OpenShift Container Platform 4.16

スタートガイド

OpenShift Container Platform のスタートガイド

Last Updated: 2024-07-02

OpenShift Container Platform 4.16 スタートガイド

OpenShift Container Platform のスタートガイド

法律上の通知

概要

本書では、OpenShift Container Platform を使い始めるのに役立つ情報を提供します。これには、Kubernetes および OpenShift Container Platform で使用される一般的な用語の定義が含まれます。本書には、OpenShift Container Platform Web コンソールに関する手順の説明と、コマンドラインインターフェイスを使用したアプリケーションの作成およびビルドも含まれます。

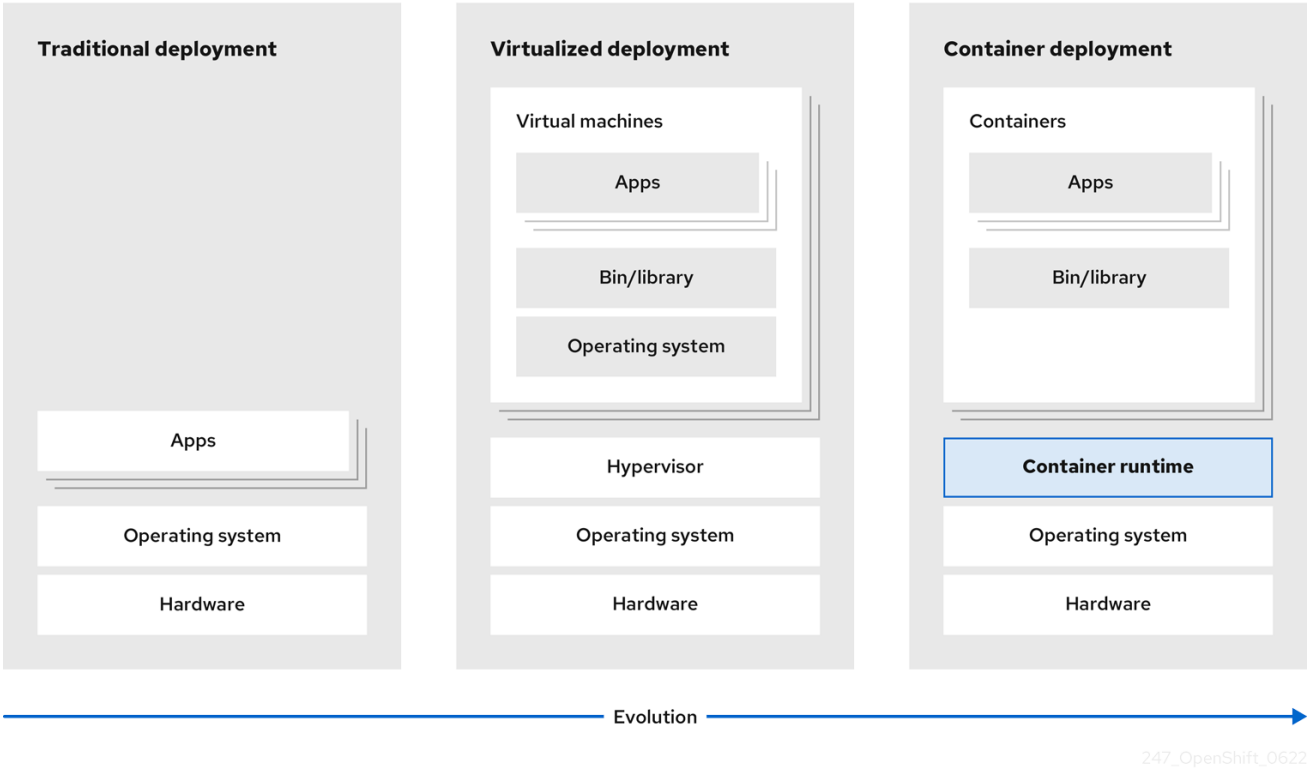
目次

第1章 KUBERNETES の概要

Kubernetes は、Google が開発したオープンソースのコンテナーオーケストレーションツールです。Kubernetes を使用して、コンテナーベースのワークロードを実行および管理できます。最も一般的な Kubernetes のユースケースは、相互接続されたマイクロサービスのアレイをデプロイし、クラウドネイティブな方法でアプリケーションを構築することです。オンプレミス、パブリック、プライベート、またはハイブリッドクラウド全体のホストにまたがることができる Kubernetes クラスターを作成できます。

従来、アプリケーションは単一のオペレーティングシステムの上にデプロイされていました。仮想化を使用すると、物理ホストを複数の仮想ホストに分割できます。共有リソース上の仮想インスタンスで作業することは、効率やスケーラビリティの面で最適とは言えません。仮想マシン (VM) は物理マシンと同じ数のリソースを消費するため、CPU、RAM、ストレージなどのリソースを VM に提供するとコストがかかる可能性があります。また、共有リソースでの仮想インスタンスの使用により、アプリケーションのパフォーマンスが低下する場合があります。

図1.1 従来のデプロイメント向けのコンテナーテクノロジーの進化



この問題を解決するには、コンテナー化された環境でアプリケーションを分離するコンテナー化テクノロジーを使用することができます。VM と同様に、コンテナーには独自のファイルシステム、vCPU、メモリー、プロセススペース、依存関係などがあります。コンテナーは基盤となるインフラストラクチャーから切り離されており、クラウドや OS ディストリビューション間で移植可能です。コンテナーは本来、全機能を備えた OS よりもはるかに軽量で、オペレーティングシステムカーネルで実行される軽量の分離されたプロセスです。仮想マシンは、(コンテナーよりも) 起動に時間がかかり、物理ハードウェアを抽象化したものです。VM は、ハイパーバイザーを使用して単一のマシンで実行されます。

Kubernetes を利用すると、以下のようなアクションを行うことができます。

- リソースの共有
- 複数のホストにまたがるコンテナーのオーケストレーション
- 新しいハードウェア設定のインストール

- ヘルスチェックと自己修復アプリケーションの実行
- コンテナー化されたアプリケーションのスケーリング

1.1. KUBERNETES コンポーネント

表1.1 Kubernetes コンポーネント

コンポーネント	目的
kube-proxy	クラスター内のすべてのノードで実行され、Kubernetes リソース間のネットワークトラフィックを維持します。
kube-controller-manager	クラスターの状態を管理します。
kube-scheduler	Pod をノードに割り当てます。
etcd	クラスターデータを保存します。
kube-apiserver	API オブジェクトのデータを検証および設定します。
kubelet	ノード上で実行され、コンテナーマニフェストを読み取ります。定義されたコンテナーが開始され、実行されていることを確認します。
kubectl	ワークロードの実行方法を定義できるようにします。**kubectl** コマンドを使用して、**kube-apiserver** と対話します。
Node	ノードは、Kubernetes クラスター内の物理マシンまたは VM です。コントロールプレーンはすべてのノードを管理し、Kubernetes クラスター内のノード全体で Pod をスケジュールします。
コンテナーランタイム	コンテナーランタイムは、ホストオペレーティングシステムでコンテナーを実行します。Pod をノードで実行できるように、各ノードにコンテナーランタイムをインストールする必要があります。
永続ストレージ	デバイスがシャットダウンされた後でもデータを保存します。Kubernetes は永続ボリュームを使用して、アプリケーションデータを保存します。
container-registry	コンテナーイメージを保存してアクセスします。
Pod	Pod は、Kubernetes における最小の論理単位です。Pod には、ワーカーノードで実行する 1 つ以上のコンテナーが含まれています。

1.2. KUBERNETES リソース

カスタムリソースは、KubernetesAPI のエクステンションです。カスタムリソースを使用して、Kubernetes クラスターをカスタマイズできます。Operator は、カスタムリソースを使用してアプリケーションとそのコンポーネントを管理するソフトウェアエクステンションです。Kubernetes は、クラスターリソースの処理中に一定の望ましい結果を得る必要がある場合に、宣言型モデルを使用しま

す。Operator を使用することで、Kubernetes は宣言的な方法で状態を定義します。命令型コマンドを使用して、Kubernetes クラスターリソースを変更できます。Operator は、リーソスの目的とされる状態と、実際の状態を継続的に比較して、実際の状態を目的の状態と合わせられるようにアクションを実行します。

