```

```

                ret_flag=1
            elif [ $var -eq 0 ]
            then
                if [[ -z "$grpdisp_list" ]];then
                    grpdisp_list=$package_name
                else
                    grpdisp_list=$grpdisp_list,"$package_name"
                fi
                if [[ -z "$sucgroup_list" ]];then
                    sucgroup_list=$package_name
                else
                    sucgroup_list=$sucgroup_list,"$package_name"
                fi
            fi
        else
            #Syntax failure scenario if not properly provided
            if [[ -z "$synpack_list" ]];then
                synpack_list=$package_rec
            else
                synpack_list=$synpack_list,"$package_rec"
            fi
            ret_flag=-1
            syn_flag=1
        fi
    fi
done < $cust_loc/linux-packages.txt
echo "VALIDATION_CHECK_END" >> $LOG_NAME
if [ $syn_flag -eq 1 ]
then
    echo "Syntax Error: $synpack_list" > /tmp/tmp_syn
    return 2
fi

if [ $ret_flag -eq 1 ]
then
    echo "Valid Packages: $sucdisp_list,$grpdisp_list" > /tmp/tmp_val
    echo "Invalid Packages: $errpack_list" >> /tmp/tmp_val
    return 3
fi
if [ $ret_flag -eq 0 ]
then
    echo "INSTALL_START" >> $LOG_NAME
    if [ ! -z "$sucpack_list" ];then
        yum -y install $sucpack_list 1>>$PKG_LOGNAME 2>&1
        resp=$?
        if [ $resp -eq 1 ];then
            /bin/rm -rf $PKG_LOGNAME
            ret_flag=2
            for pkg_name in $sucpack_list
            do
                yum -y install $pkg_name 1>>$PKG_LOGNAME
            done
            res=$?
        fi
    fi
fi

```



```

        if [ $res -eq 1 ];then
            if [ -z "$failed_pkgs" ];then
                failed_pkgs=$pkg_name
            else
                failed_pkgs=$failed_pkgs,"$pkg_name"
            fi
            echo "Package Name: $pkg_name" >> $ERR_LOG_NAME
            cat /tmp/tmp_log >> $ERR_LOG_NAME
            cat /tmp/tmp_log >> $PKG_LOGNAME
        elif [ $res -eq 0 ];then
            if [ -z "$install_pkgs" ];then
                install_pkgs=$pkg_name
            else
                install_pkgs=$install_pkgs,"$pkg_name"
            fi
        fi
    done
    cat $PKG_LOGNAME >> $LOG_NAME
fi
if [ $resp -eq 0 ]
then
    cat $PKG_LOGNAME >> $LOG_NAME
    install_pkgs=$sucdisp_list
fi
fi
if [ ! -z "$sucgroup_list" ];then
    yum -y groupinstall "$sucgroup_list" 1>>$GRP_LOGNAME 2>&1
    resp=$?
    if [ $resp -eq 1 ];then
        ret_flag=2
        /bin/rm -rf $GRP_LOGNAME
        IFS=","
        for grp_name in $sucgroup_list
        do
            yum -y groupinstall "$grp_name" 1>>$GRP_LOGNAME
            ret_res=$?
            if [ $ret_res -eq 1 ];then
                if [ -z "$failed_pkgs" ];then
                    failed_pkgs=$grp_name
                else
                    failed_pkgs=$failed_pkgs,"$grp_name"
                fi
                echo "Group Name: $grp_name" >>
                $ERR_LOG_NAME
                cat /tmp/tmp_log >> $ERR_LOG_NAME
                cat /tmp/tmp_log >> $GRP_LOGNAME
            elif [ $ret_res -eq 0 ];then
                if [ -z "$install_pkgs" ];then
                    install_pkgs=$grp_name
                else

```

```

install_pkgs=$install_pkgs", "$grp_name
                                fi
                                fi
                                done
                                cat $GRP_LOGNAME >> $LOG_NAME
                                fi
                                if [ $resp -eq 0 ]
                                then
                                    cat $GRP_LOGNAME >> $LOG_NAME
                                    if [ -z "$install_pkgs" ];then
                                        install_pkgs=$sucgroup_list
                                    else
                                        install_pkgs=$install_pkgs", "$sucgroup_list
                                    fi
                                fi
                                fi
                                echo "INSTALL_END" >> $LOG_NAME
                                fi
                                if [ -z "$failed_pkgs" ];then
                                    ret_flag=0
                                fi
                                if [ $ret_flag -eq 0 ];then
                                    echo "Installed Packages: $sucdisp_list,$grpdisp_list" > /tmp/tmp_succ
                                    return 4
                                fi
                                if [ $ret_flag -eq 2 ];then
                                    echo "Installable Packages: $install_pkgs" > /tmp/tmp_err
                                    echo "Failed Packages: $failed_pkgs" >> /tmp/tmp_err
                                    return 5
                                fi
                            }
#End of install_package function

export -f install_packages
timeout "$timeout_value"m bash -c install_packages
rest_status=$?

# Timeout scenario
if [ $rest_status -eq 124 ]
then
    echo "RESULT_START"
    echo "SYNTAX_ERROR"
    echo "Error Message : Timed out while installing & configuring linux
packages/groups. Reduce the number of specified linux packages/groups."
    echo "RESULT_END"
    cleanup
    exit 1
fi

# Syntax error scenario
if [ $rest_status -eq 2 ]
then

```

```
        echo "RESULT_START"
        echo "SYNTAX_ERROR"
    cat /tmp/tmp_syn
        echo "RESULT_END"
        cleanup
    exit 1
fi

#Validation error scenario
if [ $rest_status -eq 3 ]
then
    echo "RESULT_START"
    echo "VALIDATION_FAILURE"
    cat /tmp/tmp_val
    echo "RESULT_END"
    cat $LOG_NAME
    cleanup
    exit 1
fi

#Success scenario
if [ $rest_status -eq 4 ]
then
    echo "RESULT_START"
    echo "SUCCESS"
    cat /tmp/tmp_succ
    echo "RESULT_END"
    cleanup
    cat $LOG_NAME
    exit 0
fi

#Transaction error scenario
if [ $rest_status -eq 5 ]
then
    echo "RESULT_START"
    echo "ERROR_PACKAGE"
    cat /tmp/tmp_err
    echo "RESULT_END"
    echo "ERROR_PKGS_START"
    cat $ERR_LOG_NAME
    echo "ERROR_PKGS_END"
    cleanup
    cat $LOG_NAME
    exit 1
fi
```

DockerイメージをOracle Cloud Infrastructureレジストリにプッシュ

Dockerイメージをビルドした後、Oracle Cloud Infrastructureレジストリ(OCIR)にプッシュしてクラウドで使用できるようにすることができます。

OCIRへのプッシュとOKEへのデプロイもご参照ください。

1. コマンド行ウィンドウで、ローカルDockerイメージのタグを作成します。 次の書式を使用します。

```
image_tag="<region-code>.ocir.io/<tenancy>/accs/<リポジトリ名>/<app-name>:latest"
```

このコマンドにおいてリポジトリ名には任意の文字列を指定できますが、通常はプロジェクト名やユーザー名などを小文字にしたものを指定します。例えば、以下のようなコマンドになります。例:

```
[root@accs-migration-poc-vm accs2oke]#  
image_tag="nrt.ocir.io/sehubjapacprod/accs/shengjun.zhu/notice:latest"
```

2. タグをDockerイメージに割り当てます。

```
sudo docker tag <local-image-name>:latest $image_tag
```

例:

```
[root@accs-migration-poc-vm accs2oke]# sudo docker tag notice-image:latest  
$image_tag
```

3. Oracle Cloud Infrastructureレジストリにログインします。 Oracle Cloud Infrastructureテナンシ、ユーザー名および認証トークンを指定してください。

```
sudo docker login <region-code>.ocir.io --username=<tenancy_name>/<username>  
--password="<auth-token>"
```

例:

```
[root@accs-migration-poc-vm accs2oke]# sudo docker login nrt.ocir.io --  
username=sehubjapacprod/oracleidentitycloudservice/shengjun.zhu@oracle.com -  
-password="6h.}82DrE4+k#sfDa)3<"  
WARNING! Using --password via the CLI is insecure. Use --password-stdin.  
WARNING! Your password will be stored unencrypted in  
/root/.docker/config.json.  
Configure a credential helper to remove this warning. See  
https://docs.docker.com/engine/reference/commandline/login/#credentials-store  
  
Login Succeeded
```

4. Dockerイメージをレジストリにプッシュします。

```
sudo docker push $image_tag
```

例:

