Oracle Private Cloud ApplianceおよびKubernetes上のOracle WebLogic Server

Oracle Exalogic Elastic Cloudシステムからの移行を含む、Oracle Private Cloud ApplianceおよびKubernetesへのOracle WebLogic Serverのデプロイ

ホワイト・ペーパー/2019年8月28日

ORACLE

本書の目的

本書では、Oracle WebLogic ServerアプリケーションをOracle Private Cloud ApplianceまたはOracle Private Cloud at CustomerのKubernetes上にデプロイする方法について説明し、オラクルによって全面的にサポートされ、異なるクラウド環境間で移植可能なクラウド・ネイティブ・インフラストラクチャにおいて、それらのアプリケーションを実行する方法をご紹介します。本書では、Oracle Exalogic Elastic Cloudシステム上にデプロイされたアプリケーションをアプリケーションに変更を加えることなくこのインフラストラクチャに移行する方法に焦点を当て、読者が今日のクラウド・ネイティブ・インフラストラクチャを導入する際に、アプリケーションへの投資を維持できるようにします。

免責事項

本文書には、ソフトウェアや印刷物など、いかなる形式のものも含め、オラクルの独占的な所有物である占有情報が含まれます。この機密文書へのアクセスと使用は、締結および遵守に同意したOracle Software License and Service Agreementの諸条件に従うものとします。本文書と本文書に含まれる情報は、オラクルの事前の書面による同意なしに、公開、複製、再作成、またはオラクルの外部に配布することはできません。本文書は、ライセンス契約の一部ではありません。また、オラクル、オラクルの子会社または関連会社との契約に組み込むことはできません。

本書は情報提供のみを目的としており、記載した製品機能の実装およびアップグレードの計画を支援することのみを意図しています。マテリアルやコード、機能の提供をコミットメント（確約）するものではなく、購買を決定する際の判断材料になさらないでください。本書に記載されている機能の開発、リリース、および時期については、弊社の裁量により決定されます。

製品アーキテクチャの性質により、コードが大幅に不安定化するリスクなしに、本書に記載されているすべての機能を安全に含めることができない場合があります。

目次

はじめに

Oracle WebLogic Serverは、DockerコンテナおよびKubernetes上で正しく動作することを保証されています。オラクルは、既存の物理または仮想システムからKubernetes内にOracle WebLogic Serverアプリケーションを移行して実行可能にする処理を容易にするオープン・ソースのツールを開発しました。それらのツールにより、既存のOracle WebLogic Serverドメインの構成およびアプリケーションのイントロスペクション、それらからのDockerイメージの作成、Kubernetesへのアプリケーションのデプロイ、それらの監視、ログの保持、アプリケーションのライフサイクル管理、継続的に処理されるメンテナンスとアプリケーション更新プロセスにおける自動化CI/CDプロセスとDevOpsプロセスの使用を実現できます。

Oracle Private Cloud ApplianceとOracle Private Cloud at Customerでは、Oracle Container Runtime for DockerおよびOracle Container Services for Use with Kubernetesを含む、Oracle Linux Cloud Native Environmentが完全にサポートされます。それらは、Oracle WebLogic Severのアプリケーションにとって、DockerおよびKubernetes内で実行される場合に理想的な実行時環境であり、オラクルの完全な統合システムによってサポートされます。オラクルでは、Oracle Exalogic Elastic Cloudシステム上ですでにOracle WebLogic Serverアプリケーションを実行しており、クラウド・ネイティブ・インフラストラクチャとDevOpsプラクティスの導入を望んでいるお客様に対し、Oracle Private Cloud ApplianceとOracle Private Cloud at Customerへの移行を検討するよう推奨しています。

このホワイト・ペーパーでは、サポート・ツールと推奨プラクティスの概要を含め、そのような移行作業を実施する方法について説明します。説明および例では特に、Oracle Private Cloud Applianceをターゲット・システムとするシナリオを取り上げますが、ツールとプラクティスはOracle Private Cloud at Customerにも適用可能です。

DockerとKubernetesへ向かう市場とテクノロジーのトレンド

プライベートおよびパブリックのクラウド・テクノロジーの導入に向けた業界の勢いは引き続き強まっています。クラウド・アプリケーションを構築するエンタープライズのお客様は、アプリケーションの開発とデプロイメントのためにマイクロサービスとコンテナベースのアーキテクチャを利用しており、これらのアーキテクチャにより、DockerコンテナおよびKubernetesオーケストレーション・テクノロジーをクラウド・ネイティブ・インフラストラクチャの事実上の標準として導入する動きが促進されてきました。ベンダーとオープン・ソース・プロジェクトは、Kubernetes上へのエンタープライズ・アプリケーションのデプロイメントに対応しているソフトウェア・ツールや機能を提供することによって、この動きに呼応しています。

こうしたトレンドは、より従来型の物理または仮想化システム上で数十、数百、あるいは数千単位のアプリケーションを実行しているOracle WebLogic Serverのお客様に影響を与えます。Oracle WebLogic Serverの多くのお客様が、マイクロサービス・アプリケーション・インフラストラクチャ向けに構築されている同じクラウド・ネイティブ・インフラストラクチャでアプリケーションを実行しているか、実行することを計画しています。お客様の意向は、デプロイ、監視、ロギング、トレース、ロード・バランシングのためのアプリケーションすべてにおいて、Helm、Prometheus、Grafana、Elastic Stack、Ingressなどのテクノロジーを含め、同じツールと機能を使用および利用することです。また、Oracle WebLogic Serverアプリケーションを新しいマイクロサービスと統合することも考えています。

オラクルでは、従来の物理および仮想システムからKubernetesへのOracle WebLogic Serverアプリケーションの移行作業をお客様が容易に行えるようにするツールを開発することによって、こうした業界のトレンドに対処してきました。移行後のアプリケーションは、Oracle WebLogic Serverとサポート・ツールに備わった移植性により、プライベート・クラウドまたはパブリック・クラウドで柔軟に実行できます。

Kubernetes上でのOracle WebLogic Serverアプリケーションのデプロイと管理

Oracle WebLogic Server 12.2.1.3は、DockerおよびKubernetesでの正しい動作を保証されており、将来のバージョンも同様に保証されます。Oracle WebLogic Server DockerコンテナのGitHubプロジェクトには、Oracle WebLogic Server Dockerイメージのカスタマイズと作成に使用できるDockerファイルとスクリプトが含まれています。