図1.2 Kubernetes クラスターの概要

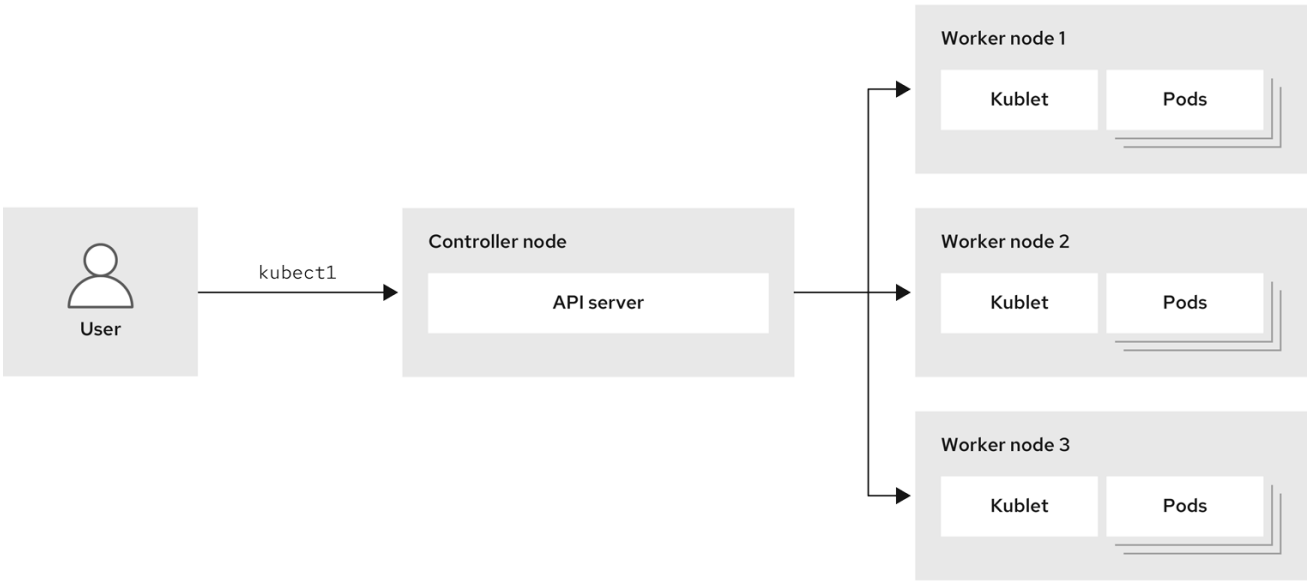
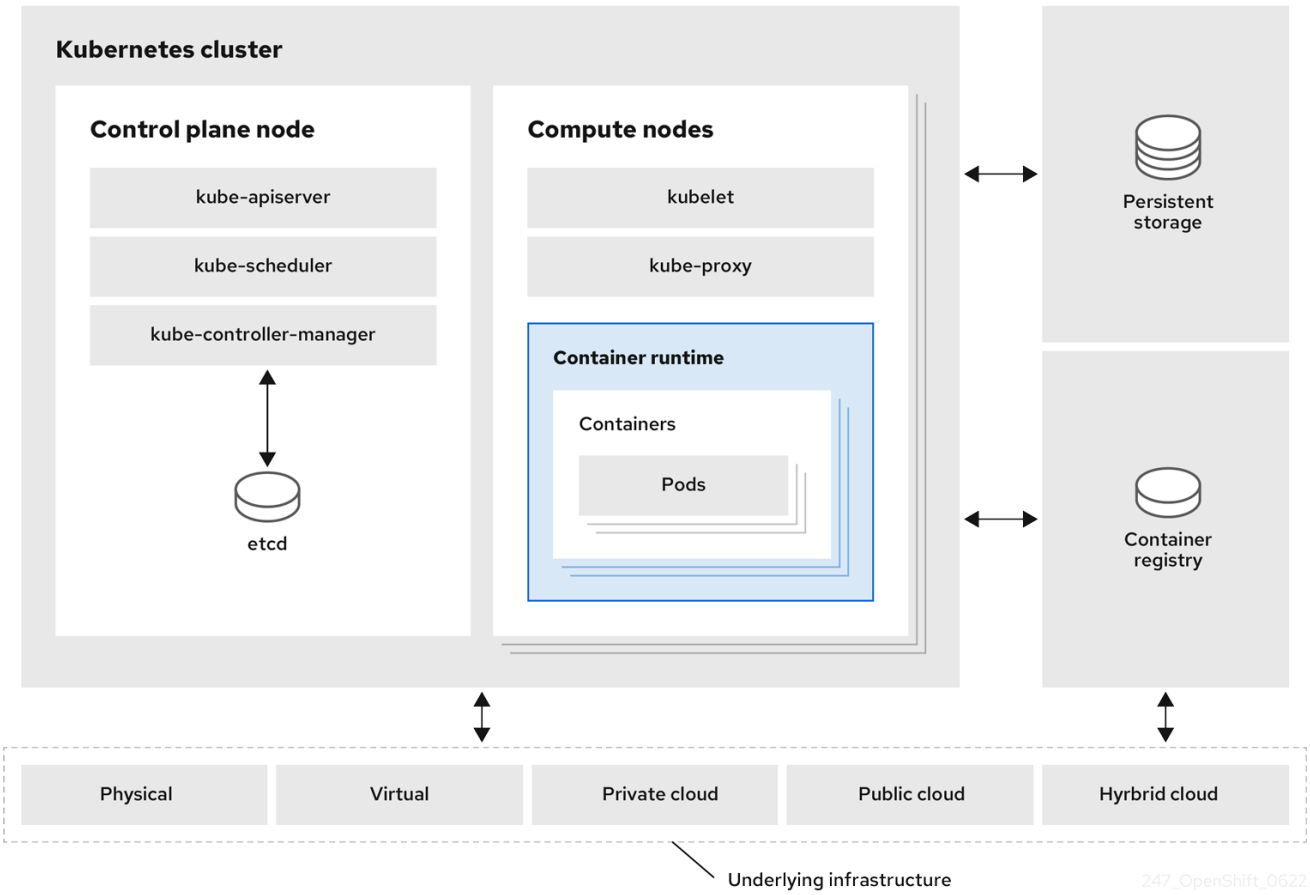


表1.2 Kubernetes リソース

リソース	目的
Service	Kubernetes はサービスを使用して、実行中のアプリケーションを一連の Pod に公開します。
ReplicaSet	Kubernetes は **ReplicaSets** を使用して、一定の Pod 数を維持します。
Deployment	アプリケーションのライフサイクルを維持するリソースオブジェクト。

Kubernetes は、OpenShift Container Platform のコアコンポーネントです。OpenShift Container Platform は、コンテナー化されたアプリケーションの開発および実行に使用できます。Kubernetes を基盤とする OpenShift Container Platform には、大規模な通信、ビデオストリーミング、ゲーム、銀行取引、およびその他のアプリケーションのエンジンと同様のサービスを提供する技術が組み込まれています。OpenShift Container Platform を使用することで、コンテナー化されたアプリケーションを単一のクラウドを超えてオンプレミスおよびマルチクラウド環境へと拡張することができます。

図1.3 Kubernetes のアーキテクチャー



クラスターは、クラウド環境内の複数のノードで設定される単一の計算ユニットです。Kubernetes クラスターには、コントロールプレーンとワーカーノードが含まれます。さまざまなマシンや環境で Kubernetes コンテナーを実行できます。コントロールプレーンノードは、クラスターの状態を制御および維持します。ワーカーノードを使用して Kubernetes アプリケーションを実行できます。Kubernetes の namespace を使用して、クラスター内のクラスターリソースを区別できます。namespace のスコープは、デプロイメント、サービス、Pod などのリソースオブジェクトに適用できます。ストレージクラス、ノード、永続ボリュームなどのクラスター全体のリソースオブジェクトに、namespace を使用することはできません。

1.3. KUBERNETES の概念ガイドライン

OpenShift Container Platform を使い始める前に、以下に示す、概念面での Kubernetes のガイドラインを考慮してください。

- 1 つまたは複数のワーカーノードを使用することからスタートし、コンテナーのワークロードを実行します。
- 1 つまたは複数のコントロールプレーンノードからワークロードのデプロイを管理します。
- Pod と呼ばれるデプロイメント単位にコンテナーをラップします。Pod を使うことでコンテナーに追加のメタデータが付与され、複数のコンテナーを単一のデプロイメントエンティティーにグループ化する機能が提供されます。
- 特殊な種類のアセットを作成します。たとえば、サービスは一連の Pod とそのアクセス方法を定義するポリシーによって表されます。このポリシーにより、コンテナーはサービス用の特定の IP アドレスを持っていない場合でも、必要とするサービスに接続することができます。レプ

リケーションコントローラーは、一度に実行するのに必要な Pod レプリカ数を示すもう一つの特殊なアセットです。この機能を使うと、現在の需要に対応できるようにアプリケーションを自動的にスケーリングすることができます。

OpenShift Container Platform クラスターへの API は 100% Kubernetes です。他の Kubernetes で実行されているコンテナーと OpenShift Container Platform で実行されているコンテナーの間では、何も変更はありません。アプリケーションに変更はありません。OpenShift Container Platform は、Kubernetes にエンタープライズ対応の拡張機能を提供する付加価値機能をもたらします。OpenShift Container Platform CLI ツール (**oc**) は **kubectl** と互換性があります。Kubernetes API は OpenShift Container Platform 内で 100% アクセス可能ですが、**kubectl** コマンドラインには使いやすい機能があまりありません。OpenShift Container Platform は、**oc** のような一連の機能とコマンドラインツールを提供します。Kubernetes はアプリケーションの管理には優れていますが、プラットフォームレベルの要件やデプロイメントプロセスの指定や管理には対応しません。OpenShift Container Platform が提供する強力かつ柔軟なプラットフォーム管理ツールとプロセスは、重要な利点となります。コンテナー化プラットフォームに、認証、ネットワーキング、セキュリティー、監視、およびログ管理を追加する必要があります。

第2章 OPENSHIFT CONTAINER PLATFORM の概要

OpenShift Container Platform は、クラウドベースの Kubernetes コンテナープラットフォームです。OpenShift Container Platform の基盤は、Kubernetes に基づいているため、同じテクノロジーを共有しています。アプリケーションおよびアプリケーションをサポートするデータセンターで、わずか数台のマシンとアプリケーションから、何百万ものクライアントに対応する何千ものマシンに拡張できるように設計されています。

OpenShift Container Platform を使用すると、以下を実行できます。

- 開発者および IT 組織に、セキュアでスケーラブルなリソースへのアプリケーションのデプロイに使用できる、クラウドアプリケーションプラットフォームを提供する。
- 設定および管理のオーバーヘッドを最小限に抑える。
- Kubernetes プラットフォームをお客様のデータセンターおよびクラウドに導入する。
- セキュリティー、プライバシー、コンプライアンス、ガバナンスの要件を満たす。

Kubernetes を基盤とする OpenShift Container Platform には、大規模な通信、ビデオストリーミング、ゲーム、銀行取引、およびその他のアプリケーションのエンジンと同様にサービスを提供する技術が組み込まれています。Red Hat のオープンテクノロジーに実装することで、コンテナー化されたアプリケーションを、単一クラウドを超えてオンプレミスおよびマルチクラウド環境へと拡張することが可能です。

2.1. OPENSHIFT CONTAINER PLATFORM の共通用語集

この用語集では、一般的な Kubernetes および OpenShift Container Platform の用語を定義します。

Kubernetes

Kubernetes は、コンテナー化されたアプリケーションのデプロイ、スケーリング、管理を自動化するための、オープンソースのコンテナーオーケストレーションエンジンです。

コンテナー

コンテナーは、ワーカーノードの OCI 準拠のコンテナーで実行されるアプリケーションインスタンスおよびコンポーネントです。コンテナーは、Open Container Initiative (OCI) 準拠のイメージのランタイムです。イメージはバイナリーアプリケーションです。ワーカーノードは、多数のコンテナーを実行できます。ノードの能力は、ベースとなるリソースがクラウド、ハードウェア、または仮想化のいずれであっても、そのリソースのメモリーおよび CPU の機能に関連します。

Pod

Pod は、1 つのホスト上に共にデプロイされる 1 つまたは複数のコンテナーです。一緒に配置されたコンテナーのグループおよびボリュームや IP アドレスなどの共有リソースで設定されます。Pod は、定義、デプロイ、および管理される最小のコンピュート単位でもあります。
OpenShift Container Platform では、Pod はデプロイ可能な最小単位として個別のアプリケーションコンテナーに代わるものです。

Pod は、OpenShift Container Platform ではオーケストレーションされた単位です。OpenShift Container Platform は、同じノード上の Pod のすべてのコンテナーをスケジュールし、実行します。複雑なアプリケーションは数多くの Pod で設定され、それぞれが独自のコンテナーを持ちます。これらは外部と連携し、また OpenShift Container Platform 環境内で相互に連携します。

レプリカセットおよびレプリケーションコントローラー

Kubernetes のレプリカセットと OpenShift Container Platform のレプリケーションコントローラーの両方が利用可能です。このコンポーネントの役割は、指定された数の Pod レプリカが常時実行さ

れるようにすることです。Pod が終了または削除されると、レプリカセットまたはレプリケーションコントローラーが別の Pod を起動します。必要以上の Pod が実行されている場合、指定されたレプリカ数となるように、レプリカセットは必要な数だけ削除します。

Deployment および DeploymentConfig

OpenShift Container Platform は、Kubernetes の **Deployment** オブジェクトと OpenShift Container Platform の **DeploymentConfigs** オブジェクトの両方を実装します。ユーザーはどちらかを選択できます。