```
[root@accs-migration-poc-vm accs2oke]# sudo docker push $image_tag
The push refers to repository
[nrt.ocir.io/sehubjapacprod/accs/shengjun.zhu/notice]
162685c7f34c: Pushed
b999f9f09261: Pushed
091f5ab4d968: Pushed
baf7d2d1e99b: Pushed
9e35d0ce4d4b: Pushed
406954a29cb9: Pushed
922b485ea6a7: Pushed
latest: digest:
sha256:2653cca6375cc83a83dec436ae49efc382ff5ef9396cfe1f3b03b09a49d9b6a4
size: 1792
```

環境変数の設定

Oracle Application Container Cloud Serviceアプリケーションに必要な環境変数をKubernetes構成に移行します。

環境変数には3つのタイプがあります:

- Oracle Application Container Cloud Service環境変数: アプリケーションをデプロイすると、Oracle Application Container Cloud Serviceはこれらの変数を自動的に作成します。これらの環境変数は必須で、構成ファイルから削除することはできません。「環境変数の構成」を参照してください。
- カスタム環境変数: これらの変数は、`deployment.json`ファイルで、またはOracle Application Container Cloud Serviceコンソールを使用して、アプリケーションに対して定義したものです。
- サービス・バインディング環境変数: Oracle Application Container Cloud Serviceでは、アプリケーションでサービス・バインディングを追加すると、これらの変数が自動的に作成されます。

アプリケーションの環境変数を移行するには:

1. アプリケーションで1つ以上のサービス・バインディングが構成されている場合は、各サービス・バインディングの環境変数を検索します。Oracle Application Container Cloud Serviceコンソールから、アプリケーション・デプロイメントを選択し、「環境変数」セクションのサービス・バインディング変数を識別します。

ORACLE Cloud

Infrastructure Classic

SE

Notice

アプリケーション / Notice

URL: https://Notice-jptest01.apaas.ap5.oraclecloud.com

概要

2
インスタンス

4 GB
合計メモリー

デプロイメント

1
サービス・バインディング

0
ユーザー定義変数

管理

2
使用可能な更新

Oct 09, 2019 8:45:24 AM
UTC

2
ログ

0
記録

As of 10月 9, 2019 08:45:22 午前 UTC

更新

デプロイ済アプリケーション

現在のバージョン: 1.0

ソース: oracle

最終デプロイ日: Sep 25, 2019 6:20:09 AM UTC

コミットID: 1A2B345

アーカイブ・サイズ: 21.13 MB

ビルド番号: 24

アーカイブ名: Notice-0.0.1-SNAPSHOT-dist.zip

リリース・ノート: Notice Spring Boot Web Service

ランタイム

起動コマンド: java -jar Notice-0.0.1-SNAPSHOT.jar

Java SEのバージョン: 8

トポロジ

インスタンス: 2

メモリー(GB): 2

サービス・バインディング

Add

	型	名前	ユーザー名	アクション
	dbaas	DBCSDemo	oracleusr2	

環境変数

アプリケーションはこれらの環境変数にランタイムでアクセスできます。APP_HOMEなどの事前定義された変数は変更できません。別のクラウド・サービスの環境変数を変更するには、サービス・バインディングを編集します。カスタム環境変数を追加または更新できます。

☐ システム変数の非表示

Add

Name	Value	Actions
APP_HOME		
PORT		
DBAAS_DEFAULT_CONNECT_DESCRIPTOR	146.56.2.52:1521/PDB1.jptest01.oraclecloud.internal	

環境変数

アプリケーションはこれらの環境変数にランタイムでアクセスできます。APP_HOMEなどの事前定義された変数は変更できません。別のクラウド・サービスの環境変数を変更するには、サービス・バインディングを編集します。カスタム環境変数を追加または更新できます。

☐ システム変数の非表示

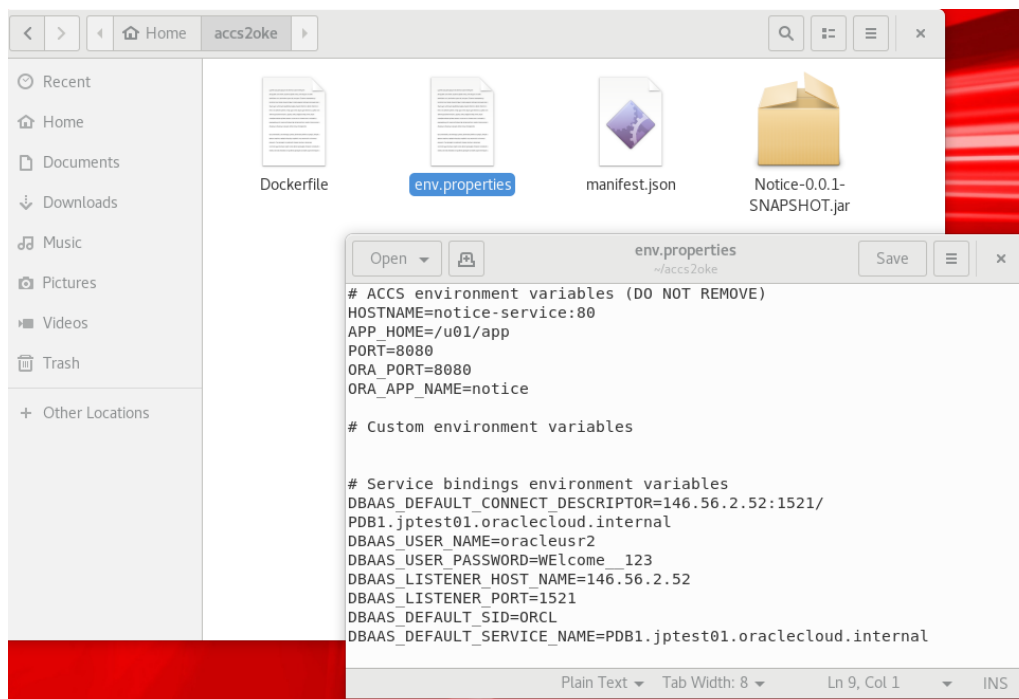
Add

Name	Value	Actions
APP_HOME		
PORT		
DBAAS_DEFAULT_CONNECT_DESCRIPTOR	146.56.2.52:1521/PDB1.jptest01.oraclecloud.internal	
DBAAS_USER_NAME	oracleusr2	
DBAAS_USER_PASSWORD	*****	

ページ 1 /2 (1-5 / 9 アイテム) 1 2

2. プロジェクト・ディレクトリで、env.propertiesファイルを作成します。

46 / 82



3. env.propertiesファイルを編集し、アプリケーションの環境変数を追加します。次のテンプレートを使用:

```

# ACCS environment variables (DO NOT REMOVE)
HOSTNAME=<app-name>-service:<port>
APP_HOME=/u01/app
PORT=8080
ORA_PORT=8080
ORA_APP_NAME=<app-name>

# Custom environment variables
<key>=<value>
<key>=<value>

# Service bindings environment variables
<key>=<value>
<key>=<value>
  
```

4. **<app-name>** プレースホルダーおよび **プレースホルダーに適切な値を指定します**。`値は、80,または443(SSLが有効な場合)です。ワーカー・アプリケーションの場合、ポートは80です 例:

```

# ACCS environment variables (DO NOT REMOVE)
HOSTNAME=notice-service:80
APP_HOME=/u01/app
PORT=8080
ORA_PORT=8080
ORA_APP_NAME=notice

# Custom environment variables
  
```

```
# Service bindings environment variables
DBAAS_DEFAULT_CONNECT_DESCRIPTOR=146.56.2.52:1521/PDB1.jpctest01.oraclecloud.
internal
DBAAS_USER_NAME=*****
DBAAS_USER_PASSWORD=*****
DBAAS_LISTENER_HOST_NAME=146.56.2.52
DBAAS_LISTENER_PORT=1521
DBAAS_DEFAULT_SID=ORCL
DBAAS_DEFAULT_SERVICE_NAME=PDB1.jpctest01.oraclecloud.internal
```

- アプリケーションがJava EEアプリケーションの場合は、Java EE Systemおよびサービス・バインディング・プロパティの構成を参照してください。
- Kubernetes構成マップを環境変数ファイルから作成します。 `<app-name>`および`<path-to-kubernetes-env-properties-file>`プレースホルダーを値で置き換えます。

```
kubectl create configmap
    <app-name>-config-var-map --from-env-file=<path-to-kubernetes-env-
properties-file>
```

例:

```
[root@accs-migration-poc-vm accs2oke]# kubectl create configmap notice-
config-var-map --from-env-file=/root/accs2oke/env.properties
configmap/notice-config-var-map created
```

Java EE Systemおよびサービス・バインディング・プロパティの構成

Java Enterprise Edition (Java EE)アプリケーションにシステムまたはサービス・バインディングのプロパティが必要な場合は、`env.properties`ファイルで指定する必要があります。

- Java EEアプリケーションのシステム・プロパティを使用するには、`env.properties`ファイル内で `EXTRA_JAVA_PROPERTIES`プロパティを定義します。

```
EXTRA_JAVA_PROPERTIES=<value>
```

- Java EEアプリケーションが次のJNDIサービス・バインディング・プロパティのいずれかを使用する場合は、`env.properties`ファイルに追加する必要があります。

- `jndi-name`
- `max-capacity`
- `min-capacity`
- `driver-properties`


```
<ocic-service-type>_SERVICE_BINDING_NAME=<service-name>
<ocic-service-type>_PROPERTIES=jndi-name:<jndi-name>|max-capacity:<max-
capacity>|min-capacity:<min-capacity>|driver-properties:<driver-properties>
```

プレースホルダー	説明
<code><ocic-service-type></code>	たとえば、サービス・タイプ: DBAAS, MYSQLCS, etc.
<code><service-name></code>	サービスの名前。
<code><jndi-name></code>	サービスのJNDI名。たとえば、" <code>jdbc/<value></code> "という形式にする必要があります: " <code>jdbc/dbcs</code> "。
<code><max-capacity></code>	接続プールの最大容量。
<code><min-capacity></code>	接続プールの最小容量。
<code><driver-properties></code>	JDBCドライバ・プロパティのセミコロン区切りのリスト。

例:

```
# ACCS environment variables(DO NOT REMOVE)
HOSTNAME=employees-service:443
APP_HOME=/u01/app
PORT=8080
ORA_PORT=8080
ORA_APP_NAME=employees

# Application environment variables
APP_LIB_FOLDER=./lib

# Service bindings environment variables

DBAAS_DEFAULT_CONNECT_DESCRIPTOR=146.56.2.52:1521/PDB1.jptest01.oraclecloud.internal
DBAAS_USER_NAME=oracleusr2
DBAAS_USER_PASSWORD=oracle
DBAAS_LISTENER_HOST_NAME=146.56.2.52
DBAAS_LISTENER_PORT=1521
DBAAS_DEFAULT_SID=ORCL
DBAAS_DEFAULT_SERVICE_NAME=PDB1.jptest01.oraclecloud.internal