イメージの作成方法、アプリケーションのデプロイ方法、データソースとJMSサーバーなどの構成方法を示すサンプルが数多く用意されています。Oracle WebLogic Service Dockerイメージは、Oracle Container RegistryとDocker Storeの両方のパブリック・リポジトリで公開されています。Oracle WebLogic Serverのイメージは、レイヤー化、イメージ外部での状態の保持、コンテナ内での単一サーバーの実行において、Dockerのベスト・プラクティスに従っています。それらのイメージは、開発と本番の両方のユースケースでサポートされています。My Oracle Supportのドキュメント2017945.1を参照してください。

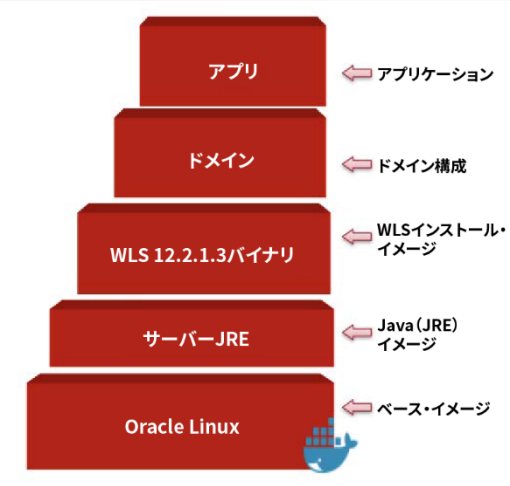

図1：Oracle WebLogic Server Dockerイメージ。

動作保証およびサポートされているバージョンのDockerおよびKubernetesについては、Supported Virtualizationのページを参照してください。

KubernetesでOracle WebLogic Serverドメインを実行するには、以下で説明するオープン・ソースのツールを使用して、ログの管理、監視、保持、およびOracle WebLogic Server Dockerのイメージおよびアプリケーションの保守を行うことができます。

WebLogic Kubernetes Operator

Kubernetes Operatorは、アプリケーション固有のコントローラの1つで、Kubernetes APIを拡張して特定のアプリケーション・タイプのインスタンスを作成、構成、管理します。

オラクルでは、WebLogic Kubernetes Operatorを提供し、Oracle WebLogic Serverインスタンスのコンテナ・インフラストラクチャとしてのKubernetesの使用方法を簡素化するOperatorパターンを導入しました。WebLogic Kubernetes Operatorは、Kubernetesを拡張して、Oracle WebLogic Serverドメインを作成、構成、管理します。WebLogic Kubernetes OperatorのGitHubプロジェクト、およびクイック・スタート・ガイドとサンプルを含むドキュメントを参照してください。

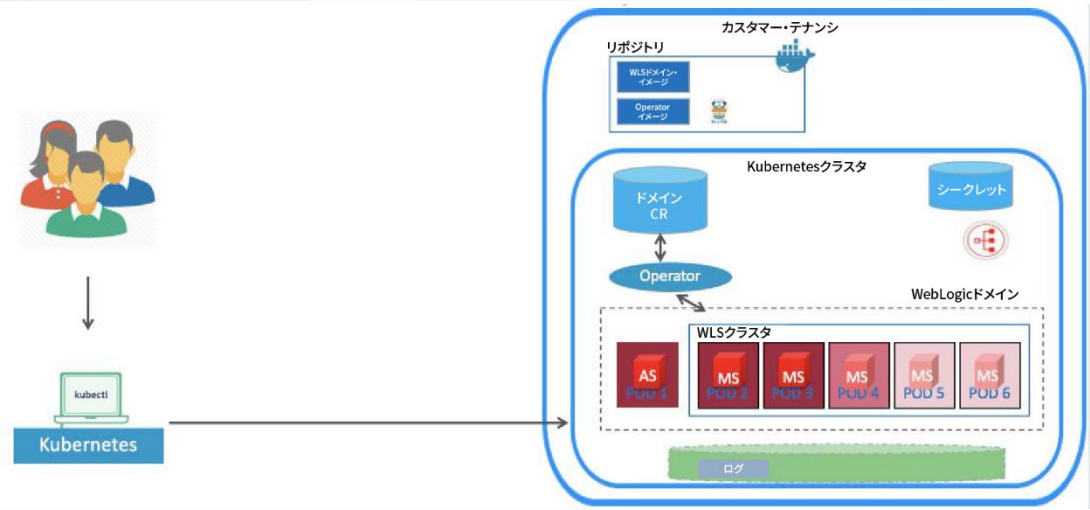

図2：WebLogic Kubernetes Operatorによって管理されるKubernetes上のOracle WebLogic Server

WebLogic Kubernetes Operatorには、Oracle WebLogic Serverドメインでライフサイクル操作を実行する方法に関する情報が取り込まれます。たとえば、Oracle WebLogic Serverのアプリケーションに期待されるサービス・レベルを達成するため、ドメインのローリング再起動などの動作を実行する場合に、状況に応じてOracle WebLogic Serverの正常なシャットダウンを実行する方法を認識しています。

Operatorによって、以下のようないくつかの操作が実行されます。

- Oracle WebLogic Serverドメインのプロビジョニング
- ライフサイクル管理
- Dockerイメージの更新
- Oracle WebLogic Serverクラスタの拡張および縮小
- ロールベースのアクセス制御（RBAC）ロールの定義
- KubernetesのPod間の通信、およびクラスタ化された管理対象サーバー間のロード・バランシング・リクエストのためのKubernetesサービスの作成

WebLogic Deploy Tooling

WebLogic Deploy Tooling（WDT）により、Oracle WebLogic Serverドメインのプロビジョニングとアプリケーション・デプロイメントを用意に自動化できます。保守が必要なWebLogic Scripting Tool（WLST）スクリプトを記述する代わりに、WDTでは、ドメイン、アプリケーション、アプリケーションが使用するリソースを記述した宣言的なメタデータ・モデルを作成します。このメタデータ・モデルにより、ドメインのライフサイクル操作を容易にプロビジョニング、デプロイ、実行することができ、このモデルはアプリケーションの継続的デリバリに最適です。WebLogic Deploy ToolingのGitHubプロジェクトには、ドキュメントとサンプルが含まれています。ブログ「Make WebLogic Domain Provisioning and Deployment Easy!」では、サンプル一式が紹介されています。

WDTにより、ドメイン構成とアプリケーション・バイナリをイントロスペクトし、Dockerイメージ内のドメインを含むか、または永続ボリューム上のドメイン構成を参照するDockerイメージを作成することができます。WDTは、10.3.6、12.1.3、12.2.1.3ドメインのイントロスペクション、およびOracle WebLogic Server 12.2.1.3でのそれらのドメインの移行と再作成に対応できます。