Deployment オブジェクトは、アプリケーションを Pod としてロールアウトする方法を制御します。これらは、レジストリーから取得してノード上に Pod としてデプロイするコンテナーイメージの名前を特定します。デプロイする Pod のレプリカ数を設定し、プロセスを管理するレプリカセットを作成します。示されるラベルは、Pod をデプロイするノードをスケジューラーに指示します。ラベルのセットは、レプリカセットがインスタンス化する Pod 定義に含まれます。

Deployment オブジェクトは、**Deployment** オブジェクトのバージョンと、許容されるアプリケーションの可用性を管理するための各種のロールアウトストラテジーに基づいて、ワーカーノードにデプロイされる Pod を更新できます。OpenShift Container Platform の **DeploymentConfig** オブジェクトにはさらに変更トリガーの機能が追加され、新しいバージョンのコンテナーイメージが利用可能になったり、その他の変更が生じたりすると、新しいバージョンの **Deployment** オブジェクトを自動的に作成できます。

サービス

サービスは、Pod の論理セットとアクセスポリシーを定義します。Pod が作成および破棄されるときに使用する他のアプリケーションの永続的な内部 IP アドレスおよびホスト名を提供します。

サービス層は、アプリケーションコンポーネント同士を結合します。たとえば、フロントエンドの Web サービスは、そのサービスと通信してデータベースインスタンスに接続します。サービスを使用すると、アプリケーションコンポーネント間の単純な内部負荷分散が可能になります。OpenShift Container Platform は、検出を容易にするために、サービス情報を実行中のコンテナーに自動的に挿入します。

ルート

ルートは、www.example.com などの外部から到達可能なホスト名を指定して、サービスを公開する手段です。各ルートは、ルート名、サービスセレクター、セキュリティー設定 (任意) で設定されます。ルーターは、定義されたルートやそのサービスによって識別されるエンドポイントを使用して、外部クライアントがアプリケーションへの到達に使用できる名前を指定できます。完全なマルチレイヤーのアプリケーションをデプロイすることは簡単ですが、OpenShift Container Platform 環境外からのトラフィックは、ルーティングレイヤーがないとアプリケーションに到達できません。

ビルド

ビルドとは、入力パラメーターを結果として作成されるオブジェクトに変換するプロセスです。ほとんどの場合、このプロセスは入力パラメーターまたはソースコードを実行可能なイメージに変換するために使用されます。**BuildConfig** オブジェクトはビルドプロセス全体の定義です。OpenShift Container Platform は、ビルドイメージからコンテナーを作成し、それらを統合レジストリーにプッシュすることで、Kubernetes を活用します。

プロジェクト

OpenShift Container Platform はプロジェクトを使用して、ユーザーや開発者のグループが連携できるようにし、分離およびコラボレーションの単位として機能します。リソースのスコープを定義し、プロジェクト管理者およびコラボレーターがリソースを管理できるようにし、クォータや制限でユーザーのリソースを制限し、追跡します。

プロジェクトは、追加のアノテーションを備えた Kubernetes namespace です。これは、通常ユーザーのリソースへのアクセスを管理する中心的な手段です。プロジェクトにより、ユーザーコミュニティーは、他のコミュニティーと切り離された状態で独自のコンテンツを整理し、管理することができます。ユーザーは、管理者からプロジェクトへのアクセス権限を受け取る必要があります。ただし、クラスター管理者は、開発者が専用のプロジェクトを作成するのを許可できます。この場合、ユーザーは自動的に専用のプロジェクトにアクセスできます。

各プロジェクトには、オブジェクト、ポリシー、制約、およびサービスアカウントの独自のセットがあります。

プロジェクトは namespace としても知られています。

Operator

Operator は Kubernetes ネイティブアプリケーションです。Operator の目的は、運用上の知識をソフトウェアに配置することです。従来、この知識は管理者の頭の中、シェルスクリプトのさまざまな組み合わせ、または Ansible などの自動化ソフトウェアにのみ存在していました。これは Kubernetes クラスター外にあり、統合するのが困難でした。Operator により、これらのすべてが変わります。
Operator は、ご自分のアプリケーションに合わせて構築されます。また、Kubernetes の概念や API とネイティブに統合することで、一般的な Day 1 作業 (インストールおよび設定) ならびに Day 2 作業 (スケールアップ/ダウン、設定の変更、更新、バックアップ、フェイルオーバー、および復元) を、Kubernetes クラスター内で実行されるソフトウェアの一部で実装し自動化します。これは Kubernetes ネイティブアプリケーションと呼ばれます。

Operator では、アプリケーションを Pod、デプロイメント、サービス、または設定マップなどの基本要素のコレクションとして扱わないでください。代わりに、Operator は、アプリケーションにとって意味のあるオプションを公開する単一のオブジェクトとして扱う必要があります。

2.2. OPENSHIFT CONTAINER PLATFORM について

OpenShift Container Platform は、コンテナーベースのアプリケーションのライフサイクルと、ベアメタル、仮想化、オンプレミス、クラウドなどのさまざまなコンピューティングプラットフォームへの依存関係を管理するための Kubernetes 環境です。OpenShift Container Platform はコンテナーをデプロイ、設定、および管理します。OpenShift Container Platform は、そのコンポーネントの使いやすさ、安定性、およびカスタマイズを提供します。

OpenShift Container Platform は、ノードと呼ばれる多数のコンピューティングリソースを利用します。ノードには、Red Hat Enterprise Linux CoreOS (RHCOS) として知られる、Red Hat Enterprise Linux (RHEL) に基づく軽量で安全なオペレーティングシステムがあります。

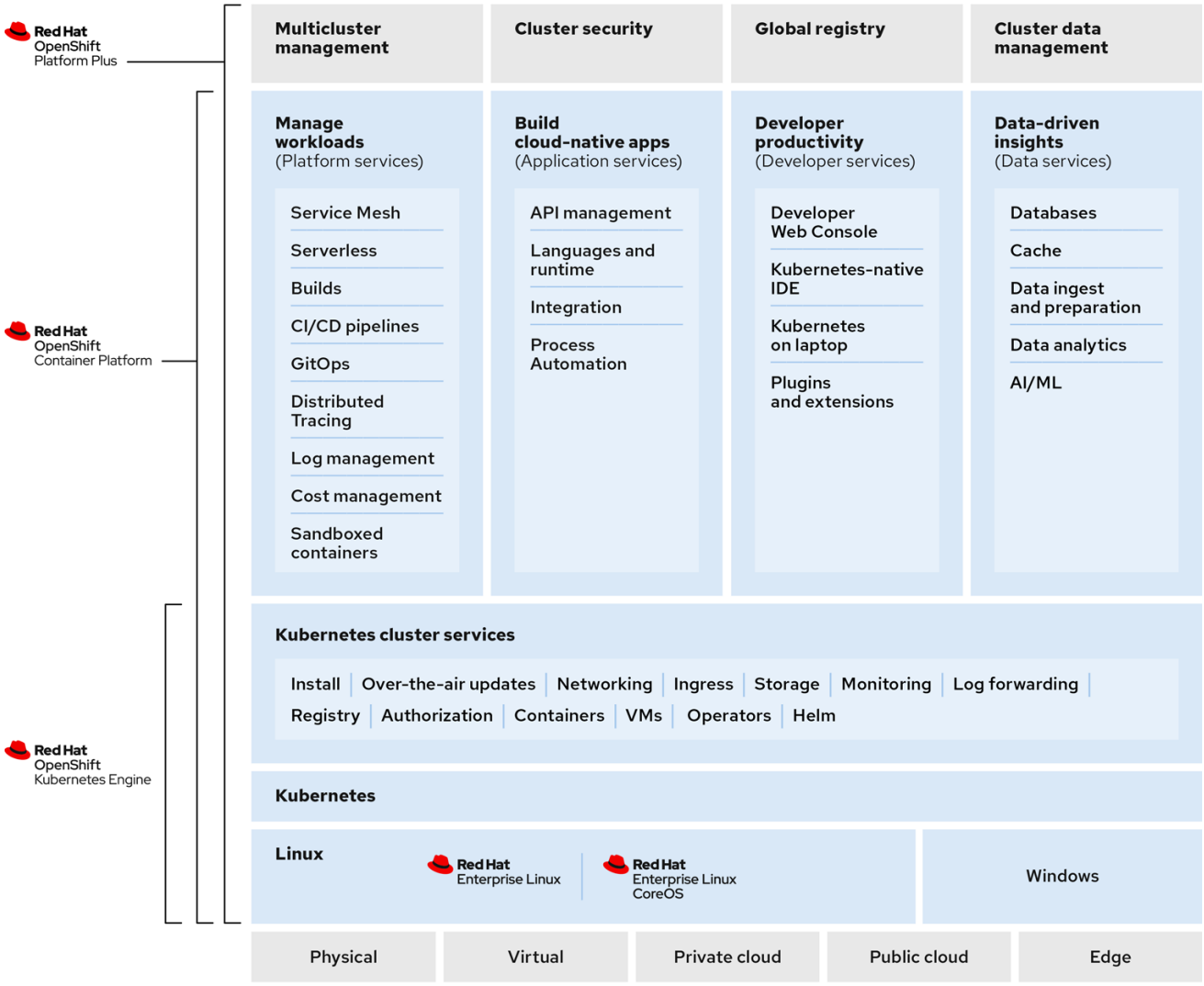
ノードは、起動して設定した後、スケジュールされたコンテナーワークロードのイメージを管理および実行するために、CRI-O や Docker などのコンテナーランタイムを取得します。Kubernetes エージェント (kubelet) は、ノード上のコンテナーワークロードをスケジュールします。kubelet は、ノードをクラスターに登録し、コンテナーワークロードの詳細を受け取ります。

OpenShift Container Platform は、クラスターのネットワーキング、負荷分散、およびルーティングを設定および管理します。OpenShift Container Platform は、クラスターのヘルスとパフォーマンスのモニタリング、ロギング、およびアップグレードの管理のためのクラスターサービスを追加します。

コンテナーイメージレジストリーと OperatorHub は、クラスター内でさまざまなアプリケーションサービスを提供するための Red Hat 認定製品とコミュニティービルドソフトウェアを提供します。これらのアプリケーションとサービスは、クラスターにデプロイされたアプリケーション、データベース、フロントエンドとユーザーインターフェイス、アプリケーションランタイムとビジネスの自動化、およびコンテナーアプリケーションの開発とテストのための開発者サービスを管理します。

クラスター内のアプリケーションを管理するには、ビルド済みのイメージから実行されるコンテナーのデプロイを手動で設定するか、Operator と呼ばれるリソースを使用します。ビルド前のイメージとソースコードからカスタムイメージをビルドし、これらのカスタムイメージをローカルの内部、プライベート、またはパブリックレジストリーに保存できます。

マルチクラスター管理レイヤーは、単一のコンソールで、複数のクラスターのデプロイメント、設定、コンプライアンス、およびワークロードの分散を管理できます。



2.3. OPENSHIFT CONTAINER PLATFORM のインストール

OpenShift Container Platform インストールプログラムは柔軟性を提供します。インストールプログラムを使用して、インストールプログラムがプロビジョニングし、クラスターで維持するインフラストラクチャーでクラスターをデプロイしたり、ユーザーが独自に準備し、維持するインフラストラクチャーでクラスターをデプロイしたりすることができます。