# System properties
# Only for "Java EE" runtime. Remove for other runtimes.
EXTRA_JAVA_PROPERTIES=-DconfigPath=/u01/app/conf/ -DlogFile=/u01/app/logs/app.log

# Service binding properties
# Only for "Java EE" runtime. Remove for other runtimes.
DBAAS_SERVICE_BINDING_NAME=DBCSDemo
```

```
DBAAS_PROPERTIES=jndi-name:jdbc/testds|max-capacity:5|min-capacity:1|
#DBAAS_PROPERTIESのところはuserとdatabaseは不要です
```

Kubernetes ClusterとOracle Cloud Services間の接続の有効化

Oracle Application Container Cloud Serviceのアプリケーションがサービス・バインディングを使用して、他のOracle Cloudサービスとの通信を有効にする場合、アプリケーションの移行後にそれらのサービスと通信できるようにする必要があります。

次の2つのシナリオがあります:

1. 使用しているサービスがOracle Cloud Infrastructure Classicにある場合は、Oracle Cloud Infrastructure Classicサービスで、KubernetesクラスタのノードにアタッチされているNAT GatewayのパブリックIPアドレスをサービスに接続できるようにするアクセス・ルールを作成します。たとえば、アプリケーションでOracle Database Cloud Serviceを使用する場合は、[Database Cloud Serviceへのネットワーク・アクセスの管理](#)を参照してください。

注意: NAT GatewayのパブリックIPアドレスは、メニューからOracle Cloud Infrastructureコンソールに配置できます: 開発者サービス、コンテナ・クラスタ(OKE)、クラスタ詳細、ノード・プール・セクション、ノード・インスタンスの詳細、仮想クラウド・ネットワーク詳細、NATゲートウェイ、パブリックIPアドレス。

2. サービスがOracle Cloud Infrastructureにある場合は、サービスがデプロイされているVCNおよびサブネットを探します。Kubernetesクラスタからサービスへのトラフィックを可能にするために、インGRESS・セキュリティ・ルールが存在することを確認してください。Oracle Cloud Infrastructureのドキュメントの[セキュリティ・リスト](#)を参照してください。

Kubernetes構成ファイルの作成

Oracle Cloud Infrastructure Container Engine for Kubernetesにアプリケーションをデプロイする前に、アプリケーションのデプロイメントとサービスの構成ファイルを作成する必要があります。

デプロイメントおよびサービス構成ファイルは、Kubernetesでアプリケーションのインスタンスを作成および更新する手順を提供します。デプロイメントを作成および管理するには、Kubernetesコマンドライン・インタフェースを使用します。

デプロイメント構成ファイルの作成

1. 次のテンプレートを使用してdeployment.yamlファイルを作成します:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: "<app-name>-deployment"
spec:
  replicas: ${replicas}
  selector:
    matchLabels:
      app: "<app-name>-selector"
```

```

template:
  metadata:
  labels:
    app: "<app-name>-selector"
  spec:
  containers:
    - name: "<app-name>"
      image: "<region-code>.ocir.io/<tenancy>/accs/<oci-account-username>/<app-name>:latest"
      command: ["${command}"]
      args: ["${args}"]
      ports:
        - containerPort: 8080
      env:
        - name: ORA_INSTANCE_NAME
      valueFrom:
        fieldRef:
          fieldPath: metadata.name
      envFrom:
        - configMapRef:
            name: "<app-name>-config-var-map"
      resources:
      limits:
        memory: "${memory}i"
      requests:
        memory: "${memory}i"
      # The following section "livenessProbe" should be removed if Health
      Check URL
      # is not available.
      livenessProbe:
      httpGet:
        path: "${healthCheckHttpPath}"
        port: 8080
      initialDelaySeconds: 5
      periodSeconds: 600
      timeoutSeconds: 30
      failureThreshold: 3
      imagePullSecrets:
        - name: "<app-name>-secret"

```

2. `<app-name>`, `<region-code>`, `<tenancy>` プレースホルダーおよび `<oci-account-username>` プレースホルダーに適切な値を指定します。
3. 変数に適切な値を指定します:

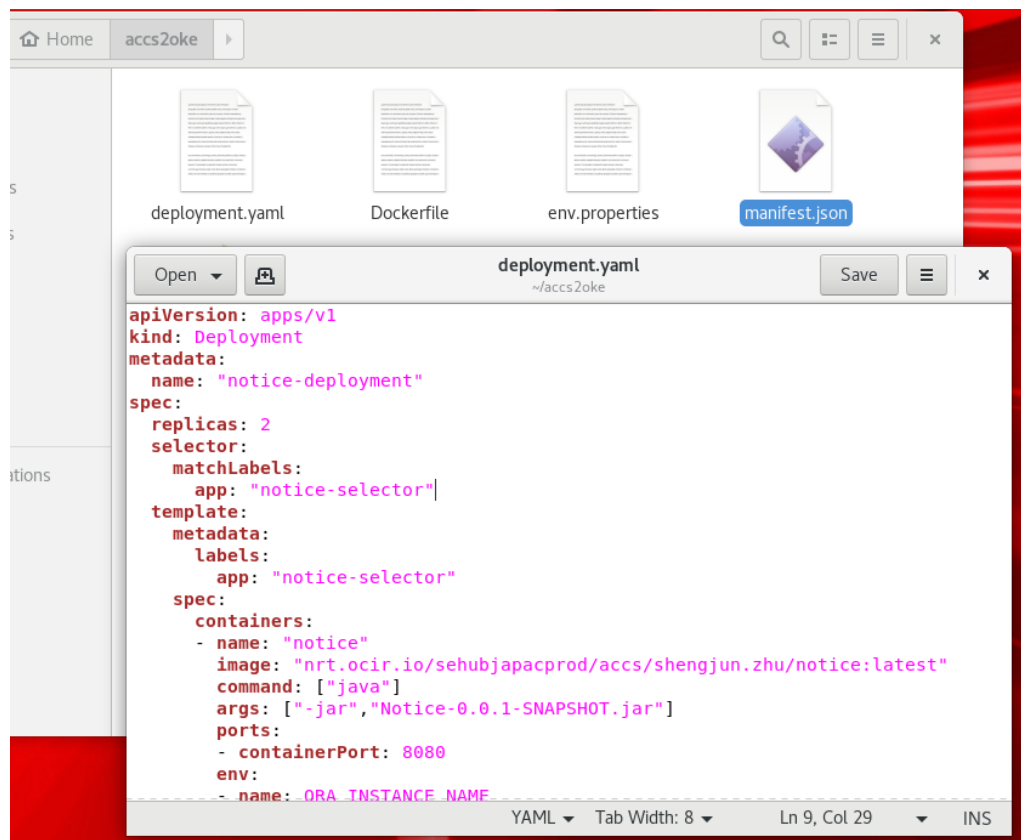
プレースホルダー	説明
<code>\${replicas}</code>	アプリケーション・レプリカの数指定します。このプロパティが欠落している場合、デフォルト値は2です。

プレースホルダー	説明
<code>\${command}</code>	<p>アプリケーションの起動に使用するオペレーティング・システム・コマンドを指定します。</p> <ul style="list-style-type: none"> manifest.jsonファイル内の<code>command</code>プロパティの最初のワードを入力します。たとえば、コマンド・プロパティの値が:<code>"sh bin/startapp.sh -config conf/app.properties"</code>の場合、<code>\${command}</code>値は<code>sh</code>です。 manifest.jsonファイルにこのプロパティが存在しない場合は、値を空の<code>cmd: []</code>のままにします。
<code>\${args}</code>	<p>アプリケーションの起動に使用する引数を指定します。</p> <ul style="list-style-type: none"> manifest.jsonファイル内の<code>command</code>プロパティの2番目の単語から始まるすべての単語のカンマ区切りのリストを入力します。たとえば、manifest.jsonファイル内の<code>command</code>プロパティの値が<code>"sh bin/startapp.sh -config conf/app.properties"</code>の場合、<code>\${args}</code>値は<code>"bin/startapp.sh", "-config", "conf/app.properties"</code>になります。 このプロパティが欠落している場合は、値を空のままにします。たとえば、<code>args: []</code>。
<code>\${memory}</code>	<p>アプリケーションに必要なメモリーの量を指定します。このプロパティが欠落している場合、デフォルト値は2Gです。</p>
<code>\${healthCheckHttpPath}</code>	<p>アプリケーションのヘルス・チェックの実行に使用するHTTPパスを指定します。このHTTPパスのURLは、200以上のHTTPレスポンス・コードを返し、400未満では成功を示す必要があります。</p> <ul style="list-style-type: none"> manifest.json ファイルに<code>healthCheck.http-endpoint</code>プロパティを入力します。 manifest.jsonファイルにこのプロパティが存在しない場合は、値<code>"/"</code>を入力します。

例:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: "notice-deployment"
spec:
  replicas: 2
  selector:
    matchLabels:
      app: "notice-selector"
  template:
    metadata:
      labels:
        app: "notice-selector"
    spec:
      containers:
        - name: "notice"
```