WDT Discover Domain Toolを使用してドメインをイントロスペクトすると、次の2つのファイルが作成されます。

1- ドメイン構成のyamlモデル。
2- アプリケーション・バイナリが保存されているアーカイブ。

yamlモデルを作成したら、Kubernetesの要件を満たすように、構成をカスタマイズして検証できます。WDT Create Domain Toolの呼出しにはドメインのyamlモデルとアーカイブが必要となり、アプリケーションがデプロイされたドメイン・ホームが作成されます。WDTによってドメイン・ホームとアプリケーションがデプロイされたDockerイメージを作成するサンプルについては、WDT GitHubプロジェクトを参照できます。

Oracle WebLogic Image Tool

Oracle WebLogic Image Toolはオープン・ソース・ツールで、これにより、ユーザー独自のカスタマイズされたイメージを含め、Oracle WebLogic Server Dockerイメージのビルド、パッチ適用、更新を自動化できます。このツールは、CI/CDプロセスでスクリプトを作成して使用可能です。Oracle WebLogic Image ToolのGitHubプロジェクト（https://github.com/oracle/weblogic-image-tool）を参照してください。

このツールには、主に4つのユースケースがあります。

1. ユーザーが以下を選択することのできる、カスタマイズされたOracle WebLogic Server Dockerイメージを作成する。
    a. OSのベース・イメージ（Oracle Linux 7.5など）。
    b. Javaのバージョン（8u202など）。
    c. Oracle WebLogic ServerまたはOracle Fusion Middleware Infrastructure（Oracle FMW Infrastructure）のインストーラ・バージョン（12.2.1.3など）。
    d. 特定のパッチ・セット更新（PSU）。
    e. 1つ以上の暫定または"個別"パッチ。
2. WebLogicのベース・インストール・イメージにパッチ適用する。
3. WebLogic Deploy Toolモデルを使用して、Oracle WebLogic ServerまたはOracle FMW Infrastructureのドメイン・イメージにパッチを適用およびビルドする。
4. ドメイン構成にパッチを適用および更新するか、既存のイメージに新しいアプリケーションをデプロイする。

Oracle WebLogic Image Toolを使用することにより、DockerおよびKubernetesで実行されているOracle WebLogic Serverのアプリケーションすべてにパッチを適用および更新する場合の自動化プロセスに、上記のユースケースを取り込むことができます。オラクルでは、セキュリティ上の理由により、Oracle WebLogic Serverのバイナリ、最新のJava更新プログラム、最新のOSに、四半期ごとのPatch Set Update（PSU）を継続的に適用するよう推奨しています。Image Toolを使用すると、この更新プロセスが簡素化されます。

Oracle WebLogic Image Toolでは、パッチを指定してダウンロードするためのREST APIを提供するMy Oracle Support（MOS）内に組み込まれている重要な新機能を利用します。Image Toolは、このREST APIを使用してユーザーが指定するすべての個別パッチとPSUをMOSから自動的にダウンロードします。このツールでは、MOS APIを呼出して、パッチの競合の有無をチェックします。ユーザーは、必要なサポート・エンタイトルメントと一緒にMOS資格証明を入力し、Oracle WebLogic ServerとJavaのインストーラを手動でダウンロードしてからこのツールを呼び出す必要がります。パッチとインストーラは、複数回ダウンロードしなくてもよいようにキャッシュされます。

Oracle WebLogic Image Toolは、Dockerのベスト・プラクティスに従って推奨のレイヤー化イメージでのイメージを自動的にビルドし、クリーンアップを実行してイメージ・サイズを最小化します。このツールにより確実に、イメージがパッチ適用可能な状態を維持し、標準と、公開して文書化されたOracleツールとAPIのみを使用するようになります。

YouTubeビデオで、Oracle WebLogic Image Toolを使用してカスタマイズされたOracle WebLogic Serverのインストール・イメージを作成する方法のデモが公開されています。KubernetesにOracle WebLogic Serverドメインをデプロイする際にOracle WebLogic Image Toolを使用してCI/CDプロセスを自動化する方法については、オラクルのドキュメントを、Jenkinsを使用した場合についてはYouTubeビデオのデモを参照してください。

Oracle WebLogic Monitoring Exporter

Oracle WebLogic Monitoring Exporterは、Prometheusなどのモニタリング・ツールによって読み取りおよび収集でき、Grafanaダッシュボードに表示できるOracle WebLogic Serverのメトリックを表示します。Oracle WebLogic Monitoring Exporterは、オープン・ソースとしてこちらから入手できます。

Oracle WebLogic Serverは、豊富なメトリック・セットと、サーバー、クラスタ、アプリケーション、およびOracle WebLogic Serverドメインで稼働している他のリソースに関するパフォーマンスおよび診断の詳細なデータを含む、ランタイム状態情報を生成します。Oracle WebLogic Monitoring Exporterにより管理者は、Kubernetes環境の監視に一般に使用されるツールであるPrometheusとGrafanaを使用して、このデータを容易に監視することができます。

図3：KubernetesおよびGrafanaダッシュボードのOracle WebLogic Server

PrometheusにエクスポートするOracle WebLogic Serverのランタイム・メトリックを使用し、そのメトリックに基づいてPrometheusでルールを設定することにより、Oracle WebLogic Serverクラスタを自動スケーリングできます。ルールに適合する場合、PrometheusはWebLogic Kubernetes Operatorを呼び出してOracle WebLogic Serverクラスタをスケーリングします。

Oracle WebLogic Monitoring Exporterの設計および実装について詳しくは、Exporting Metrics from WebLogic Serverを参照してください。PrometheusおよびGrafanaを使用したKubernetes監視について詳しくは、KubernetesのUsing Prometheus and Grafana to Monitor WebLogic Serverを参照してください。Prometheusダッシュボードと標準構成のGrafanaダッシュボードによりKubernetes内で稼働しているOracle WebLogic Serverドメインの監視方法を示すエンド・ツー・エンドのサンプルを試してください。このサンプルでは、Prometheusでのアラート・ルールの設定方法も示されます。アラート条件が満たされると、Prometheusサーバーはアラートを生成し、それにより、電子メール、Slack、webhooksなどのさまざまなレシーバに通知を送信できます。