インストールプロセス、サポートされるプラットフォーム、ならびにクラスターをインストールおよび準備する方法の選択に関する詳細は、以下を参照してください。

- OpenShift Container Platform インストールの概要
- インストールプロセス
- OpenShift Container Platform クラスターでサポートされるプラットフォーム
- クラスターのインストールタイプの選択

2.3.1. OpenShift Local の概要

OpenShift Local は、OpenShift Container Platform クラスターのビルドを開始するための迅速なアプリケーション開発をサポートします。OpenShift Local は、ローカルのコンピューターで実行し、セットアップおよびテストをシンプル化し、コンテナーベースのアプリケーションを開発するのに必要なすべてのツールと共にクラウド開発環境をローカルにエミュレートすることを目的として設計されています。

OpenShift Local は、使用するプログラミング言語にかかわらずアプリケーションをホストし、事前に設定された最小限の Red Hat OpenShift Container Platform クラスターをローカル PC に提供します。その際に、サーバーベースのインフラストラクチャーは必要ありません。

ホストされる環境では、OpenShift Local はマイクロサービスを作成してイメージに変換し、Linux、macOS、または Windows 10 以降を実行するノートパソコンまたはデスクトップ上の Kubernetes がホストするコンテナーで直接それらを実行できます。

OpenShift Local の詳細は、Red Hat OpenShift Local の概要 を参照してください。

2.4. 次のステップ

2.4.1. 開発者の場合

OpenShift Container Platform を使用して、コンテナー化されたアプリケーションを開発し、デプロイします。OpenShift Container Platform は、コンテナー化されたアプリケーションを開発し、デプロイするためのプラットフォームです。OpenShift Container Platform のドキュメントは、次の点で活用できます。

- **OpenShift Container Platform での開発** を理解する：単純なコンテナーから高度な Kubernetes デプロイメントおよび Operator に至るまで、コンテナー化された各種アプリケーションについて説明します。
- **プロジェクトを使用** する：OpenShift Container Platform Web コンソールまたは OpenShift CLI (**oc**)からプロジェクトを作成し、開発するソフトウェアを整理し、共有します。
- **アプリケーションを操作する**:

OpenShift Container Platform Web コンソール の **Developer** パースペクティブ を使用して、アプリケーションを作成およびデプロイ します。

Topology ビュー を使用して、アプリケーションの表示、ステータスの監視、コンポーネントの接続およびグループ化、ならびにコードベースの変更を行います。

- **開発者 CLI ツール(odo)を使用** する： **odo** CLI ツールにより、開発者は単一コンポーネントまたはマルチコンポーネントのアプリケーションを作成でき、デプロイメント、ビルド、およびサービスルート設定を自動化できます。odo は複雑な Kubernetes および OpenShift Container Platform の概念を抽象化し、アプリケーションの開発に集中できるようにします。
- **CI/CD パイプラインを作成する**: パイプラインは、分離されたコンテナーで実行されるサーバーレス、クラウドネイティブ、継続的インテグレーション、および継続的デプロイメントシステムです。パイプラインは、標準の Tekton カスタムリソースを使用してデプロイメントを自動化し、マイクロサービスベースのアーキテクチャーで機能する分散型チーム向けに設計されています。
- **Helm チャートをデプロイ** する：Helm 3 は、開発者が Kubernetes でアプリケーションパッケージの定義、インストール、および更新を行うのに役立つパッケージマネージャーです。Helm チャートは、Helm CLI を使用してデプロイできるアプリケーションを記述するパッケージング形式です。

- **イメージビルド** を理解する：さまざまな種類のソースマテリアル(Git リポジトリー、ローカルバイナリー入力、および外部アーティファクト)が含まれる各種のビルドストラテジー(Docker、S2I、カスタム、およびパイプライン)から選択します。次に、基本的なビルドから高度なビルドまで、ビルドタイプの例に従います。

- **コンテナーイメージを作成** する：コンテナーイメージは、OpenShift Container Platform （および Kubernetes）アプリケーションで最も基本的なビルディングブロックです。イメージストリームを定義すると、開発の進捗に応じて、イメージの複数のバージョンを 1 つの場所に集約できます。S2I コンテナーを使用すると、Ruby、Node.js、Python などの特定タイプのコードを実行するように設定されたベースコンテナーに、ソースコードを挿入することができます。

- **デプロイメントを作成** する： **Deployment** および **DeploymentConfig** オブジェクトを使用して、アプリケーションの詳細な管理を行います。**Workloads** ページまたは OpenShift CLI (**oc**) を使用して、デプロイメントを管理 します。ローリング、再作成およびカスタム のデプロイメントストラテジーについて説明しています。

- **テンプレートを作成** する：既存のテンプレートを使用するか、アプリケーションのビルドまたはデプロイ方法を記述する独自のテンプレートを作成します。テンプレートは、イメージと説明、パラメーター、レプリカ、公開されたポートおよびアプリケーションの実行またはビルド方法を定義するその他のコンテンツを組み合わせることができます。

- **Operator につい** て理解する：Operator は、OpenShift Container Platform 4.16 で推奨される、クラスターアプリケーションの作成方法です。Operator Framework について、またインストールされた Operator を使用してアプリケーションをプロジェクトにデプロイする方法を説明します。

- **Operator を** 開発する：Operator は、OpenShift Container Platform 4.16 で推奨される、クラスターアプリケーションの作成方法です。Operator の構築、テスト、およびデプロイのワークフローを説明します。次に、Ansible または Helm をベースにして独自の Operator を作成するか、Operator SDK を使用して ビルトイン Prometheus モニタリング を設定します。

- **REST API リファレンス**: OpenShift Container Platform アプリケーションプログラミングインターフェイスのエンドポイントについて説明します。

2.4.2. 管理者の場合

- **OpenShift Container Platform の管理**: OpenShift Container Platform 4.16 コントロールプレーンのコンポーネントについて説明します。OpenShift Container Platform コントロールプレーンおよびワーカーノードが マシン API および Operator によりどのように管理および更新されるかを参照して ください。

- **ユーザーとグループを管理** する：クラスターの使用または変更について、さまざまなレベルのパーミッションを持つユーザーおよびグループを追加します。

- **認証を管理** する：OpenShift Container Platform で、ユーザー、グループ、および API 認証がどのように機能するかを確認します。OpenShift Container Platform は、複数のアイデンティティープロバイダーをサポートします。

- ネットワークの **管理**: OpenShift Container Platform のクラスターネットワークは、 Cluster Network Operator (CNO)によって管理されます。CNO は、 kube-proxy の iptables ルールを使用して、ノードとそれらのノードで実行されている Pod 間のトラフィックを転送します。Multus Container Network Interface は 複数のネットワークインターフェイス を Pod に割り当てる機能を追加します。ネットワークポリシー 機能を使用して、Pod を分離したり、選択したトラフィックを許可したりできます。

- **ストレージを管理** する：OpenShift Container Platform では、クラスター管理者は永続ストレージを設定できます。

- **Operator を管理** する：Red Hat、ISV、およびコミュニティー Operator の一覧は、クラスター管理者によって確認でき、クラスターにインストール できます。インストール後に、クラスターで Operator を 実行 したり、 アップグレード したり、バックアップしたり、Operator を管理できます。

- **カスタムリソース定義(CRD)を使用してクラスター を変更する**: Operator で実装されたクラスター機能は、CRD で変更できます。CRD の 作成 および CRD からの リソースの管理 について説明します。

- **リソースクォータを設定** する：CPU、メモリー、その他のシステムリソースから選択し、 クォータを設定 します。

- **リソースをプルーニングおよび回収** する：不要な Operator、グループ、デプロイメント、ビルド、イメージ、レジストリー、および cron ジョブをプルーニングして領域を回収します。

- クラスターの **スケーリング** および **チューニング** を行う：**クラスター** 制限の設定、ノードのチューニング、クラスターモニタリングのスケーリング、および環境に合わせてネットワーク、ストレージ、ルートの最適化を行います。

- 非接続環境 **で OpenShift Update Service を使用する：非接続**環境で OpenShift Container Platform の更新を推奨するために、ローカルの OpenShift Update Service のインストールおよび管理について確認します。

- **クラスターを監視** する： モニタリングスタックの設定 について確認します。モニタリングの設定後、Web コンソールを使用して モニタリングダッシュボード にアクセスします。インフラストラクチャーメトリクスに加え、独自サービスのメトリクスも収集して表示できます。

- **リモートヘルスモニタリング**: OpenShift Container Platform はクラスターについての匿名の集計情報を収集します。Telemetry および Insights Operator を使用すると、このデータは Red Hat によって受信され、OpenShift Container Platform を改善するために使用されます。リモートヘルスモニタリングで収集されるデータ を表示できます。

第3章 WEB コンソールを使用したアプリケーションの作成およびビルド

3.1. 作業を開始する前に

- Web コンソールへのアクセス を 確認します。
- 実行中の OpenShift Container Platform インスタンスにアクセスできる必要があります。アクセスできない場合は、クラスター管理者にお問い合わせください。

3.2. WEB コンソールへのログイン

OpenShift Container Platform Web コンソールにログインしてクラスターにアクセスし、これを管理できます。

前提条件

- OpenShift Container Platform クラスターへのアクセス。

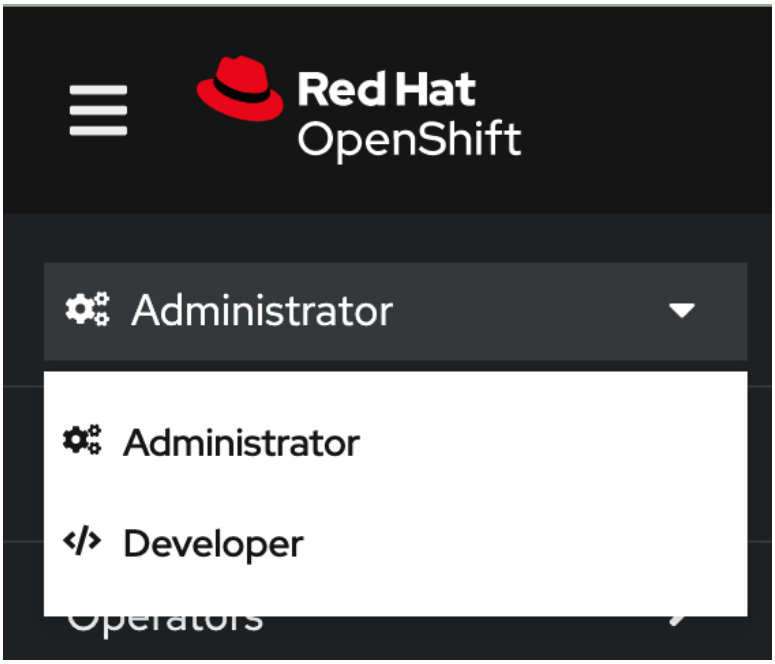
手順

- ログイン認証情報を使用して OpenShift Container Platform Web コンソールにログインします。

Projects ページにリダイレクトされます。管理者以外のユーザーの場合、デフォルトのビューは **Developer** パースペクティブです。クラスター管理者の場合、デフォルトのビューは **Administrator** パースペクティブです。**cluster-admin** 権限がない場合は、Web コンソールに **Administrator** パースペクティブは表示されません。