```
    image: "nrt.ocir.io/sehubjapacprod/accs/shengjun.zhu/notice:latest"
    command: ["java"]
    args: ["-jar","Notice-0.0.1-SNAPSHOT.jar"]
    ports:
      - containerPort: 8080
    env:
      - name: ORA_INSTANCE_NAME
    valueFrom:
      fieldRef:
        fieldPath: metadata.name
    envFrom:
      - configMapRef:
          name: "notice-config-var-map"
    resources:
    limits:
      memory: "2Gi"
    requests:
      memory: "2Gi"
    # The following section "livenessProbe" should be removed if Health
Check URL
    # is not available.
    livenessProbe:
      httpGet:
        path: "/"
        port: 8080
      initialDelaySeconds: 5
      periodSeconds: 600
      timeoutSeconds: 30
      failureThreshold: 3
    imagePullSecrets:
      - name: "notice-secret"
```



サービス構成ファイルの作成

1. 次のテンプレートを使用してservice.yamlファイルを作成します:

```
kind: Service
apiVersion: v1
metadata:
  name: "<app-name>-service"
  # The following section "annotations" should be removed if SSL endpoint
  # is not required.
  annotations:
    service.beta.kubernetes.io/oci-load-balancer-ssl-ports: '443'
    service.beta.kubernetes.io/oci-load-balancer-tls-secret: "<app-name>-
    tls-certificate"
spec:
  type: "${type}"
ports:
  - port: ${port}
    protocol: TCP
    targetPort: 8080
selector:
  app: "<app-name>-selector"
```

2. <app-name>プレースホルダーに適切な値を指定します。
3. \${variable}プレースホルダーに適切な値を指定します:

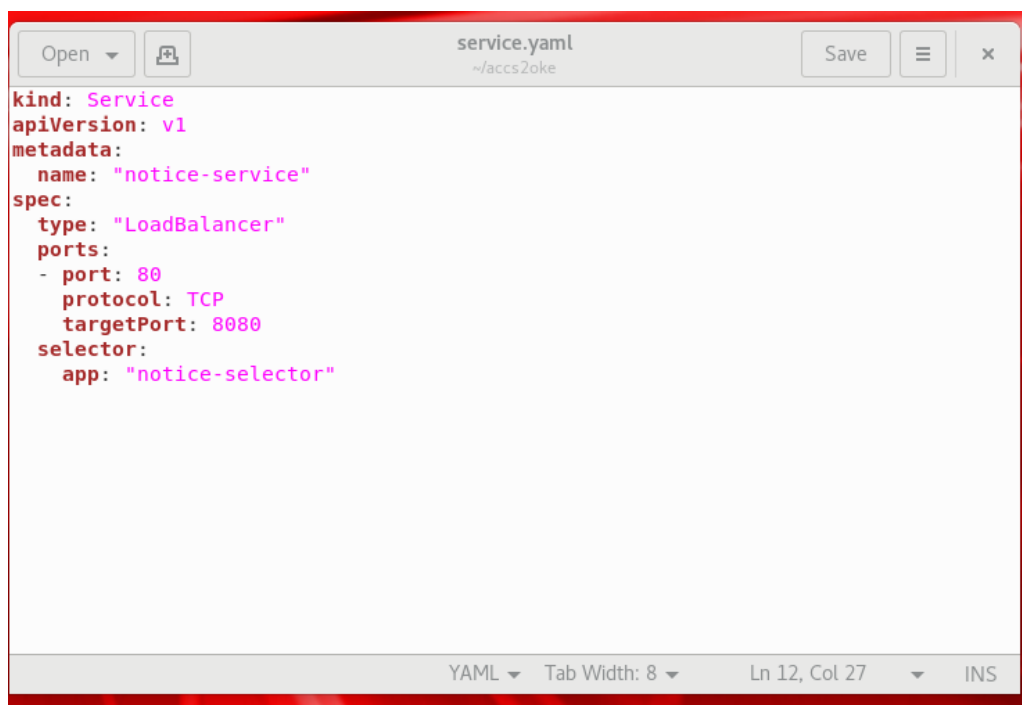
プレー スホルダー 説明

アプリケーション・タイプを指定します。 Oracle Application Container Cloud Serviceアプリケーションのタイプがwebの場合は、値LoadBalancerを入力します。 アプリケーションのタイプがworkerの場合は、値ClusterIPを入力します。

アプリケーションのパブリック・ポートを指定します。 SSLエンドポイントが必要な場合は443を入力し、それ以外の場合は80を入力します。 ワーカー・アプリケーションの場合、値は80です。

例: JAVASE

```
kind: Service
apiVersion: v1
metadata:
  name: "notice-service"
spec:
  type: "LoadBalancer"
  ports:
  - port: 80
    protocol: TCP
    targetPort: 8080
  selector:
    app: "notice-selector"
```



例: JAVAEE(SSLを設定する)

```

kind: Service
apiVersion: v1
metadata:
  name: "employees-service"
  # The following section "annotations" should be removed if SSL endpoint
  # is not required.
  annotations:
    service.beta.kubernetes.io/oci-load-balancer-ssl-ports: '443'
    service.beta.kubernetes.io/oci-load-balancer-tls-secret: "employees-tls-
certificate"
  spec:
    type: LoadBalancer
  ports:
    - port: 443
      protocol: TCP
      targetPort: 8080
  selector:
    app: "employees-selector"

```

Docker Registry SecretおよびSSL証明書を設定

アプリケーションのデプロイ時にKubernetesがOracle Cloud Infrastructureレジストリからイメージをプルするには、Kubernetesシークレットを作成する必要があります。アプリケーションがSSLエンドポイントを必要とする場合、アプリケーションの証明書および秘密キーを使用してTLSシークレットを作成する必要があります。

シークレットには、`docker login`コマンド(認証トークンを含む)を使用してOracle Cloud Infrastructureレジストリに手動でログインした場合と同じ詳細がすべて含まれます。

1. Dockerレジストリ・シークレットを作成するには、コマンド行ウィンドウで、次のコマンドを実行します。 `<app-name>`, `<region-code>`, `<tenancy>`, `<oci-account-username>`, プレースホルダーおよび `<auth-token>` プレースホルダーに適切な値を指定します。

```

kubectl create secret
  docker-registry <app-name>-secret --docker-server="<region-
code>.ocir.io" --docker-username=<tenancy>/<oci-account-username>
  --docker-password='<auth-token>' --docker-email=<oci-account-
username>

```

例:

```

[root@accs-migration-poc-vm accs2oke]# kubectl create secret docker-registry
notice-secret --docker-server="nrt.ocir.io" --docker-
username=sehubjapacprod/oracleidentitycloudservice/shengjun.zhu@oracle.com -
-docker-password='6h.}82DrE4+k#sfDa)3<' --docker-
email=shengjun.zhu@oracle.com
secret/notice-secret created

```


2. TLSシークレットを作成するには、コマンド行ウィンドウで、次のコマンドを実行します。 `<app-name>`, `<path-to-tls-key-file>` プレースホルダーおよび `<path-to-tls-cert-file>` プレースホルダーに適切な値を指定します。

```
kubectl create secret tls  
    <app-name>-tls-certificate --key <path-to-tls-key-file> --cert  
    <path-to-tls-cert-file>
```

例:(JAVAEE用SSL証明書)

```
$ kubectl create secret tls employees-tls-certificate --key  
/home/user1/kubernetes/tls.key --cert /home/user1/kubernetes/tls.crt  
secret/webapp01-tls-certificate created
```

注意: `<path-to-tls-cert-file>` と `<path-to-tls-key-file>` は、公開証明書とキー・ファイルへの絶対パスです。

アプリケーションのデプロイ

`deployment.yaml` ファイルおよび `service.yaml` ファイルを使用して、アプリケーションをデプロイします。

アプリケーションのHTTPSリダイレクション、ロード・バランサ・ポリシーまたはセッション固定性にHTTPをカスタマイズする場合は、「イングレス・コントローラを配置」で説明するステップを使用してアプリケーションをデプロイする必要があります。

1. Kubernetesデプロイメントを作成します。

```
kubectl create -f <path-to-kubernetes-deployment-yaml>
```

例:

```
[root@accs-migration-poc-vm accs2oke]# kubectl create -f deployment.yaml  
deployment.apps/notice-deployment created
```

2. デプロイメントのロールアウト・ステータスを確認してください。

```
kubectl rollout status deployment <app-name>-deployment
```

例:

```
[root@accs-migration-poc-vm accs2oke]# kubectl rollout status deployment
notice-deployment
deployment "notice-deployment" successfully rolled out
```

デプロイ・チェックの完了には数分かかる場合があります。

3. Kubernetesサービスを作成します。

```
kubectl create -f <path-to-kubernetes-service-yaml>
```

例:

```
[root@accs-migration-poc-vm accs2oke]# kubectl create -f service.yaml
service/notice-service created
```

4. サービスのステータスを確認してください。

```
kubectl get svc <app-name>-service
```

例:

```
[root@accs-migration-poc-vm accs2oke]# kubectl get svc notice-service
NAME                TYPE                CLUSTER-IP      EXTERNAL-IP      PORT(S)
AGE
notice-service      LoadBalancer       10.96.230.153    140.238.46.116   80:31246/TCP      90s
```

サービスの開始には数分かかる場合があります。完了したら、EXTERNAL-IP列の下にパブリックIPアドレスをメモします。

注意: クラスター・アプリケーションをデプロイした場合、サービス・インスタンスは、サービス名を使用して相互に通信できます。サービス名は、HOSTNAME環境変数としてアプリケーションで使用可能になります。

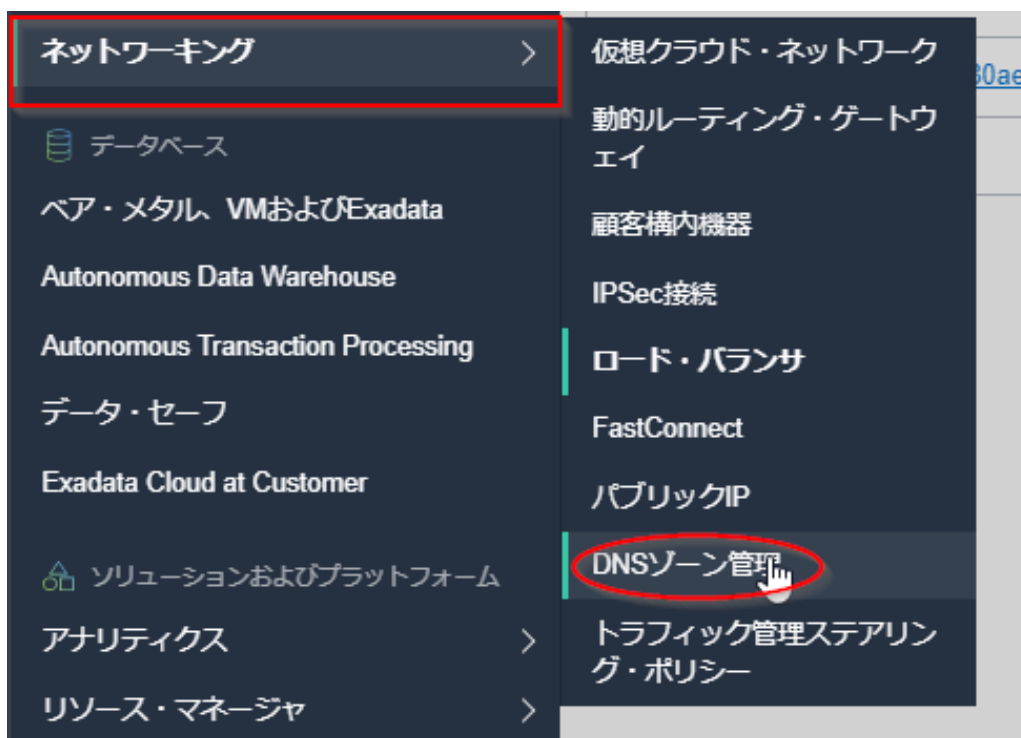
カスタムURLの設定

アプリケーションにカスタムURLを使用する場合は、パブリック・ドメイン名が必要です。ドメイン名をアプリケーションのパブリックIPアドレスにマップする必要があります。

DNSゾーンの作成

ドメイン・ネーム・システム(DNS)のゾーンは、特定の組織または管理者によって管理されるグローバルDNSの連続した部分です。すでにDNSゾーンを作成している場合、次のステップは必要ありません。

1. Oracle Cloud Infrastructureコンソールから、ナビゲーション・メニューを開きます。「コア・インフラストラクチャ」の下で「ネットワーキング」に移動し、「DNSゾーン管理」をクリックします。



2. 「ゾーンの作成」をクリックします。



3. 「ゾーン名」を入力し、他のフィールドのデフォルト値をそのまま使用します。「送信」をクリックします

ゾーンの作成 [ヘルプ](#) [取消](#)

メソッド
手動

ゾーン・タイプ ⓘ
プライマリ

ゾーン名 ⓘ
taosheng.tk

タグ付けとは、テナンシ内のリソースを整理およびトラッキングできるメタデータ・システムです。タグは、リソースにアタッチできるキーと値から構成されます。
[タグ付けの詳細](#)

タグ・ネームスペース タグ・キー 値

なし(フリーフォーム・タグの追加) + 追加タグ

[送信](#) [取消](#)

注意: DNSゾーンのDNSレコードをホストするネーム・サーバーのリストが表示されます。

ORACLE Cloud

ネットワーキング » DNS - ゾーン » ゾーンの詳細

DNS - taosheng.tk

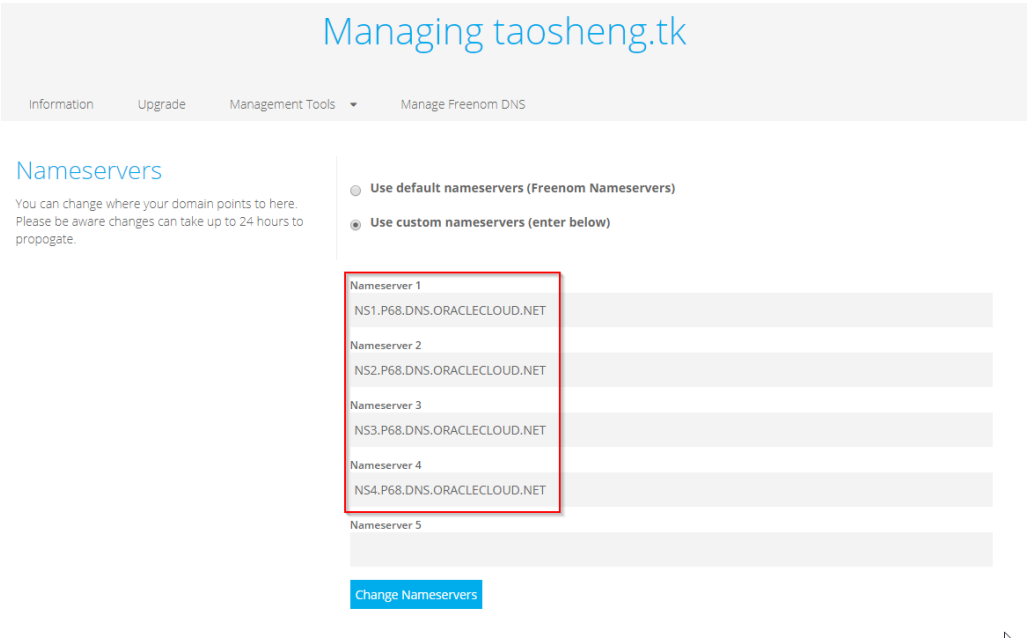
[タグの追加](#) [新規コンパートメントの選択](#)

[ゾーン情報](#) [タグ](#)

ゾーン情報

タイプ: PRIMARY
シリアル: 1
作成日: 2019年10月10日(木) 7:28:07 UTC
OCID: ...dd257e [表示](#) [コピー](#)
ネーム・サーバー: ns1.p68.dns.oraclecloud.net, ns2.p68.dn

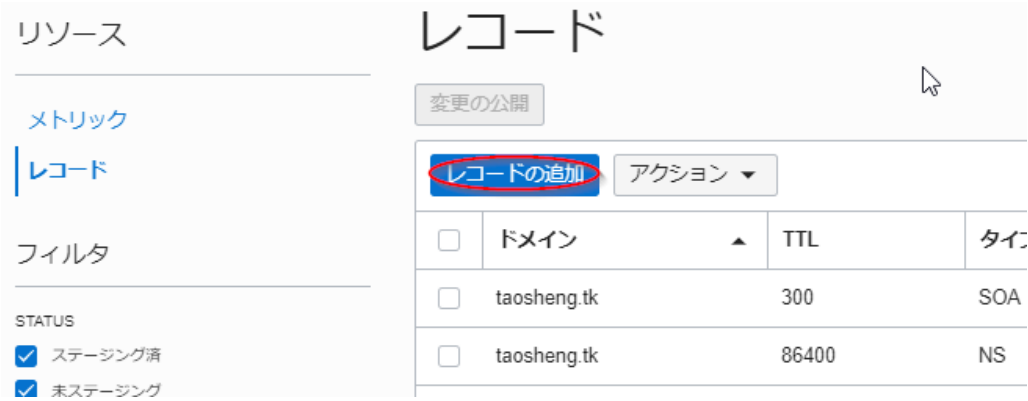
4. 公開DNSホスティング・プロバイダ・サービスにネーム・サーバーを追加してください。



DNSレコードの追加

DNSゾーンを作成したら、DNSレコードを追加して、サブドメイン名を作成する必要があります。

- 1. ゾーン情報ページで、レコードの追加、をクリックし、次の値を入力または選択します：
 - 1. レコード型: A - IPv4アドレス
 - 2. 名前: アプリケーション名を入力します。
 - 3. TTL: 適当な数字を入力します。
 - 4. アドレス: アプリケーションのパブリックIPアドレスを入力します。



レコードの追加

レコード型
A - IPv4アドレス

ホスト名をIPv4アドレスに関連付けるために使用するホスト・レコード。

名前 オプション
notice .taosheng.tk

TTL
300 TTL単位
秒

notice.taosheng.tkのすべてのAレコードは、TTLに対する最終変更を反映するように更新されます [ロック](#)

RDATAモード
基本

ADDRESS (アドレス)
140.238.46.116

☐ 別のレコードの追加

[送信](#) [取消](#)

2. 「送信」をクリックします。
3. 「変更の公開」をクリックします。

リソース

メトリック

レコード

フィルタ

ステータス

レコード

[変更の公開](#)

[レコードの追加](#) [アクション ▼](#)

☐ ドメイン

☐ notice.taosheng.tk

4. アプリケーションのカスタムURLは、次のフォームを備えています: `http[s]://<name>.<zone-name>[:<port>]/`。たとえば、レコードの名前にwebapp01を入力し、ゾーン名がexample.comの場合、HTTPS URLは`https://webapp01.example.com`になります。

インGRESS・コントローラを配置

アプリケーションに対して、HTTPをHTTPSにリダイレクト、ロード・バランサまたはセッション・スティッキーにカスタマイズする場合は、NginxインGRESS・コントローラを設定する必要があります。インGRESS・コントローラは、外部ロード・バランサとアプリケーション間のトラフィックを処理します。

ロール・ベース・アクセスの構成

cluster-adminロールを作成し、Oracle Cloud Infrastructure管理者ユーザーに割り当てます。次に、rbac.yamlファイルを使用して、サービス・アカウント、クラスター・ロールおよびクラスター・ロール・バインディングを作成します。

1. cluster-adminロールをOracle Cloud Infrastructure管理者ユーザーに割り当てるには、コマンド・ライン・ウィンドウで、次のコマンドを実行します。 `<user-ocid>`プレースホルダーに適切な値を指定します。