Oracle WebLogic Logging Exporter

Oracle WebLogic Logging Exporterは、サーバー・インスタンスからElastic StackにOracle WebLogic Serverのログを直接エクスポートできます。ログは、分析してからKibanaダッシュボードに表示できます。Oracle WebLogic Logging Exporterのコードは、サンプルと一緒にGitHubプロジェクトで見つけることができます。

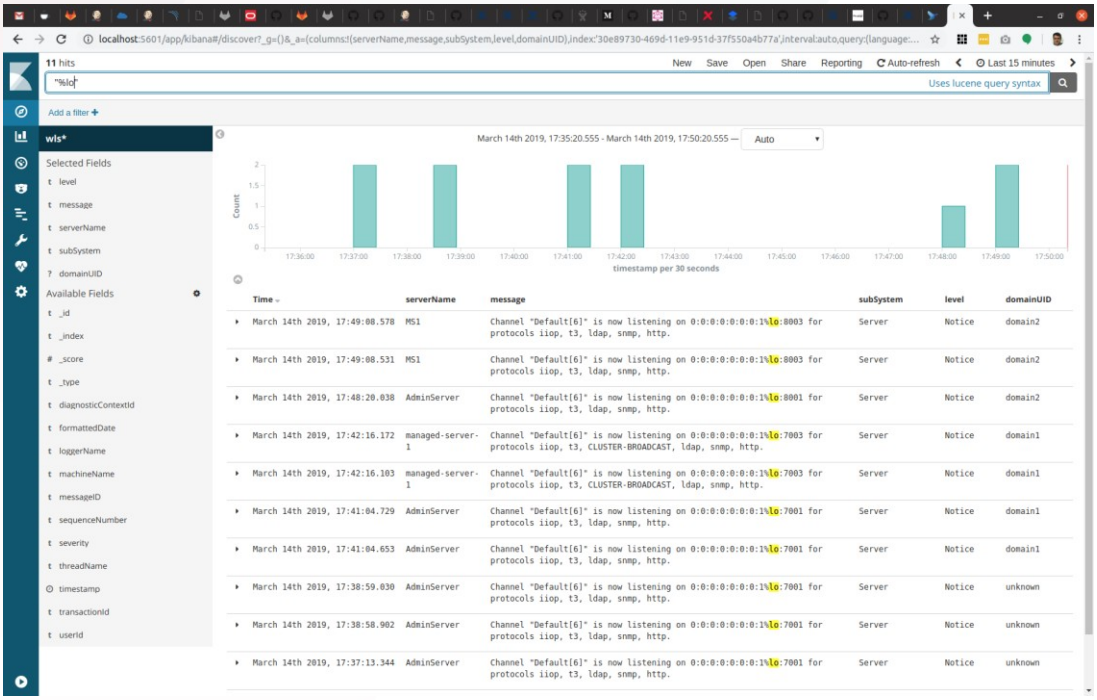

図4：KibanaでのOracle WebLogic Serverのログ

まとめると、上記のツールにより、Kubernetesクラスタ内で稼働するOracle WebLogic Serverアプリケーションのデプロイ・プロセスが簡素化されます。それらのツールにより、今日のクラウド・ネイティブ・インフラストラクチャの新機能を利用してアプリケーションを管理し、進化させ、同じ環境で実行されている他のアプリケーションと統合することができます。

Oracle Exalogic Elastic Cloud上のOracle WebLogic ServerデプロイメントのOracle Private Cloud Applianceへの移行

Oracle Private Cloud Applianceは、プライベート・クラウドを迅速にデプロイすることができるように設計されたオラクルのエンジニアド・システムです。コンピューティング・リソース、ネットワーク・ハードウェア、ストレージ・プロバイダ、オペレーティング・システム、アプリケーションが連携動作するように設計されており、単一ユニットとして管理および操作します。この点でOracle Private Cloud Applianceは、Oracle Exalogic Elastic Cloudシステムに似ています。

それに加えて、Oracle Private Cloud Applianceは、オラクルによって全面的にサポートされる環境で稼働中のKubernetesに対応できます。Oracle Linux Cloud Native Environmentを使用して、Oracle Private Cloud Appliance上にKubernetesクラスタを作成できます。このプロセスについては、ホワイト・ペーパー『Deploy Application Containers on Oracle Private Cloud Appliance/Private Cloud at Customer』で説明されています。Kubernetesが稼働するOracle Private Cloud Applianceは、コンテナベースのKubernetesプラットフォームへの移行を考え、Oracle WebLogic Serverワークロードを実行しているOracle Elastic Cloudのお客様にとって、魅力的な移行ターゲットです。

一般に、アプリケーションは変更なしで移行可能です。お客様は、そのような移行を計画する場合、次のような環境間の一般的な比較項目について考慮する必要があります。これについては、このホワイト・ペーパーの以降のセクションでさらに詳しく説明します。

- Oracle WebLogic Serverのバージョン10.3.6および12.1.3は、ライフサイクルの終わりが間近に迫っています。更新されたRESTサポート、JSON処理、自動スケーリング、REST管理などのOracle WebLogic Server 12.2.1.Xの新機能により、クラウド・システムとの統合が行いやすくなっています。そのためKubernetesサポートの焦点は、12.2.1.X（以降のバージョン）のサポートに当てられています。以前のバージョンを使用しているお客様は、移行プロセスの一環として12.2.1.3以降に移行する必要があります。WDTは一般に、アプリケーションで無効になったAPIが使用されているのでない限り、そのようなアプリケーションとドメイン構成を"そのまま"移行できます。Oracle WebLogic Serverのバージョンごとのアップグレードについて詳しくは、アップグレードのドキュメントを参照してください。
- WDTとOracle WebLogic Image Toolは、必要なDockerイメージの作成をサポートします。
- 移行により、アプリケーションが使用する基盤のコンピューティング・インフラストラクチャが変更されます。移行先環境の管理対象サーバーは、WebLogic Kubernetes Operatorのカスタム・リソースにより定義されるKubernetesのPod、ノード、クラスタで稼働します。
- コンソールやWLSTなどのOracle WebLogic Serverの管理ツールは、移行後の環境でもサポートされます。ただしユーザーは、ユーザーのOracle WebLogic Serverアプリケーションの管理を進める場合のネイティブKubernetesツールの使用法を考慮する必要があります。
- Oracle Traffic DirectorはOracle Public Cloud Applianceへの移行用としてサポートされていますが、TraefikやVoyagerなどのKubernetesのネイティブ・ロードバランサのほうがKubernetesにより適切であり、Oracle Traffic Directorの代替ロードバランサとして推奨されています。
- Oracle Exadataシステムで稼働するOracle RACクラスタを含むデータベースへのアクセスなど、HTTPおよびT3プロトコルを介した外部システムへのアクセスがサポートされています。SDPプロトコルはOracle Private Cloud Applianceではサポートされていないため、Oracle Exalogic Cloudシステム上で実行されているドメイン内で使用されている既存のSDPがある場合は、削除する必要があります。
- Oracle WebLogic Server、Oracle Coherence、Oracle Application Development Frameworkは現在、WebLogic KubernetesのツールとともにKubernetesでの使用がサポートされています。追加のOracle Fusion Middleware製品のサポートも計画されています。
- Oracle Exalogic Elastic Cloud Softwareのライセンスの移行については、ライセンスのドキュメントを参照してください。