Web コンソールは、**Administrator** パースペクティブと **Developer** パースペクティブという 2 つのパースペクティブを提供します。**Developer** パースペクティブは、開発者のユースケースに固有のワークフローを提供します。

図3.1 パースペクティブスイッチャー



パースペクティブスイッチャーを使用して、**Developer** パースペクティブに切り替えます。**Topology** ビューがアプリケーションを作成するオプションと共に表示されます。

3.3. 新規プロジェクトの作成

プロジェクトにより、ユーザーコミュニティーは、切り離された状態で独自のコンテンツを整理し、管理することができます。プロジェクトは、Kubernetes namespace に対する OpenShift Container Platform 拡張です。プロジェクトには、ユーザーのセルフプロビジョニングを可能にする追加機能があります。

ユーザーは、管理者からプロジェクトへのアクセス権限を受け取る必要があります。クラスター管理者は、開発者が専用のプロジェクトを作成するのを許可できます。ほとんどの場合、ユーザーは自動的に専用のプロジェクトにアクセスできます。

各プロジェクトには、オブジェクト、ポリシー、制約、およびサービスアカウントの独自のセットがあります。

前提条件

- OpenShift Container Platform Web コンソールにログインしている。
- **Developer** パースペクティブを使用している。
- OpenShift Container Platform でアプリケーションおよび他のワークロードを作成するための適切なプロジェクト内のロールおよびパーミッションがある。

手順

1. **+Add** ビューで、**Project → Create Project** を選択します。
2. **Name** フィールドで、**user-getting-started** を入力します。
3. オプション:**Display name** フィールドに **Getting Started with OpenShift** と入力します。

注記

Display name および **Description** フィールドは任意です。

4. **Create** をクリックします。

初めてのプロジェクトを OpenShift Container Platform に作成しました。

関連情報

- デフォルトのクラスターロール
- Web コンソールを使用したプロジェクトの表示
- Developer パースペクティブを使用したプロジェクトに対するアクセスパーミッションの提供
- Web コンソールを使用したプロジェクトの削除

3.4. 表示パーミッションの付与

OpenShift Container Platform は、すべてのプロジェクトにいくつかの特別なサービスアカウントを自動的に作成します。デフォルトのサービスアカウントは、Pod を実行する役割を担います。OpenShift Container Platform は、このサービスアカウントを使用して、起動するすべての Pod に挿入します。

以下の手順では、デフォルトの **ServiceAccount** オブジェクトに **RoleBinding** オブジェクトを作成します。サービスアカウントは OpenShift Container Platform API と通信し、プロジェクト内の Pod、サービス、およびリソースについて把握します。

前提条件

- OpenShift Container Platform Web コンソールにログインしている。
- イメージがデプロイされている。
- **Administrator** パースペクティブに切り替えられている。

手順

1. **User Management** に移動し、**RoleBindings** をクリックします。
2. **Create binding** をクリックします。
3. **Namespace role binding (RoleBinding)** を選択します。
4. **Name** フィールドで、**sa-user-account** を入力します。
5. **Namespace** フィールドで、**user-getting-started** を検索して選択します。
6. **Role name** フィールドで **view** を検索して **view** を選択します。
7. **Subject** フィールドで **ServiceAccount** を選択します。
8. **Subject namespace** フィールドで、**user-getting-started** を検索して選択します。
9. **Subject name** フィールドに **default** を入力します。
10. **Create** をクリックします。

関連情報

- 認証について
- RBAC の概要

3.5. 初めてのイメージのデプロイ

OpenShift Container Platform でアプリケーションをデプロイする最も簡単な方法は、既存のコンテナーイメージを実行することです。以下の手順では、**national-parks-app** という名前のアプリケーションのフロントエンドコンポーネントをデプロイします。Web アプリケーションは対話型のマップを表示します。マップには、全世界の主要な 国立公園の場所が表示されます。

前提条件

- OpenShift Container Platform Web コンソールにログインしている。
- **Developer** パースペクティブを使用している。

- OpenShift Container Platform でアプリケーションおよび他のワークロードを作成するための適切なプロジェクト内のロールおよびパーミッションがある。

手順

1. **Developer** パースペクティブの **+Add** ビューで、**Container images** をクリックしてダイアログを開きます。
2. **Image Name** フィールドに、**quay.io/openshiftroadshow/parksmap:latest** を入力します。
3. 現在の値が以下のようであることを確認します。
   a. アプリケーション:**national-parks-app**
   b. 名前:**parksmap**
4. **Resource** に **Deployment** を選択します。
5. **Create route to the application**を選択します。
6. **Advanced Options** セクションで **Labels** をクリックし、ラベルを追加して後でこのデプロイメントを特定するのを容易にします。ラベルを使用すると、Web コンソールおよびコマンドラインで、コンポーネントを特定し、絞り込むことができます。以下のラベルを追加します。
   - **app=national-parks-app**
   - **component=parksmap**
   - **role=frontend**
7. **Create** をクリックします。

Topology ページにリダイレクトされ、ここで **national-parks-app** アプリケーションに **parksmap** デプロイメントを確認できます。

関連情報

- Developer パースペクティブを使用したアプリケーションの作成
- Web コンソールを使用したプロジェクトの表示
- アプリケーションのトポロジーの表示
- Web コンソールを使用したプロジェクトの削除

3.5.1. Pod の検証

OpenShift Container Platform は、Pod の Kubernetes の概念を活用しています。これはホスト上に共にデプロイされる 1 つ以上のコンテナーであり、定義、デプロイ、管理される最小のコンピュート単位です。Pod は、コンテナーに対して、(物理または仮想) マシンインスタンスとほぼ同等のものです。

Overview パネルで、**parksmap** デプロイメントの多くの機能にアクセスできます。**Details** タブおよび **Resources** タブを使用すると、アプリケーション Pod をスケーリングし、ビルドのステータス、サービス、ルートを確認できます。

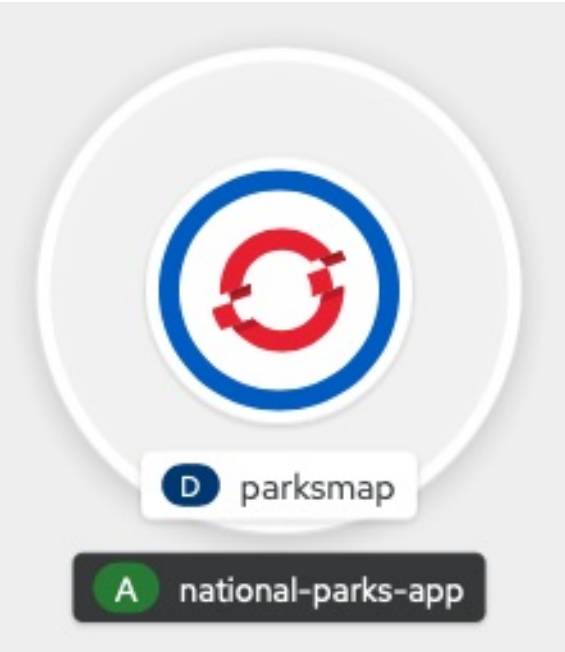
前提条件

- OpenShift Container Platform Web コンソールにログインしている。
- **Developer** パースペクティブを使用している。
- イメージがデプロイされている。

手順

- Topology ビューで **D parksmap** をクリックし、**Overview** パネルを開きます。

図3.2 parksmap デプロイメント



Overview パネルには、**Details**、**Resources**、および **Observe** のタブが含まれます。**Details** タブはデフォルトで表示されている場合があります。

表3.1 Overview パネルのタブの定義

タブ	定義
Details	アプリケーションをスケーリングし、ラベル、アノテーション、およびアプリケーションのステータスなどの Pod 設定を表示できます。
Resources	デプロイメントに関連付けられているリソースを表示します。
	Pod は、OpenShift Container Platform アプリケーションの基本単位です。使用されている Pod の数、それらのステータス、およびログを表示することができます。
	Pod 用に作成された **サービス** および割り当てられたポートは、**Services** の見出しにリスト表示されます。
	ルート は Pod への外部アクセスを有効にし、URL を使用してそれらにアクセスします。

タブ	定義
Observe	Pod に関連する各種の **イベント** および **メトリック** 情報を表示します。

関連情報

- アプリケーションおよびコンポーネントとの対話
- アプリケーション Pod のスケーリングおよびビルドとルートの確認
- Topology ビューに使用するラベルとアノテーション

3.5.2. アプリケーションのスケーリング

Kubernetes では、**Deployment** オブジェクトはアプリケーションのデプロイメント方法を定義します。ほとんどの場合、ユーザーは **Pod**、**Service**、**ReplicaSets**、および **Deployment** リソースを共に使用します。ほとんどの場合、OpenShift Container Platform は必要なリソースを作成します。

national-parks-app イメージをデプロイすると、デプロイメントリソースが作成されます。以下の例では、1 つの **Pod** のみがデプロイされます。

以下の手順では、2 つのインスタンスを使用するように **national-parks-image** をスケーリングします。

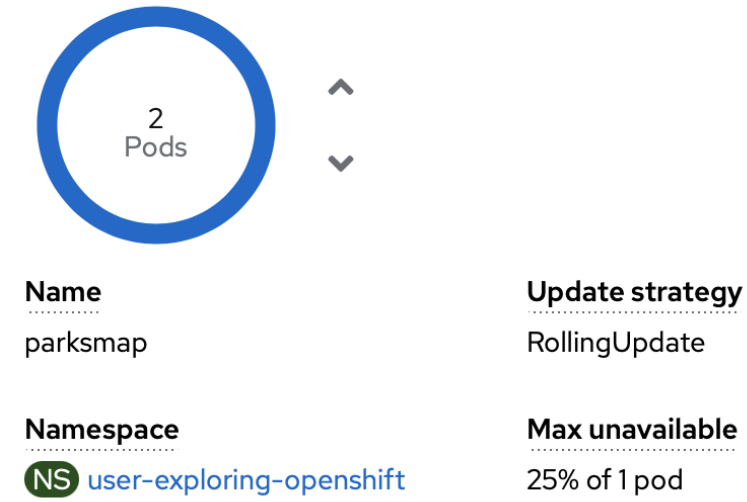
前提条件

- OpenShift Container Platform Web コンソールにログインしている。
- **Developer** パースペクティブを使用している。
- イメージがデプロイされている。

手順

1. **Topology** ビューで、**national-parks-app** アプリケーションをクリックします。
2. **Details** タブをクリックします。
3. 上矢印を使用して、Pod を 2 つのインスタンスにスケーリングします。

図3.3 アプリケーションのスケーリング



注記

OpenShift Container Platform が既存イメージの新規インスタンスを起動しているので、アプリケーションのスケーリングを迅速に行うことができます。