```
kubectl create clusterrolebinding ingress-binding --clusterrole=cluster-admin --user=<user-ocid>
```

例:

```
[root@accs-migration-poc-vm accs2oke]# kubectl create clusterrolebinding ingress-binding --clusterrole=cluster-admin --user=ocid1.user.oc1..aaaaaaa7xr3xmknhocrigquyummkj7onlfdoiwc126l4qr7jsyut7kl4sgq clusterrolebinding.rbac.authorization.k8s.io/ingress-binding created
```

2. 次のコンテンツを使用してrbac.yamlファイルを作成します:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: nginx-ingress-serviceaccount
  namespace: default

---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRole
metadata:
  name: nginx-ingress-clusterrole
rules:
- apiGroups:
  - ""
  resources:
  - configmaps
  - endpoints
  - nodes
  - pods
  - secrets
  verbs:
  - list
  - watch
- apiGroups:
  - ""
  resources:
  - nodes
  verbs:
  - get
- apiGroups:
  - ""
```

```

    resources:
    - services
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - "extensions"
  resources:
  - ingresses
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - ""
  resources:
  - events
  verbs:
  - create
  - patch
- apiGroups:
  - "extensions"
  resources:
  - ingresses/status
  verbs:
  - update

---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: Role
metadata:
name: nginx-ingress-role
rules:
- apiGroups:
  - ""
  resources:
  - configmaps
  - pods
  - secrets
  - namespaces
  verbs:
  - get
- apiGroups:
  - ""
  resources:
  - configmaps
  resourceNameNames:
  # Defaults to "<election-id>-<ingress-class>"
  # Here: "<ingress-controller-leader>-<nginx>"
  # This has to be adapted if you change either parameter
  # when launching the nginx-ingress-controller.
  - "ingress-controller-leader-nginx"
  verbs:

```



```

    - get
    - update
  - apiGroups:
    - ""
    resources:
    - configmaps
    verbs:
    - create
  - apiGroups:
    - ""
    resources:
    - endpoints
    verbs:
    - get
    - create
    - update

---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: RoleBinding
metadata:
  name: nginx-ingress-role-nisa-binding
  roleRef:
    apiGroup: rbac.authorization.k8s.io
    kind: Role
    name: nginx-ingress-role
  subjects:
  - kind: ServiceAccount
    name: nginx-ingress-serviceaccount

---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: nginx-ingress-clusterrole-nisa-binding
  roleRef:
    apiGroup: rbac.authorization.k8s.io
    kind: ClusterRole
    name: nginx-ingress-clusterrole
  subjects:
  - kind: ServiceAccount
    name: nginx-ingress-serviceaccount
    namespace: default

---

```

3. サービス・アカウント、クラスター・ロールおよびサービス・アカウントとクラスター・ロール間のクラスター・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。 `<path-to-rbac-yaml>` プレースホルダーに適切な値を指定します。

```
kubectl create -f <path-to-rbac-yaml>
```

例:

```
[root@accs-migration-poc-vm accs2oke]# kubectl create -f rbac.yaml
serviceaccount/nginx-ingress-serviceaccount created
clusterrole.rbac.authorization.k8s.io/nginx-ingress-clusterrole created
role.rbac.authorization.k8s.io/nginx-ingress-role created
rolebinding.rbac.authorization.k8s.io/nginx-ingress-role-nisa-binding
created
clusterrolebinding.rbac.authorization.k8s.io/nginx-ingress-clusterrole-nisa-
binding created
```

デフォルト・バックエンドの作成

デフォルト・バックエンドのKubernetesデプロイメントとサービスを作成します。

1. 次のコンテンツを使用してdefault-deployment.yamlファイルを作成します:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: default-http-backend
  labels:
    app: default-http-backend
spec:
  replicas: 1
  selector:
    matchLabels:
      app: default-http-backend
  template:
    metadata:
      labels:
        app: default-http-backend
    spec:
      terminationGracePeriodSeconds: 60
      containers:
        - name: default-http-backend
          # Any image is permissable as long as:
          # 1. It serves a 404 page at /
          # 2. It serves 200 on a /healthz endpoint
          image: gcr.io/google_containers/defaultbackend:1.0
          livenessProbe:
            httpGet:
              path: /healthz
              port: 8080
              scheme: HTTP
          initialDelaySeconds: 30
          timeoutSeconds: 5
          ports:
            - containerPort: 8080
```

```
resources:
limits:
  memory: 128Mi
requests:
  memory: 128Mi
```

2. デフォルト・バックエンド用のKubernetesデプロイメントを作成します。

```
kubectl create -f <path-to-default-deployment-yaml>
```

例:

```
[root@accs-migration-poc-vm accs2oke]# kubectl create -f default-
deployment.yaml
deployment.apps/default-http-backend created
```

3. デプロイメントのロールアウト・ステータスを確認してください。

```
kubectl rollout status deployment default-http-backend
```

例:

```
[root@accs-migration-poc-vm accs2oke]# kubectl rollout status deployment
default-http-backend
deployment "default-http-backend" successfully rolled out
```

デプロイ・チェックの完了には数分かかる場合があります。

4. 次のコンテンツを使用してdefault-service.yamlファイルを作成します:

```
apiVersion: v1
kind: Service
metadata:
name: default-http-backend
labels:
  app: default-http-backend
spec:
ports:
- port: 80
  targetPort: 8080
selector:
  app: default-http-backend
```

- デフォルト・バックエンドのKubernetesサービスを作成します。

```
kubectl create -f <path-to-default-service-yaml>
```

例:

```
[root@accs-migration-poc-vm accs2oke]# kubectl create -f default-  
service.yaml  
service/default-http-backend created
```

- サービスの展開ステータスを確認してください。

```
kubectl get svc default-http-backend
```

例:

```
[root@accs-migration-poc-vm accs2oke]# kubectl get svc default-http-backend  
NAME                                TYPE                CLUSTER-IP      EXTERNAL-IP    PORT(S)  
AGE  
default-http-backend    ClusterIP    10.96.189.119   <none>         80/TCP  
36s
```

デプロイメントの完了には数分かかる場合があります。

インGRESS・コントローラを作成

NginxインGRESS・コントローラ用のKubernetesデプロイメントとサービスを作成します。

インGRESS・コントローラ・デプロイメントを作成

- 次のテンプレートを使用してnginx-deployment.yamlファイルを作成します。

```
apiVersion: apps/v1  
kind: Deployment  
metadata:  
  name: nginx-ingress-controller  
  labels:  
    app: nginx-ingress-controller  
spec:  
  replicas: 1  
  selector:  
    matchLabels:  
      app: nginx-ingress-controller  
template:
```

```
metadata:
labels:
  app: nginx-ingress-controller
spec:
terminationGracePeriodSeconds: 60
serviceAccountName: nginx-ingress-serviceaccount
containers:
- image: quay.io/kubernetes-ingress-controller/nginx-ingress-
controller:0.23.0
  name: nginx-ingress-controller
  readinessProbe:
    httpGet:
      path: /healthz
      port: 10254
      scheme: HTTP
  livenessProbe:
    httpGet:
      path: /healthz
      port: 10254
      scheme: HTTP
  initialDelaySeconds: 10
  timeoutSeconds: 1
  ports:
  - containerPort: 80
    hostPort: 80
  - containerPort: 443
    hostPort: 443
  env:
  - name: POD_NAME
    valueFrom:
      fieldRef:
        fieldPath: metadata.name
  - name: POD_NAMESPACE
    valueFrom:
      fieldRef:
        fieldPath: metadata.namespace
  args:
  - /nginx-ingress-controller
  - --default-backend-service=$(POD_NAMESPACE)/default-http-backend
```

2. NginxインGRESS・コントローラ用のKubernetesデプロイメントを作成します。

```
kubectl create -f <path-to-nginx-deployment-yaml>
```

例:

```
[root@accs-migration-poc-vm accs2oke]# kubectl create -f nginx-
deployment.yaml
deployment.apps/nginx-ingress-controller created
```

3. デプロイメントのロールアウト・ステータスを確認してください。

```
kubectl rollout status deployment nginx-ingress-controller
```

例:

```
[root@accs-migration-poc-vm accs2oke]# kubectl rollout status deployment
nginx-ingress-controller
Waiting for deployment "nginx-ingress-controller" rollout to finish: 0 of 1
updated replicas are available...
deployment "nginx-ingress-controller" successfully rolled out
```

デプロイ・チェックの完了には数分かかる場合があります。

イングレス・コントローラ・サービスを作成

1. 次のテンプレートをを使用してnginx-service.yamlファイルを作成します:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-ingress-controller
  namespace: default
  labels:
    app: nginx-ingress-controller
spec:
  type: LoadBalancer
  ports:
    - port: 80
      name: http
  selector:
    app: nginx-ingress-controller
```

2. Nginxイングレス・コントローラ用のサービスを作成するには、次のコマンドを実行します。 <path-to-nginx-service-yaml> プレースホルダーに適切な値を指定します。

```
kubectl create -f <path-to-nginx-service-yaml>
```

例:

```
[root@accs-migration-poc-vm accs2oke]# kubectl create -f nginx-service.yaml
service/nginx-ingress-controller created
```