このホワイト・ペーパーの以降のセクションでは、WebLogic Kubernetesのツールを使用して、Oracle Exalogic Elastic Cloudシステム上で実行されているOracle WebLogic Serverのアプリケーションを、Oracle Private Cloud Applianceで稼働しているKubernetesクラスタに移行する方法について説明します。

環境の準備

このセクションでは、作成したKubernetesクラスタにOracle WebLogic Serverのアプリケーションをデプロイするための使用環境の準備方法について説明します。ここでは、上記で参照したOracleホワイト・ペーパーに従ってOracle Private Cloud Appliance上にKubernetesクラスタをすでに作成していることを前提としています。

最初のステップは、Kubernetesクラスタにアクセスすることです。これには、*kubectl*コマンドを使用します。*Kubectl*は、Kubernetesクラスタに対してコマンドを実行するためのコマンドライン・インタフェースです。*kubeconfig*ファイルを作成し、ローカル・ボックスから*kubectl*コマンドを実行できるようにする必要があります。ホワイト・ペーパー『Deploy application containers on Oracle Private Cloud Appliance/Private Cloud at Customer』の「Setting up Kubernetes Master Node」セクションでは、kubeconfigファイルの作成方法が説明されています。

```
$ mkdir -p $HOME/.kubesudo
$ cp -i /etc/kubernetes/admin.conf $HOME/.kube/configsudo
$ chown $(id -u):$(id -g) $HOME/.kube/config
```

ローカル・ボックスで*kubectl*コマンドを実行するには、次を実行します。

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

HelmとTillerのインストール

Helmは、Kubernetes上で実行されるアプリケーション・パッケージ・マネージャです。これにより、便利なHelmチャートを使用してアプリケーション構造を記述し、簡単なコマンドで管理することができます。

WebLogic Kubernetes Operatorプロジェクトでは、KubernetesクラスタにOperatorをインストールするためのHelmチャートを作成しました。Helmのインストールと使用方法については、Install Helm and Tillerを参照してください。

ロードバランサ、Prometheus、Grafana、Elastic Stackはすべて、Helmチャートを使用してKubernetesクラスタ内にインストールします。

WebLogic Kubernetes Operatorプロジェクトからのフィルのダウンロード

ローカル・マシンにOperatorリポジトリのクローンを作成し、このホワイト・ペーパー全体で述べられているさまざまなサンプル・ファイルにアクセスできるようにする必要があります。まず、Oracle Private Cloud ApplianceでOracle WebLogic Serverを実行するために必要なファイルを格納する、コマンドを実行するためのディレクトリを作成します。

```
$ mkdir –p ~/WLS_K8S_PCA
$ cd ~/WLS_K8S_PCA
$ git clone https://github.com/oracle/weblogic-kubernetes-operator
```

Oracle WebLogic Server Dockerイメージの作成

WebLogic Kubernetes Operatorでは、2つのOracle WebLogic Server Dockerイメージ・モデルに対応しています。1つはドメイン・ホームが永続ボリューム（PV/PVC）に保管されるモデル、もう1つはドメイン・ホームがDockerイメージ内に保管されるモデルです。このホワイト・ペーパーでは、Dockerイメージ内にOracle WebLogic Server 12.2.1.3ドメインをデプロイします。イメージは、Oracle Linuxスリムのベース・イメージ、JDK、Oracle WebLogic Serverバイナリによってレイヤー化され、Operator、ドメイン・ホーム、アプリケーションで実行するために必要なパッチが適用されます。ここでは、Oracle WebLogic Image ToolとWebLogic Deploy Tooling（WDT）との間の統合を使用してイメージをビルドします。この統合によって操作を非常にシンプルにして、Oracle WebLogic Serverイメージを1つのコマンドラインで作成します。

最初のステップは、WDTを使用してOracle Exalogic Elastic Cloudでのドメイン構成をイントロスペクトすることです。WDT Discover Domain Toolにはブートストラップ・メカニズムを装備しており、既存のドメインを調べて、そこから構成およびバイナリの情報を収集することにより、モデルを作成してファイルをアーカイブします。このツールによって作成されるモデル・ファイルは、WDT Create Domain ToolまたはDeploy Applications Toolから直接使用することはできません。これは、WDT Discover Domain Toolでは、既存のドメインからパスワードを検出しないからです。代わりに、検出したパスワードとして--FIX ME--プレースホルダを使用します。ドメイン・ユーザーも検出可能ではないため、domainInfoセクションで、Create Domain Toolが必要とするAdminUserNameフィールドとAdminPasswordフィールドは作成されません。ツールでは、他のツールのいずれかを実行する場合の要件に適合するように、生成されたモデルとアーカイブ・ファイルを編集する場合の開始点が指定されます。

WDT Discover Domain Toolを実行するには、Oracle Exalogic Elastic Cloud上でのドメインの場所と、アーカイブ・ファイルのファイル名を入力します。たとえば次のとおりです。

```
$ weblogic-deploy¥bin¥discoverDomain.cmd -oracle_home c:¥wls12213 -
domain_home domains¥DemoDomain -archive_file archive.zip -model_file
simple-topology.yaml
```

archive.zipには、ドメインにデプロイしようとするアプリケーションが格納されています。WDTによって生成されたarchive.zipを解凍した場合は、特定のディレクトリ*wlsdeploy*にアプリケーションが保存されています。