4. 下矢印を使用して、Pod を 1 つのインスタンスにスケールダウンします。

関連情報

- クラスターのスケーリングに関する推奨プラクティス
- Horizontal Pod Autoscaler について
- Vertical Pod Autoscaler Operator について

3.6. PYTHON アプリケーションのデプロイ

以下の手順では、**parksmap** アプリケーション用にバックエンドサービスをデプロイします。Python アプリケーションは、MongoDB データベースに対して二次元地理空間クエリーを実行して、世界中のすべての国立公園を探して地図上の座標を返します。

デプロイされるバックエンドサービスは **nationalparks** です。

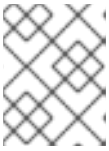
前提条件

- OpenShift Container Platform Web コンソールにログインしている。
- **Developer** パースペクティブを使用している。
- イメージがデプロイされている。

手順

1. **Developer** パースペクティブの **+Add** ビューで、**Import from Git** をクリックしてダイアログを開きます。

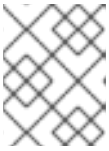
2. Git Repo URL フィールドに **https://github.com/openshift-roadshow/nationalparks-py.git** の URL を入力します。
ビルダーイメージは自動的に検出されます。



注記

検出されたビルダーイメージが Dockerfile の場合、**Edit Import Strategy** を選択します。**Builder Image** を選択し、**Python** をクリックします。

3. **General** セクションまでスクロールします。
4. 現在の値が以下のようであることを確認します。
   a. アプリケーション:**national-parks-app**
   b. 名前:**nationalparks**
5. **Resource** に **Deployment** を選択します。
6. **Create route to the application**を選択します。
7. **Advanced Options** セクションで **Labels** をクリックし、ラベルを追加して後でこのデプロイメントを特定するのを容易にします。ラベルを使用すると、Web コンソールおよびコマンドラインで、コンポーネントを特定し、絞り込むことができます。以下のラベルを追加します。
   a. **app=national-parks-app**
   b. **component=nationalparks**
   c. **role=backend**
   d. **type=parksmap-backend**
8. **Create** をクリックします。
9. **Topology** ビューで、**nationalparks** アプリケーションを選択します。



注記

Resources タブをクリックします。**Builds** セクションで、ビルドが実行されていることを確認できます。

関連情報

- サービスのアプリケーションへの追加
- Git のコードベースのインポートおよびアプリケーションの作成
- アプリケーションのトポロジーの表示
- Developer パースペクティブを使用したプロジェクトに対するアクセスパーミッションの提供
- Web コンソールを使用したプロジェクトの削除

3.7. データベースへの接続

national-parks-app アプリケーションが場所情報を保存する MongoDB データベースをデプロイして接続します。**national-parks-app** アプリケーションをマップ可視化ツールのバックエンドとしてマークすると、**parksmap** デプロイメントは OpenShift Container Platform の検出メカニズムを使用して地図を自動的に表示します。

前提条件

- OpenShift Container Platform Web コンソールにログインしている。
- **Developer** パースペクティブを使用している。
- イメージがデプロイされている。

手順

1. **Developer** パースペクティブの **+Add** ビューで、**Container images** をクリックしてダイアログを開きます。
2. **Image Name** フィールドに **quay.io/centos7/mongodb-36-centos7** を入力します。
3. **Runtime icon** フィールドで、**mongodb** を検索します。
4. **General** セクションまでスクロールダウンします。
5. 現在の値が以下のようであることを確認します。
   a. アプリケーション:**national-parks-app**
   b. 名前:**mongodb-nationalparks**
6. **Resource** に **Deployment** を選択します。
7. **Create route to the application**の横にあるチェックボックスの選択を解除します。
8. **Advanced Options** セクションで **Deployment** をクリックし、以下の環境変数を追加します。

   表3.2 環境変数の名前および値

名前	値
MONGODB_USER	mongodb
MONGODB_PASSWORD	mongodb
MONGODB_DATABASE	mongodb
MONGODB_ADMIN_PASSWORD	mongodb

9. **Create** をクリックします。

関連情報

- サービスのアプリケーションへの追加

- Web コンソールを使用したプロジェクトの表示
- アプリケーションのトポロジーの表示
- Developer パースペクティブを使用したプロジェクトに対するアクセスパーミッションの提供
- Web コンソールを使用したプロジェクトの削除

3.7.1. シークレットの作成

Secret オブジェクトはパスワード、OpenShift Container Platform クライアント設定ファイル、プライベートソースリポジトリーの認証情報などの機密情報を保持するメカニズムを提供します。シークレットは機密内容を Pod から切り離します。シークレットはボリュームプラグインを使用してコンテナーにマウントすることも、システムが Pod の代わりにシークレットを使用して各種アクションを実行することもできます。以下の手順では、シークレット **nationalparks-mongodb-parameters** を追加し、それを **nationalparks** ワークロードにマウントします。

前提条件

- OpenShift Container Platform Web コンソールにログインしている。
- **Developer** パースペクティブを使用している。
- イメージがデプロイされている。

手順

1. **Developer** パースペクティブで、左側のナビゲーションにある **Secrets** に移動し、**Secrets** をクリックします。
2. **Create → Key/value secret** をクリックします。
   a. **Secret name** フィールドに **nationalparks-mongodb-parameters** を入力します。
   b. **Key** および **Value** に以下の値を入力します。

   表3.3 シークレットのキーおよび値

キー	値
MONGODB_USER	**mongodb**
DATABASE_SERVICE_NAME	**mongodb-nationalparks**
MONGODB_PASSWORD	**mongodb**
MONGODB_DATABASE	**mongodb**
MONGODB_ADMIN_PASSWORD	**mongodb**

   c. **Create** をクリックします。
3. **Add Secret to workload** をクリックします。

a. ドロップダウンメニューから、追加するワークロードとして **nationalparks** を選択します。

b. **Save** をクリックします。

設定をこのように変更すると、環境変数が適切に挿入された状態で **nationalparks** デプロイメントの新しいロールアウトがトリガーされます。

関連情報

- シークレットについて

3.7.2. データの読み込みおよび国立公園の地図表示

parksmap および **nationalparks** アプリケーションをデプロイし、**mongodb-nationalparks** データベースをデプロイしました。ただし、データベース **に** データが読み込まれていません。データを読み込む前に、**mongodb-nationalparks** および **nationalparks** デプロイメントに適切なラベルを追加します。

前提条件

- OpenShift Container Platform Web コンソールにログインしている。
- **Developer** パースペクティブを使用している。
- イメージがデプロイされている。

手順

1. **Topology** ビューから **nationalparks** デプロイメントに移動し、**Resources** をクリックしてルート情報を取得します。
2. URL を Web ブラウザーにコピーアンドペーストし、URL の最後に以下を追加します。

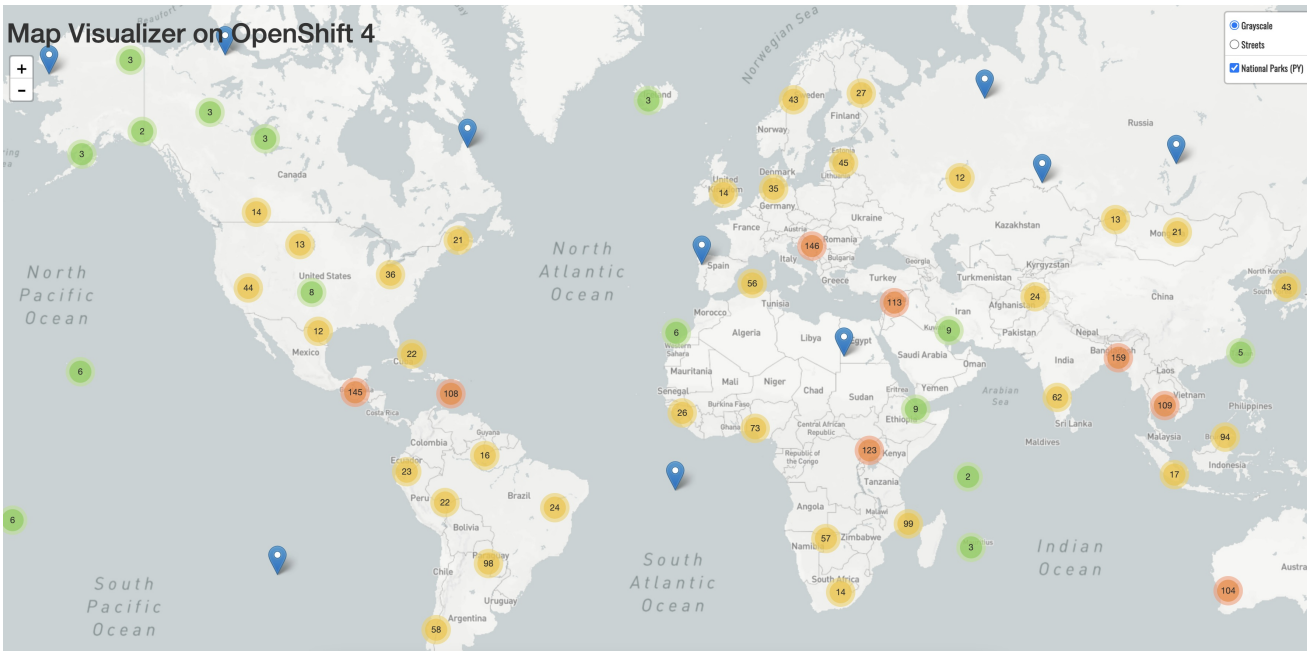
```
/ws/data/load
```

出力例

```
Items inserted in database: 2893
```

3. **Topology** ビューから **parksmap** デプロイメントに移動し、**Resources** をクリックしてルート情報を取得します。
4. URL をコピーして Web ブラウザーに貼り付けて、世界地図の国立公園を表示します。

図3.4 世界中の国立公園



関連情報

- Developer パースペクティブを使用したプロジェクトに対するアクセスパーミッションの提供
- Topology ビューに使用するラベルとアノテーション

第4章 CLI を使用したアプリケーションの作成およびビルド

4.1. 作業を開始する前に

- OpenShift CLI について を 確認します。
- 実行中の OpenShift Container Platform インスタンスにアクセスできる必要があります。アクセスできない場合は、クラスター管理者にお問い合わせください。
- OpenShift CLI (**oc**)を ダウンロードしてインストールし ておく。

4.2. CLI へのログイン

OpenShift CLI (**oc**) にログインしてクラスターにアクセスし、これを管理できます。

前提条件

- OpenShift Container Platform クラスターへのアクセス。
- OpenShift CLI (**oc**) がインストールされている。

手順

- ユーザー名とパスワード、OAuth トークン、または Web ブラウザーを使用して、CLI から OpenShift Container Platform にログインします。
  - ユーザー名とパスワードを使用してログインする場合:

    ```
    $ oc login -u=<username> -p=<password> --server=<your-openshift-server> --insecure-skip-tls-verify
    ```

  - OAuth トークンを使用してログインする場合:

    ```
    $ oc login <https://api.your-openshift-server.com> --token=<tokenID>
    ```