3. サービス・ステータスを確認してください。

```
kubectl get svc nginx-ingress-controller
```

例:

```
[root@accs-migration-poc-vm accs2oke]# kubectl get svc nginx-ingress-controller
NAME                                TYPE                CLUSTER-IP      EXTERNAL-IP
PORT(S)          AGE
nginx-ingress-controller  LoadBalancer    10.96.70.245    158.101.128.210
80:32116/TCP      59s
```

サービスの開始には数分かかる場合があります。完了したら、EXTERNAL-IP列の下にパブリックIPアドレスを書き込みます。

4. Nginxイングレス・コントローラが正しく構成されていることを確認してください。

```
curl -I http://<public-ip-address>/healthz
```

例:

```
[root@accs-migration-poc-vm accs2oke]# curl -I
http://158.101.128.210/healthz
HTTP/1.1 200 OK
Server: nginx/1.15.9
Date: Thu, 10 Oct 2019 09:20:42 GMT
Content-Type: text/html
Content-Length: 0
Connection: keep-alive
```

アプリケーションをデプロイしてイングレス・リソースを作成した後、パブリックURLを使用してアプリケーションにアクセスできます。

イングレス・コントローラ設定でのアプリケーションのデプロイ

「アプリケーションのデプロイ」に記載されているステップを使用して、アプリケーションをデプロイします。サービスを作成するには、次の構成を使用します:

1. 次のテンプレートをを使用してservice.yamlファイルを作成します:

```
kind: Service
apiVersion: v1
```

```
metadata:
  name: "<app-name>-service"
spec:
  type: ClusterIP
  ports:
    - port: 80
      protocol: TCP
      targetPort: 8080
  selector:
    app: "<app-name>-selector"
```

2. **<app-name>** プレースホルダーに適切な値を指定します。 例:

```
kind: Service
apiVersion: v1
metadata:
  name: "notice-service"
spec:
  type: "ClusterIP"
  ports:
    - port: 80
      protocol: TCP
      targetPort: 8080
  selector:
    app: "notice-selector"
```

3. Kubernetesサービスを作成します。

```
kubectl create -f <path-to-kubernetes-service-yaml>
```

注意: アプリケーションのデプロイで既に同じ名前の**Service**を作成した場合、事前に削除してください。

```
[root@accs-migration-poc-vm accs2oke]# kubectl delete -f service.yaml
service "notice-service" deleted
```

例:

```
[root@accs-migration-poc-vm accs2oke]# kubectl create -f service.yaml
service/notice-service created
```

4. サービスのステータスを確認してください。


```
kubectl get svc <app-name>-service
```

例:

```
[root@accs-migration-poc-vm accs2oke]# kubectl get svc notice-service
NAME                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
notice-service      ClusterIP     10.96.29.91   <none>         80/TCP     27s
```

サービスの開始には数分かかる場合があります。

インGRESS・リソースの作成

NginxインGRESS・コントローラがトラフィックをアプリケーションに転送する際に使用するインGRESS・リソースを作成します。

1. (オプション)アプリケーションがSSLエンドポイントを必要とする場合、次のコマンドを実行して、TLSキーおよび証明書のシークレットを作成します。 `<app-name>`, `<path-to-tls-key-file>`, プレースホルダーと `<path-to-tls-cert-file>` プレースホルダーに適切な値を指定します。

```
kubectl create secret tls ingress-tls-certificate --key <path-to-tls-key-file> --cert <path-to-tls-cert-file>
```

例:

```
$ kubectl create secret tls ingress-tls-certificate --key
/home/user1/kubernetes/tls.key --cert /home/user1/kubernetes/tls.crt
secret/ingress-tls-certificate created
```

2. 次のテンプレートをを使用して `nginx-ingress.yaml` ファイルを作成します:

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingress-resource
  annotations:
    kubernetes.io/ingress.class: "nginx"
    nginx.ingress.kubernetes.io/load-balance: "round_robin"
spec:
  # Required only for SSL endpoint
  tls:
    - hosts:
      - <host-name>
      secretName: ingress-tls-certificate
  rules:
```

```
- host: <host-name>
http:
  paths:
    - path: /
  backend:
    serviceName: <app-name>-service
    servicePort: 80
```

3. **<app-name>** プレースホルダーに適切な値を指定します。 例:

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingress-resource
  annotations:
    kubernetes.io/ingress.class: "nginx"
    nginx.ingress.kubernetes.io/load-balance: "round_robin"
spec:
  rules:
    - host: notice.taosheng.tk
      http:
        paths:
          - path: /
        backend:
          serviceName: notice-service
          servicePort: 80
```

4. イングレス・リソースを作成するには、コマンド行ウィンドウで、次のコマンドを実行します。
<path-to-nginx-ingress-yaml> プレースホルダーに適切な値を指定します。

```
kubectl create -f <path-to-nginx-ingress-yaml>
```

例:

```
[root@accs-migration-poc-vm accs2oke]# kubectl create -f nginx-ingress.yaml
ingress.extensions/ingress-resource created
```

5. イングレス・リソース・ステータスを確認するには、次のコマンドを実行します。

```
kubectl get ingress ingress-resource
```

例:

```
[root@accs-migration-poc-vm accs2oke]# kubectl get ingress ingress-resource
NAME                HOSTS                ADDRESS    PORTS    AGE
ingress-resource    notice.taosheng.tk    80        11s
```

6. Nginxイングレス・コントローラ・サービスのパブリックURLを使用して、アプリケーションをテストします。

イングレス・リソースのカスタマイズ

nginx-ingress.yamlファイル内のHTTPSリダイレクション、ロード・バランサ・ポリシーおよびセッション・スティックネスにHTTPをカスタマイズできます。

- HTTPからHTTPSへのリダイレクトはデフォルトで有効になっています。無効にする場合は、注釈セクションで次を追加: `ingress.kubernetes.io/ssl-redirect:False`。
- デフォルトのロード・バランサ・ポリシーはround_robinです。ip_hashロード・バランサ・ポリシーを使用するには、注釈を使用: `nginx.ingress.kubernetes.io/upstream-hash-by: "$binary_remote_addr"`。
- セッションの固定性を有効にするには、ip_hashをロード・バランサ・ポリシーとして使用する必要があります。

複数のアプリケーションのイングレス・リソースの作成

Kubernetesクラスタ内に複数のアプリケーションをデプロイする場合は、同じイングレス・コントローラを使用して、それらのアプリケーションのロード・バランシングを構成できます。名前ベースの仮想ホスティングを構成できます。その中に、Nginxイングレス・コントローラ・サービスの1つのIPアドレスが複数のサービスのホスト名にトラフィックをルーティングします。

例:

webapp01-serviceとwebapp02-serviceの2つのサービスがKubernetesクラスタ内に作成されています。DNS名webapp01.example.comとwebapp02.example.comは、イングレス・サービスのパブリックIPアドレスにマップされます。次に、<https://webapp01.example.com/>へのリクエストがwebapp01-serviceに転送され、<https://webapp02.example.com/>へのリクエストがwebapp02-serviceに転送されるようにイングレス・コントローラを設定できます。これら2つのサービスは、nginx-ingress.yamlファイルで構成できます。

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingress-resource
  annotations:
    kubernetes.io/ingress.class: "nginx"
    nginx.ingress.kubernetes.io/load-balance: "round_robin"
spec:
  # Required only for SSL endpoint

  tls:
    - hosts:
        - webapp01.example.com
        - webapp02.example.com
```

```
secretName: ingress-tls-certificate
rules:
- host: webapp01.example.com
  http:
    paths:
    - path: /
      backend:
        serviceName: webapp01-service
        servicePort: 80
- host: webapp02.example.com
  http:
    paths:
    - path: /
      backend:
        serviceName: webapp02-service
        servicePort: 80
```

Kubernetes Clusterからのアプリケーション・ログの取得

アプリケーションのログを、Kubernetesクラスタ内のポッドから取得できます。

1. コマンド・ライン・ウィンドウを開きます。
2. アプリケーションのポッドを探します。 `<app-name>` プレースホルダーを置換します。

```
kubectl get pods | grep "<app-name>"
```

例:

```
$ kubectl get pods | grep "webapp01"
webapp01-deployment-84b7b6b5d4-5lnhb 1/1 Running 0 10d
webapp01-deployment-84b7b6b5d4-qxfbs 1/1 Running 0 10d
```

3. アプリケーションのログをポッドからフェッチします。 `<pod-name>` プレースホルダーを前のステップのポッド名に置き換えます。

```
kubectl logs --tail=100 <pod-name>
```

例:

```
$ kubectl logs --tail=100 webapp01-deployment-84b7b6b5d4-5lnhb
Not a secure app, removing idcs.jsp
Jan 21, 2019 9:16:53 AM org.apache.coyote.AbstractProtocol init
INFO: Initializing ProtocolHandler ["http-bio-8080"]
Jan 21, 2019 9:16:53 AM org.apache.catalina.core.StandardService
```

```
startInternal
INFO: Starting service Tomcat
Jan 21, 2019 9:16:53 AM org.apache.catalina.core.StandardEngine
startInternal
INFO: Starting Servlet Engine: Apache Tomcat/7.0.50
Jan 21, 2019 9:16:53 AM org.apache.catalina.startup.ContextConfig
getDefaultWebXmlFragment
INFO: No global web.xml found
Jan 21, 2019 9:16:53 AM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-bio-8080"]
Jan 21, 2019 6:56:47 PM org.apache.coyote.http11.AbstractHttp11Processor
process
INFO: Error parsing HTTP request header
Note: further occurrences of HTTP header parsing errors will be logged at
DEBUG level.
```

移行問題のトラブルシューティング

Oracle Cloud InfrastructureへのOracle Application Container Cloud Serviceアプリケーションの移行で問題が発生した場合は、移行時にトラブルシューティングを実行できます。

Kubernetesリソースのステータスの確認

アプリケーションのデプロイメントが失敗した場合、デプロイメント、ポッドおよびサービスの詳細をフェッチできます。

1. Kubernetesクラスタ内のデプロイメントをフェッチします。