たとえば、testwebapp.warと呼ばれるアプリケーションをデプロイする場合のarchive.zipディレクトリ構造は次のようになります。

```
Archive:          archive.zip
      creating:META-INF/
    inflating:META-INF/MANIFEST.MF
      creating: wlsdeploy/
      creating: wlsdeploy/applications/
    inflating: wlsdeploy/applications/testwebapp.war
```

testwebapp.warアプリケーションのデプロイメントを記述するWDTイントロスペクションによって生成されたドメインのyamlモデルは次のようになります：

```
appDeployments:
        Application:
                'test-webapp':
                        SourcePath:
'wlsdeploy/applications/testwebapp.war'
                        Target: '@@PROP:CLUSTER_NAME@@'
                        ModuleType: war
                        StagingMode: nostage
                        PlanStagingMode: nostage
```

@@PROP:CLUSTER_NAME@@の値はdomain.propertiesファイルに格納されており、エクスポータとアプリケーションのデプロイ先となるOracle WebLogic Serverクラスタを意味します。

このホワイト・ペーパーでは、Image Toolの*create*コマンドを使用してOracle WebLogic Serverイメージをビルドします。イメージの作成準備として、次の手順に従って操作する必要があります。

1- Image Toolの解凍先ディレクトリを作成します。
   ```
   $ mkdir –p          ~/ImageTool
   $ cd ~/ImageTool
   ```
2- Image Toolがパッチにアクセスできるようにするため、My Oracle Support用の資格証明が必要になります。
3- リリース（ https://github.com/oracle/weblogic-image-tool/releases ）から、imagetool-1.1.1.zipをダウンロードします。
4- imagetool-1.1.1.zipを解凍します。
5- cd imagetool-1.1.0/bin
6- source setup.sh
7- 次の3つの環境変数を設定する必要があります。

   ```
   export WLSIMG_CACHEDIR="/path/to/cachedit"
   export WLSIMG_BLDDIR="/path/to/dir"
   export MYPWD="My Oracle Support password"
   ```

   ダウンロードしたOracle WebLogic ServerとJDKのインストーラは、$WLSIMG_CACHEDIRにコピーされます。Image Toolにより、イメージのビルドに使用するファイルすべてがそのビルド・ディレクトリに格納されます。
   Image Toolでは、ユーザーのMy Oracle Support資格証明を使用して、パッチを自動的にダウンロードします。パスワードは環境変数またはファイルに保存することができます。特にこの場合は、MYPWDと呼ばれる環境変数に保存されます。
8- JDKとOracle WebLogic Serverのインストーラをダウンロードします。

9- Oracle WebLogic ServerとJDKのインストーラを$WLSIMG_CACHEDIRにコピーします。
10- JDKをキャッシュ・ディレクトリにコピーしたら、imagetool cacheを呼び出します。

```
$ imagetool cache addInstaller ¥
--type jdk --version 8u221 –path ¥                    $WLSIMG_CACHEDIR/jdk-8u221-
linux-x64.tar.gz
```

11- キャッシュに追加する必要があるOracle WegLogic Serverのインストーラをキャッシュ・ディレクトリにコピーしたら、imagetool cacheを呼び出します。

```
$ imagetool cache addInstaller --type wls ¥
--version 12.2.1.3.0 –path ¥
$WLSIMG_CACHEDIR/fmw_12.2.1.3.0_wls_quick_Disk1_1of1.zip
```

12- WebLogic Kubernetes Operatorによって管理されるKubernetesにOracle WebLogic Serverドメインをデプロイするため、Oracle WebLogic Server 12.2.1.3では29135930と27117282の2つのパッチを必要とします。Image Toolの*create*コマンドで指定可能なオプションは、JDKのバージョン、WebLogicのバージョン、適用するパッチ、MOS資格証明、WDTドメインのyamlモデル、そのモデルに渡されるプロパティ、アプリケーションが保存されているアーカイブです。Image Toolのコマンドで使用するオプションの説明が必要な場合は、imagetool create –hを実行します。

*注：この例では、ドメイン・ホームに*sample-domain1*という名前のサンプル・ドメインを含める必要があります（例：*/u01/oracle/user_projects/domains/sample- domain）。

```
$ imagetool create --tag domain-home-in-image:12.2.1.3 ¥
--version 12.2.1.3.0 --user / <MOS user>                    ¥
--patches=29135930_12.2.1.3.0,27117282_12.2.1.3.0 ¥
-- passwordEnv MYPWD --jdkVersion=8u221                     ¥
--wdtArchive=<Directory>/archive.zip                  ¥
--wdtModel=<Directory>/simple-topology.yaml ¥
--wdtDomainHome=/u01/oracle/user_projects/domains/sample-domain1
¥
--wdtVariables=<Directory>/domain.properties ¥
--wdtVersion=1.1.1
```

Oracle Private Cloud Appliance内のワーカー・ノードへのイメージのプッシュ
Image Toolによってイメージが作成されたら、Kubernetesクラスタの各ノードでイメージを使用できるようにする必要があります。Oracle Private Cloud ApplianceのKubernetesクラスタには2つのワーカー・ノードがあり、これらは、"kubectl get nodes"コマンドを実行することによって入手できます。

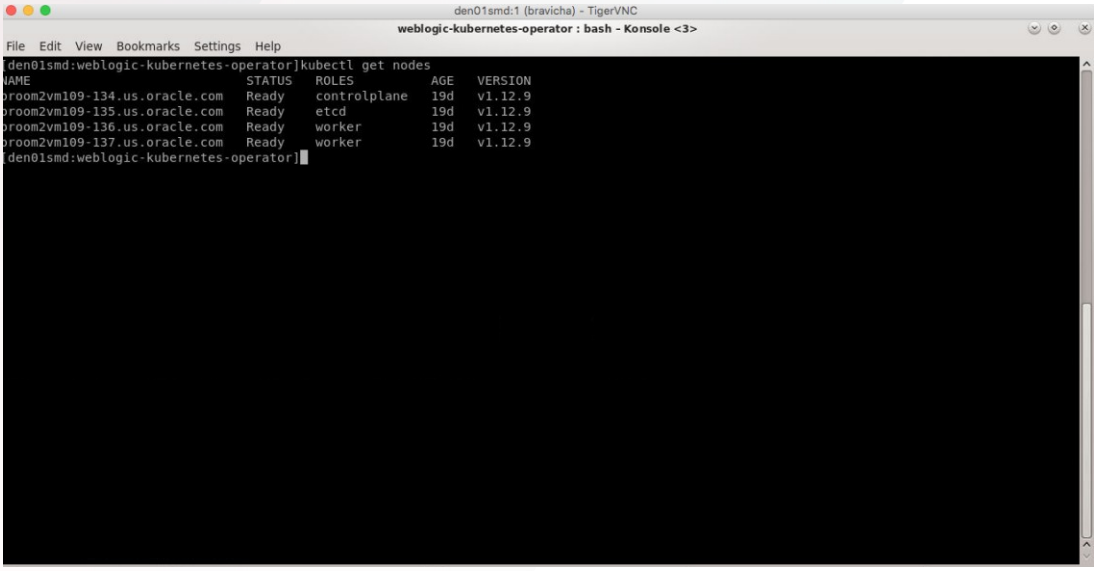

図5：Oracle Private Cloud ApplianceマシンのKubernetesノード、2つのワーカー・ノードローカル・ボックスからtarファイルにイメージを保存します。