  - Web ブラウザーの場合:

    ```
    $ oc login <cluster_url> --web
    ```

これで、プロジェクトを作成でき、クラスターを管理するための他のコマンドを実行することができます。

関連情報

- oc login
- oc logout

4.3. 新規プロジェクトの作成

プロジェクトにより、ユーザーコミュニティーは、切り離された状態で独自のコンテンツを整理し、管理することができます。プロジェクトは、Kubernetes namespace に対する OpenShift Container

Platform 拡張です。プロジェクトには、ユーザーのセルフプロビジョニングを可能にする追加機能があります。

ユーザーは、管理者からプロジェクトへのアクセス権限を受け取る必要があります。クラスター管理者は、開発者が専用のプロジェクトを作成するのを許可できます。ほとんどの場合、ユーザーは自動的に専用のプロジェクトにアクセスできます。

各プロジェクトには、オブジェクト、ポリシー、制約、およびサービスアカウントの独自のセットがあります。

前提条件

- OpenShift Container Platform クラスターへのアクセス。
- OpenShift CLI (**oc**) がインストールされている。

手順

- 新規プロジェクトを作成するには、以下のコマンドを入力します。

```
$ oc new-project user-getting-started --display-name="Getting Started with OpenShift"
```

出力例

```
Now using project "user-getting-started" on server "https://openshift.example.com:6443".
```

関連情報

- oc new-project

4.4. 表示パーミッションの付与

OpenShift Container Platform は、すべてのプロジェクトにいくつかの特別なサービスアカウントを自動的に作成します。デフォルトのサービスアカウントは、Pod を実行する役割を担います。OpenShift Container Platform は、このサービスアカウントを使用して、起動するすべての Pod に挿入します。

以下の手順では、デフォルトの **ServiceAccount** オブジェクトに **RoleBinding** オブジェクトを作成します。サービスアカウントは OpenShift Container Platform API と通信し、プロジェクト内の Pod、サービス、およびリソースについて把握します。

前提条件

- OpenShift Container Platform クラスターへのアクセス。
- OpenShift CLI (**oc**) がインストールされている。
- イメージがデプロイされている。
- **cluster-admin** または **project-admin** 権限がある。

手順

- **user-getting-started プロジェクト** の default サービスアカウントに view ロールを追加するには、以下のコマンドを入力します。

```
$ oc adm policy add-role-to-user view -z default -n user-getting-started
```

関連情報

- 認証について
- RBAC の概要
- oc policy add-role-to-user

4.5. 初めてのイメージのデプロイ

OpenShift Container Platform でアプリケーションをデプロイする最も簡単な方法は、既存のコンテナーイメージを実行することです。以下の手順では、**national-parks-app** という名前のアプリケーションのフロントエンドコンポーネントをデプロイします。Web アプリケーションは対話型のマップを表示します。マップには、全世界の主要な 国立公園の場所が表示されます。

前提条件

- OpenShift Container Platform クラスターへのアクセス。
- OpenShift CLI (**oc**) がインストールされている。

手順

- アプリケーションをデプロイするには、以下のコマンドを入力します。

```
$ oc new-app quay.io/openshiftroadshow/parksmap:latest --name=parksmap -l
'app=national-parks-app,component=parksmap,role=frontend,app.kubernetes.io/part-
of=national-parks-app'
```

出力例

```
--> Found container image 0c2f55f (12 months old) from quay.io for
"quay.io/openshiftroadshow/parksmap:latest"

    * An image stream tag will be created as "parksmap:latest" that will track this image

--> Creating resources with label app=national-parks-app,app.kubernetes.io/part-of=national-
parks-app,component=parksmap,role=frontend ...
    imagestream.image.openshift.io "parksmap" created
    deployment.apps "parksmap" created
    service "parksmap" created
--> Success
```

関連情報

- oc new-app

4.5.1. ルートの作成

外部クライアントは、ルーティング層を使用して OpenShift Container Platform で実行されているアプリケーションにアクセスできます。その背後にあるデータオブジェクトは **ルート** です。デフォルトの

OpenShift Container Platform ルーター (HAProxy) は、受信リクエストの HTTP ヘッダーを使用して、接続をプロキシー処理する場所を決定します。

オプションとして、ルートに TLS などのセキュリティーを定義できます。

前提条件

- OpenShift Container Platform クラスターへのアクセス。
- OpenShift CLI (**oc**) がインストールされている。
- イメージがデプロイされている。
- **cluster-admin** または **project-admin** 権限がある。

手順

1. 作成したアプリケーションサービスを取得するには、以下のコマンドを入力します。

```
$ oc get service
```

出力例

```
NAME      TYPE       CLUSTER-IP         EXTERNAL-IP    PORT(S)   AGE
parksmap  ClusterIP  <your-cluster-IP>  <123.456.789>  8080/TCP  8m29s
```

2. ルートを作成するには、以下のコマンドを入力します。

```
$ oc create route edge parksmap --service=parksmap
```

出力例

```
route.route.openshift.io/parksmap created
```

3. 作成したアプリケーションのルートを取得するには、以下のコマンドを入力します。

```
$ oc get route
```

出力例

```
NAME       HOST/PORT                                          PATH   SERVICES   PORT
TERMINATION   WILDCARD
parksmap   parksmap-user-getting-started.apps.cluster.example.com          parksmap
8080-tcp   edge          None
```

関連情報

- oc create route edge
- oc get

4.5.2. Pod の検証

OpenShift Container Platform は、Pod の Kubernetes の概念を活用しています。これはホスト上に共にデプロイされる 1 つ以上のコンテナーであり、定義、デプロイ、管理される最小のコンピュート単位です。Pod は、コンテナーに対して、 (物理または仮想) マシンインスタンスとほぼ同等のものです。

クラスターの Pod を表示し、それらの Pod およびクラスター全体としての正常性を判別できます。

前提条件

- OpenShift Container Platform クラスターへのアクセス。
- OpenShift CLI (**oc**) がインストールされている。
- イメージがデプロイされている。

手順

1. ノード名と共にすべての Pod をリスト表示するには、以下のコマンドを入力します。

```
$ oc get pods
```

出力例

```
NAME                     READY   STATUS    RESTARTS   AGE
parksmap-5f9579955-6sng8   1/1     Running   0          77s
```

2. すべての Pod の詳細をリスト表示するには、以下のコマンドを入力します。

```
$ oc describe pods
```

出力例

```
Name:         parksmap-848bd4954b-5pvcc
Namespace:    user-getting-started
Priority:     0
Node:         ci-ln-fr1rt92-72292-4fzf9-worker-a-g9g7c/10.0.128.4
Start Time:   Sun, 13 Feb 2022 14:14:14 -0500
Labels:       app=national-parks-app
              app.kubernetes.io/part-of=national-parks-app
              component=parksmap
              deployment=parksmap
              pod-template-hash=848bd4954b
              role=frontend
Annotations:  k8s.v1.cni.cncf.io/network-status:
                [{
                    "name": "openshift-sdn",
                    "interface": "eth0",
                    "ips": [
                        "10.131.0.14"
                    ],
                    "default": true,
                    "dns": {}
                }]
              k8s.v1.cni.cncf.io/network-status:
                [{
```

```
                "name": "openshift-sdn",
                "interface": "eth0",
                "ips": [
                    "10.131.0.14"
                ],
                "default": true,
                "dns": {}
            }]
          openshift.io/generated-by: OpenShiftNewApp
          openshift.io/scc: restricted
Status:       Running
IP:           10.131.0.14
IPs:
  IP:           10.131.0.14
Controlled By:  ReplicaSet/parksmap-848bd4954b
Containers:
  parksmap:
    Container ID:   cri-
o://4b2625d4f61861e33cc95ad6d455915ea8ff6b75e17650538cc33c1e3e26aeb8
    Image:
quay.io/openshiftroadshow/parksmap@sha256:89d1e324846cb431df9039e1a7fd0ed2ba0c51a
afbae73f2abd70a83d5fa173b
    Image ID:
quay.io/openshiftroadshow/parksmap@sha256:89d1e324846cb431df9039e1a7fd0ed2ba0c51a
afbae73f2abd70a83d5fa173b
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Sun, 13 Feb 2022 14:14:25 -0500
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-6f844 (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  kube-api-access-6f844:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
    ConfigMapName:           openshift-service-ca.crt
    ConfigMapOptional:       <nil>
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason          Age   From               Message
  ----    ------          ----  ----               -------
```

```
  Normal  Scheduled       46s   default-scheduler  Successfully assigned user-getting-
started/parksmap-848bd4954b-5pvcc to ci-ln-fr1rt92-72292-4fzf9-worker-a-g9g7c
  Normal  AddedInterface  44s   multus             Add eth0 [10.131.0.14/23] from openshift-sdn
  Normal  Pulling         44s   kubelet            Pulling image
"quay.io/openshiftroadshow/parksmap@sha256:89d1e324846cb431df9039e1a7fd0ed2ba0c51
aafbae73f2abd70a83d5fa173b"
  Normal  Pulled          35s   kubelet            Successfully pulled image
"quay.io/openshiftroadshow/parksmap@sha256:89d1e324846cb431df9039e1a7fd0ed2ba0c51
aafbae73f2abd70a83d5fa173b" in 9.49243308s
  Normal  Created         35s   kubelet            Created container parksmap
  Normal  Started         35s   kubelet            Started container parksmap
```

関連情報

- oc describe
- oc get
- oc label
- Pod の表示
- Pod ログの表示

4.5.3. アプリケーションのスケーリング

Kubernetes では、**Deployment** オブジェクトはアプリケーションのデプロイメント方法を定義します。ほとんどの場合、ユーザーは **Pod**、**Service**、**ReplicaSets**、および **Deployment** リソースを共に使用します。ほとんどの場合、OpenShift Container Platform は必要なリソースを作成します。

national-parks-app イメージをデプロイすると、デプロイメントリソースが作成されます。以下の例では、1 つの **Pod** のみがデプロイされます。

以下の手順では、2 つのインスタンスを使用するように **national-parks-image** をスケーリングします。

前提条件

- OpenShift Container Platform クラスターへのアクセス。
- OpenShift CLI (**oc**) がインストールされている。
- イメージがデプロイされている。

手順

- アプリケーションを 1 つの Pod インスタンスから 2 つの Pod インスタンスにスケーリングするには、以下のコマンドを入力します。

```
$ oc scale --current-replicas=1 --replicas=2 deployment/parksmap
```

出力例

```
deployment.apps/parksmap scaled
```

検証

1. アプリケーションが適切にスケーリングされていることを確認するには、以下のコマンドを入力します。

```
$ oc get pods
```

出力例

```
NAME                    READY   STATUS    RESTARTS   AGE
parksmap-5f9579955-6sng8   1/1     Running   0          7m39s
parksmap-5f9579955-8tgft   1/1     Running   0          24s
```

2. アプリケーションを以前の 1 つの Pod インスタンスにスケールダウンするには、以下のコマンドを入力します。

```
$ oc scale --current-replicas=2 --replicas=1 deployment/parksmap
```