```
kubectl get deployments
```

例:

```
$ kubectl get deployments
NAME                                DESIRED    CURRENT    UP-TO-DATE    AVAILABLE    AGE
webapp01-deployment                2          2          2             2            16h
webapp02-deployment                1          1          1             1            2d
javaeeapp01-deployment             1          1          1             1            1d
javaeeapp02-deployment             1          1          1             0            1d
```

DESIRED列およびAVAILABLE列の下値は、1以上にする必要があります。値が0の場合は、デプロイメントに問題があります。

2. Kubernetesクラスタ内のサービスをフェッチします。

```
kubectl get services
```

例:

```
$ kubectl get services
NAME                                TYPE                CLUSTER-IP      EXTERNAL-IP
PORT(S)          AGE
webapp01-service  LoadBalancer       10.x.x.x        132.x.x.x
80:43678/TCP     16h
webapp02-service  LoadBalancer       10.x.x.x        132.x.x.x
443:32628/TCP    2d
javaeeapp01-service LoadBalancer       10.96.112.230   10.x.x.x
443:30824/TCP    1d
javaeeapp02-service LoadBalancer       10.96.142.94    10.x.x.x
443:30830/TCP    1d
```

TYPE列の下値は、アプリケーションに従ってClusterIPまたはLoadBalancerになります。EXTERNAL-IPの下値は有効なIPアドレスにする必要があります。異なる値がある場合、サービスで問題が発生しています。

3. Kubernetesクラスタ内のpodsをフェッチします。

```
kubectl get pods
```

例:

```
$ kubectl get pods
NAME                                READY   STATUS
RESTARTS   AGE
javaeeapp01-deployment-67fb54b6c4-bdsfh  1/1     Running   0
1d
javaeeapp02-deployment-79f8f9bff9-42dqk  1/1     Running   0
1d
webapp01-deployment-7977d9f97b-4d4s9      0/1     Failed    0
16h
webapp01-deployment-7977d9f97b-7bwqf     1/1     Running   0
16h
```

STATUS列の下値は、Runningになります。値がFAILEDの場合、問題があります。

- 失敗したリソースに関する詳細をフェッチするには、次のコマンドを実行します。 `<resource-type>` および `<resource-id-or-name>` プレースホルダーに適切な値を指定します。

```
kubectl describe <resource-type> <resource-id-or-name>
```

例:

```
$ kubectl describe deployment webapp01-deployment
Name:                webapp01-deployment
Namespace:           default
CreationTimestamp:    Thu, 07 Feb 2019 17:20:03 +0530
...
Pod Template:
Labels:  app=webapp01-selector
Containers:
webapp01:
  Image:      fra.ocir.io/tenancy1/accs/oci_user1/webapp01:latest
  Port:       8080/TCP
  Host Port:  0/TCP
...
Conditions:
Type          Status  Reason
----          -
Available     False   MinimumReplicasUnavailable
Progressing    False   ProgressDeadlineExceeded
OldReplicaSets:  <none>
NewReplicaSet:   webapp01-deployment-7977d9f97b (2/2 replicas created)
Events:          <none>
```

移行中に作成されたリソースの削除

移行中に作成されたリソースを削除すると、移行が失敗した場合の迅速なリカバリが可能になります。

1. Kubernetesサービスを削除します。 `<app-name>`に適切な値を指定

```
kubectl delete service <app-name>-service
```

例:

```
$ kubectl delete service webapp01-service
service "webapp01-service" deleted
```

2. Kubernetesデプロイメントを削除します。 `<app-name>`に適切な値を指定

```
kubectl delete deployment <app-name>-deployment
```

例:

```
$ kubectl delete deployment webapp01-deployment
service "webapp01-deployment" deleted
```

3. TLSキーおよび証明書のKubernetesシークレットを削除します。 `<app-name>`に適切な値を指定

```
kubectl delete secret <app-name>-tls-certificate
```

例:

```
$ kubectl delete secret webapp01-tls-certificate  
secret "webapp01-tls-certificate" deleted
```

4. Kubernetes dockerシークレットを削除します。 `<app-name>`に適切な値を指定

```
kubectl delete secret <app-name>-secret
```

例:

```
$ kubectl delete secret webapp01-secret  
secret "webapp01-secret" deleted
```

5. 環境変数のKubernetes構成マップを削除します。 `<app-name>`に適切な値を指定

```
kubectl delete configmap <app-name>-config-var-map
```

例:

```
$ kubectl delete configmap webapp01-config-var-map  
configmap "webapp01-config-var-map" deleted
```

Kubernetesリソースを削除した後は、移行プロセスを再開できます。

4. 移行後のタスクの完了

Oracle Cloud Infrastructure ClassicからOracle Cloud InfrastructureへOracle Application Container Cloud Serviceアプリケーションを正常に移行した後、アプリケーションを完全にテストし、クリーンアップなどのオプションの構成タスクを実行します。

トピックス:

- 移行されたアプリケーションのテスト
- Oracle Cloud Infrastructure ClassicでのInfrastructure and Platformリソースのクリーンアップ

移行されたアプリケーションのテスト

アプリケーションをテストするには、アプリケーションのロード・バランサの外部IPアドレスまたはカスタムURL(構成されている場合)を使用します。 コマンドライン・ツールまたはwebブラウザを使用して、アプリケーションとの間でデータを転送できます。

コマンドライン・ツールの使用

コマンド行ツール(cURLやGNU Wgetなど)を使用すると、アプリケーションのエンドポイントにリクエストを送信してレスポンスを取得できます。

例4-1 cURLコマンド

```
[root@accs-migration-poc-vm accs2oke]# curl -k
http://notice.taosheng.tk/Notice/NoticeRest/getAll
[{"today":"2018-02-14"}]
```

例4-2 GNU Wgetコマンド

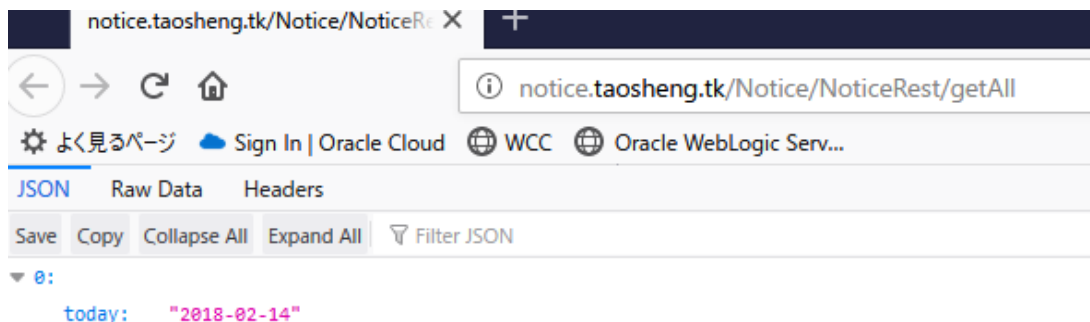
```
[root@accs-migration-poc-vm accs2oke]# wget
http://notice.taosheng.tk/Notice/NoticeRest/getAll -o response.json
```

webブラウザの使用

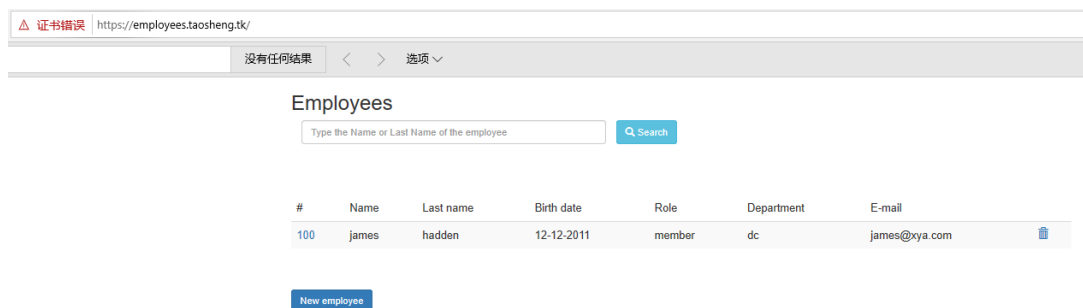
アプリケーションにユーザー・インタフェースが含まれている場合は、webブラウザを使用してアプリケーションと対話できます。 アプリケーションがユーザー・インタフェースのないREST Serviceである場合でも、webブラウザを使用してGETエンドポイントをテストできます。

アプリケーションのSSLエンドポイントを有効化した場合は、webブラウザを使用してSSL証明書を表示します。 SSL証明書が、Kubernetesデプロイメント構成で指定したものと同一であることを確認してください。

- JAVA



- JavaEE



Oracle Cloud Infrastructure ClassicでのInfrastructure and Platformリソースのクリーンアップ

Oracle Cloud Infrastructureで移行したアプリケーションをテストした後、Oracle Cloud Infrastructure Classicでアプリケーションおよびサポートするクラウド・リソースを削除できます。

使用しなくなったサービスのコストを避けるには、これらのOracle Cloud Infrastructure Classicリソースを削除します。

1. Oracle Application Container Cloud Serviceコンソールにアクセスします。
2. 「アプリケーション」表で、削除するアプリケーションの横にあるメニュー・アイコンをクリックします。
3. 「削除」をクリックします
4. 移行したOracle Application Container Cloud Serviceアプリケーションが使用しなくなったOracle Cloud Infrastructure Classic内の他のサービス・インスタンスをすべて削除します。