$ docker save domain-home-in-image:12.2.1.3 -o domain-home-in-image.tar

Oracle Private Cloud Applianceの各VMにtar化されたイメージをコピーします。

$ scp -i ~/.ssh.id_rsa domain-home-in-image.tar root@broom2vm109-137.us.oracle.com:/tmp/domain-home-in-image.tar

$ scp -i ~/.ssh.id_rsa domain-home-in-image.tar root@broom2vm109-136.us.oracle.com:/tmp/domain-home-in-image.tar

次に、ワーカー・ノードのそれぞれにssh接続し、そのイメージを各ノードにロードします。

$ docker load -i /tmp/domain-home-in-image.tar

Operatorのデプロイ

WebLogic Kubernetes Operatorプロジェクトでは、HelmチャートによってOperatorをデプロイします。「環境の準備」セクションでは、ローカル・ボックスにWebLogic Kubernetes Operator GitHubプロジェクトのクローンを作成しました。このため、プロジェクトのクローンが作成されたディレクトリへの移動が必要になります。

$ cd ~/WLS_K8S_PCA/weblogic-kubernetes-operator

Operatorを実行する名前空間を作成します。

$ kubectl create namespace sample-weblogic-operator-ns

Helmを実行してその名前空間にOperatorをインストールします。

```
$ helm install kubernetes/charts/weblogic-operator ¥
    --name sample-weblogic-operator ¥
    --namespace sample-weblogic-operator-ns ¥
    --set image=oracle/weblogic-kubernetes-operator:2.2.1¥
    --set serviceAccount=sample-weblogic-operator-sa ¥
```

```
    --set "domainNamespaces={}" ¥
    --wait
```

Operatorが実行されていることを確認します。

```
$ kubectl get pods -n sample-weblogic-operator-ns
```

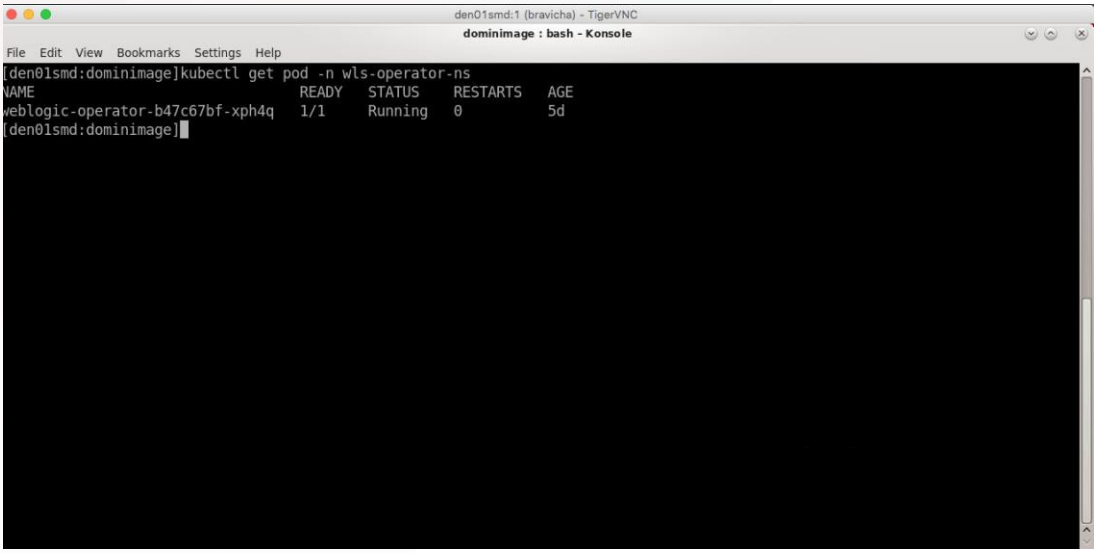

図6：Kubernetesクラスタ内で実行されるWebLogic Kubernetes Operator

ロードバランサのデプロイ

KubernetesのロードバランサはIngressをサポートします。Ingressは、ルールのコレクションと、外部HTTP（S）トラフィックを内部サービスにルーティングする構成をカプセル化するKubernetesリソースです。オラクルでは、Kubernetes上のOracle WebLogic ServerでのTraefik、Voyager、Apacheの動作を保証し、Helmチャートを使用してこれらのロードバランサをインストールする方法を示すサンプルを用意しています。Kubernetes上で稼働する場合、Oracle WebLogic Serverはまた、Ingressに対応可能な他のロードバランサをサポートし、これらのKubernetesクラスタでデフォルトでサポートされます。

Kubernetesでは、クラスタ全体にまたがる適応型ロードバランシングの場合、Oracle Traffic Director（Oracle TD）またはOracle HTTP Server（Oracle HS）のロードバランサを使用する必要はありません。WebLogic Kubernetes Operatorは、Oracle WebLogic Serverクラスタで稼働している管理対象サーバーのエンドポイントが関係するクラスタ・サービスを作成します。Oracle WebLogic Serverクラスタがスケーリングされ、アプリケーションでトラフィックの受信準備が整うと、Operatorはそのクラスタ・サービスを調整し、新しい管理対象サーバー・エンドポイントを追加します。Ingressでは、新しく追加された管理対象サーバーへのトラフィックのルーティングを開始します。Kubernetesサービスによるこの種のトラフィック・ルーティングは、TraefikやVoyagerのようなIngressに対応可能なロードバランサでのみサポートされます。

この例では、Oracle Private Cloud Applianceで実行されるKubernetesクラスタにTraefikをデプロイします。

Traefikの名前空間を作成する必要があります。

```
$ kubectl create namespace traefik
```

Traefikをインストールします。

```
$ helm install stable/traefik ¥
    --name traefik-operator ¥
    --namespace traefik ¥
```

```
    --values kubernetes/samples/charts/traefik/values.yaml                ¥
    --set "kubernetes.namespaces={traefik}" ¥
    --wait
```