関連情報

- oc scale

4.6. PYTHON アプリケーションのデプロイ

以下の手順では、**parksmap** アプリケーション用にバックエンドサービスをデプロイします。Python アプリケーションは、MongoDB データベースに対して二次元地理空間クエリーを実行して、世界中のすべての国立公園を探して地図上の座標を返します。

デプロイされるバックエンドサービスは、**nationalparks** です。

前提条件

- OpenShift Container Platform クラスターへのアクセス。
- OpenShift CLI (**oc**) がインストールされている。
- イメージがデプロイされている。

手順

1. 新しい Python アプリケーションを作成するには、以下のコマンドを入力します。

```
$ oc new-app python~https://github.com/openshift-roadshow/nationalparks-py.git --name
nationalparks -l 'app=national-parks-
app,component=nationalparks,role=backend,app.kubernetes.io/part-of=national-parks-
app,app.kubernetes.io/name=python' --allow-missing-images=true
```

出力例

```
--> Found image 0406f6c (13 days old) in image stream "openshift/python" under tag "3.9-
ubi9" for "python"

    Python 3.9
```

```
    ----------
    Python 3.9 available as container is a base platform for building and running various
Python 3.9 applications and frameworks. Python is an easy to learn, powerful programming
language. It has efficient high-level data structures and a simple but effective approach to
object-oriented programming. Python's elegant syntax and dynamic typing, together with its
interpreted nature, make it an ideal language for scripting and rapid application development
in many areas on most platforms.

    Tags: builder, python, python39, python-39, rh-python39

    * A source build using source code from https://github.com/openshift-
roadshow/nationalparks-py.git will be created
      * The resulting image will be pushed to image stream tag "nationalparks:latest"
      * Use 'oc start-build' to trigger a new build

--> Creating resources with label app=national-parks-
app,app.kubernetes.io/name=python,app.kubernetes.io/part-of=national-parks-
app,component=nationalparks,role=backend ...
    imagestream.image.openshift.io "nationalparks" created
    buildconfig.build.openshift.io "nationalparks" created
    deployment.apps "nationalparks" created
    service "nationalparks" created
--> Success
```

2. アプリケーション **nationalparks** を公開するルートを作成するには、以下のコマンドを入力します。

```
$ oc create route edge nationalparks --service=nationalparks
```

出力例

```
route.route.openshift.io/parksmap created
```

3. 作成したアプリケーションのルートを取得するには、以下のコマンドを入力します。

```
$ oc get route
```

出力例

```
NAME            HOST/PORT                                                PATH   SERVICES
PORT       TERMINATION   WILDCARD
nationalparks    nationalparks-user-getting-started.apps.cluster.example.com
nationalparks   8080-tcp   edge          None
parksmap         parksmap-user-getting-started.apps.cluster.example.com
parksmap        8080-tcp   edge          None
```

関連情報

- oc new-app

4.7. データベースへの接続

national-parks-app アプリケーションが場所情報を保存する MongoDB データベースをデプロイして

接続します。**national-parks-app** アプリケーションをマップ可視化ツールのバックエンドとしてマークすると、**parksmap** デプロイメントは OpenShift Container Platform の検出メカニズムを使用して地図を自動的に表示します。

前提条件

- OpenShift Container Platform クラスターへのアクセス。
- OpenShift CLI (**oc**) がインストールされている。
- イメージがデプロイされている。

手順

- データベースに接続するには、以下のコマンドを入力します。

```
$ oc new-app quay.io/centos7/mongodb-36-centos7 --name mongodb-nationalparks -e
MONGODB_USER=mongodb -e MONGODB_PASSWORD=mongodb -e
MONGODB_DATABASE=mongodb -e MONGODB_ADMIN_PASSWORD=mongodb -l
'app.kubernetes.io/part-of=national-parks-app,app.kubernetes.io/name=mongodb'
```

出力例

```
--> Found container image dc18f52 (8 months old) from quay.io for
"quay.io/centos7/mongodb-36-centos7"

    MongoDB 3.6
    -----------
    MongoDB (from humongous) is a free and open-source cross-platform document-oriented
database program. Classified as a NoSQL database program, MongoDB uses JSON-like
documents with schemas. This container image contains programs to run mongod server.

    Tags: database, mongodb, rh-mongodb36

    * An image stream tag will be created as "mongodb-nationalparks:latest" that will track this
image

--> Creating resources with label app.kubernetes.io/name=mongodb,app.kubernetes.io/part-
of=national-parks-app ...
    imagestream.image.openshift.io "mongodb-nationalparks" created
    deployment.apps "mongodb-nationalparks" created
    service "mongodb-nationalparks" created
--> Success
```

関連情報

- oc new-project

4.7.1. シークレットの作成

Secret オブジェクトはパスワード、OpenShift Container Platform クライアント設定ファイル、プライベートソースリポジトリーの認証情報などの機密情報を保持するメカニズムを提供します。シークレットは機密内容を Pod から切り離します。シークレットはボリュームプラグインを使用してコンテナー

にマウントすることも、システムが Pod の代わりにシークレットを使用して各種アクションを実行することもできます。以下の手順では、シークレット **nationalparks-mongodb-parameters** を追加し、それを **nationalparks** ワークロードにマウントします。

前提条件

- OpenShift Container Platform クラスターへのアクセス。
- OpenShift CLI (**oc**) がインストールされている。
- イメージがデプロイされている。

手順

1. シークレットを作成するには、以下のコマンドを入力します。

```
$ oc create secret generic nationalparks-mongodb-parameters --from-literal=DATABASE_SERVICE_NAME=mongodb-nationalparks --from-literal=MONGODB_USER=mongodb --from-literal=MONGODB_PASSWORD=mongodb --from-literal=MONGODB_DATABASE=mongodb --from-literal=MONGODB_ADMIN_PASSWORD=mongodb
```

出力例

```
secret/nationalparks-mongodb-parameters created
```

2. mongodb シークレットを **nationalpartks** ワークロードにアタッチするように環境変数を更新するには、以下のコマンドを入力します。

```
$ oc set env --from=secret/nationalparks-mongodb-parameters deploy/nationalparks
```

出力例

```
deployment.apps/nationalparks updated
```

3. **nationalparks** デプロイメントのステータスを表示するには、以下のコマンドを入力します。

```
$ oc rollout status deployment nationalparks
```

出力例

```
deployment "nationalparks" successfully rolled out
```

4. **mongodb-nationalparks** デプロイメントのステータスを表示するには、以下のコマンドを入力します。

```
$ oc rollout status deployment mongodb-nationalparks
```

出力例

```
deployment "nationalparks" successfully rolled out
deployment "mongodb-nationalparks" successfully rolled out
```

関連情報

- oc create secret generic
- oc set env
- oc rollout status

4.7.2. データの読み込みおよび国立公園の地図表示

parksmap および **nationalparks** アプリケーションをデプロイし、**mongodb-nationalparks** データベースをデプロイしました。ただし、データベース **に** データが読み込まれていません。

前提条件

- OpenShift Container Platform クラスターへのアクセス。
- OpenShift CLI (**oc**) がインストールされている。
- イメージがデプロイされている。

手順

1. 国立公園のデータを読み込むには、以下のコマンドを入力します。

```
$ oc exec $(oc get pods -l component=nationalparks | tail -n 1 | awk '{print $1;}') -- curl -s http://localhost:8080/ws/data/load
```

出力例

```
"Items inserted in database: 2893"
```

2. データが適切にロードされていることを確認するには、以下のコマンドを入力します。

```
$ oc exec $(oc get pods -l component=nationalparks | tail -n 1 | awk '{print $1;}') -- curl -s http://localhost:8080/ws/data/all
```

出力例 (一部)

```
, {"id": "Great Zimbabwe", "latitude": "-20.2674635", "longitude": "30.9337986", "name": "Great Zimbabwe"}]
```

3. ラベルをルートに追加するには、以下のコマンドを入力します。

```
$ oc label route nationalparks type=parksmap-backend
```

出力例

```
route.route.openshift.io/nationalparks labeled
```

4. マップを表示するためのルートを取得するには、以下のコマンドを入力します。

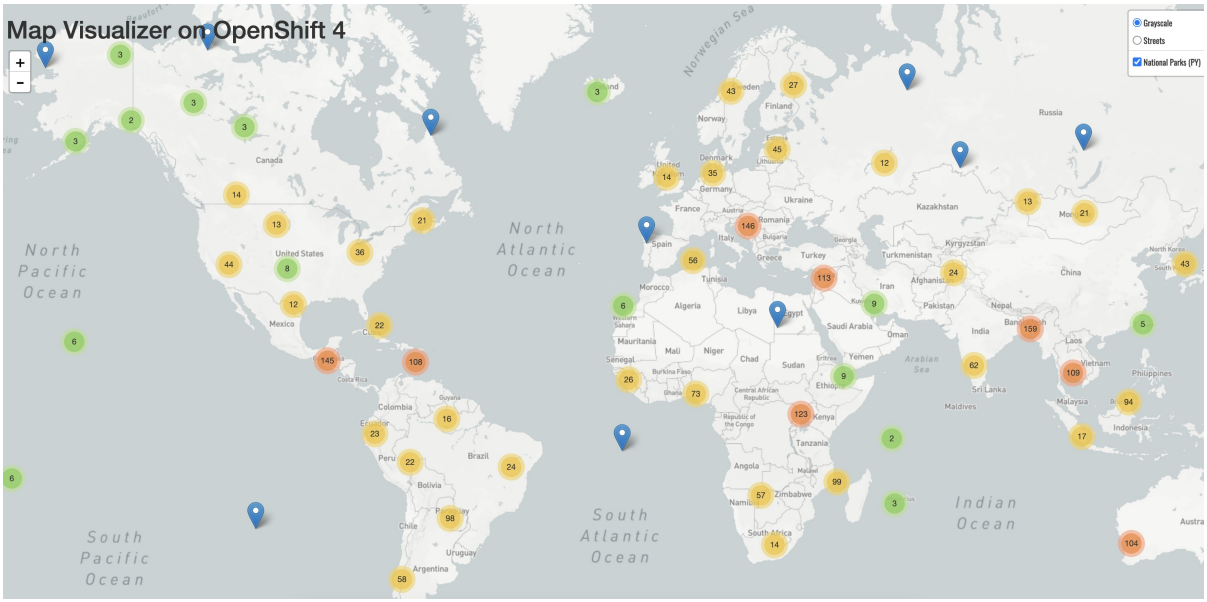
```
$ oc get routes
```

出力例

```
NAME          HOST/PORT                                                   PATH   SERVICES      PORT
TERMINATION   WILDCARD
nationalparks   nationalparks-user-getting-started.apps.cluster.example.com
nationalparks   8080-tcp   edge          None
parksmap        parksmap-user-getting-started.apps.cluster.example.com                parksmap
8080-tcp   edge          None
```

5. 上記で取得した **HOST/PORT** パスを Web ブラウザーにコピーアンドペーストします。ブラウザーに、世界中の国立公園の地図が表示されるはずです。

図4.1 世界中の国立公園



関連情報

- oc exec
- oc label
- oc get