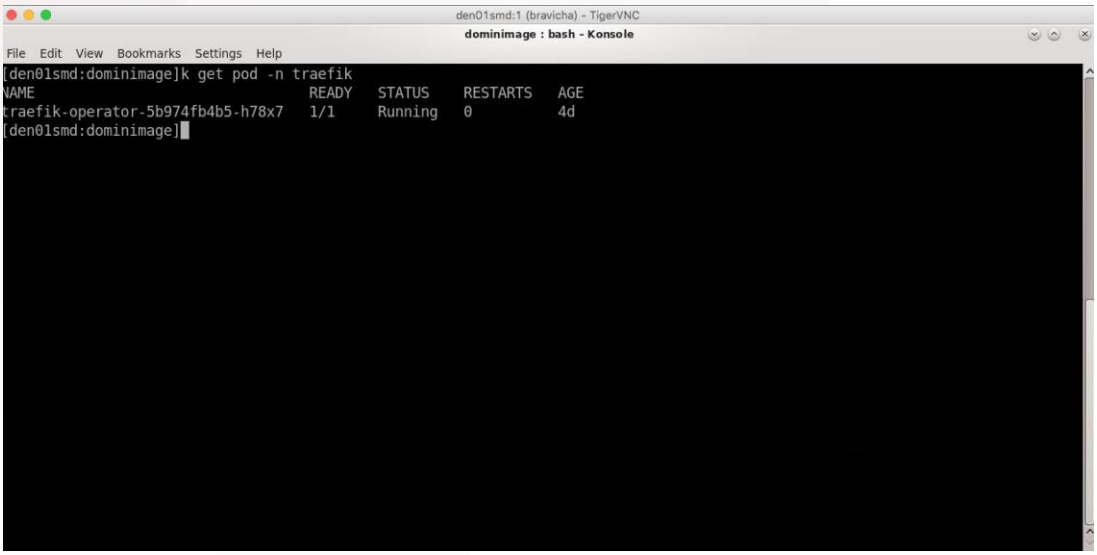

図7：KubernetesクラスタにデプロイされたTraefik Pod

ドメインのデプロイ

ドメインをデプロイするため、Kubernetes内のドメイン・オブジェクトを表し、Kubernetes APIを拡張してOracle WebLogic Serverドメインをオーケストレートする、Kubernetes Custom Resource（CR）を作成します。ドメインCRは、domain.yamlファイルを適用することによって定義および作成されます。WebLogic Kubernetes OperatorはドメインCRを監視し、実行状態をドメインCRでの定義に合致させます。たとえば、ドメインCRで2つの管理対象サーバーがクラスタで稼働するように定義すると、Operatorは、Oracle WebLogic Serverクラスタ内で稼働する2つの管理対象サーバーで2つのPodを起動します。

ドメインが実行される名前空間を作成します。

```
$ kubectl create namespace sample-domain1-ns
```

ドメインの名前空間でドメインを管理するようにOperatorを構成します。

```
$ helm upgrade ¥
    --reuse-values ¥
    --set "domainNamespaces={sample-domain1-ns}" ¥
    --wait ¥
    sample-weblogic-operator
    ¥ kubernetes/charts/weblogic-operator
```

この名前空間で作成されるIngressを管理するようにTraefikを構成します。

```
$ helm upgrade ¥
    --reuse-values ¥
    --set "kubernetes.namespaces={traefik,sample-domain1-ns}" ¥
    --wait ¥
    traefik-operator ¥
    stable/traefik
```

管理サーバーの資格証明は、資格証明を安全に保つため、Kubernetes Secretに保存されます。Kubernetes Secretを作成します。

```
$ kubernetes/samples/scripts/create-weblogic-domain-credentials/create-
weblogic-credentials.sh ¥
    -u <username> -p <password> -n sample-domain1-ns -d sample-domain1
```

domain.yamlは、ドメインのCustom Resourceを作成する前に、イメージ名、ドメインの名前空間、ドメインUIDなどのデプロイメントに関係する正しい情報に基づいて変更する必要があります。

ローカル・ディレクトリにdomain.yamlのコピーを作成します。

```
$ cp ./kubernetes/samples/scripts/create-weblogic-domain/manually-
create-domain/domain.yaml .
$ vi domain.yaml
```

domain.yamlで以下の情報を置き換えます。

```
kind:Domain
metadata:
    # ドメインの`domainUID`を使ってこれを更新します:
    name: sample-domain1
    # 実行場所のドメインの名前空間を使ってこれを更新します:
    namespace: sample-domain1-ns
    labels:
        weblogic.resourceVersion: domain-v2
        # ドメインの`domainUID`を使ってこれを更新します:
        weblogic.domainUID: sample-domain1

spec:
    # このパラメータは（コンテナから見た）Oracle WebLogic Serverのドメイン・ホームの場所を示しています。
# マウントされたボリュームまたはネットワーク・ストレージ内のイメージ自体に格納されています。
domainHome: /u01/oracle/user_projects/domains/sample-domain1

    # ドメイン・ホームがDockerイメージ内にある場合は、これを`true`に、そうではない場合は、`false`に設定してください：
domainHomeInImage: true

# ドメインの実行に使用されるDockerイメージの名前でこれを更新してください：
image: "domain-home-in-image:12.2.1.3"

# Oracle WebLogic Server の管理者資格証明を含むSecretを特定します（このファイルの最後にSecretの作成方法の例を示します）
webLogicCredentialsSecret:
    # ユーザー名sample-domain1-weblogic-credentials
    を含むSecretの名前でこれを更新してください

adminServer:
        # serverStartStateに適している値は"RUNNING"または"ADMIN"です
        # "RUNNING"は一覧表示されたサーバーが"RUNNING"モードで起動されることを意味しています

        # "ADMIN"は一覧表示されたサーバーが"ADMIN"モードで起動されることを意味しています
```

```
      serverStartState:"RUNNING"
      adminService:
        channels:
          # 管理サーバーのデフォルト・チャネルで使用するNodePortを設定するにはこれを更新します（これで管理コンソールが使用可能になります）：
          - channelName: default
            nodePort:30701
```

管理コンソールにアクセスするためには、管理サーバーのノード・ポートを公開する必要があります。Operatorは、CRでそのポートを定義してNodePortサービスを作成します。この値は、30000～32767の範囲のポートに変更できます。

ドメインCRを作成します。

```
$ kubectl apply –f domain.yaml
```

ドメインCRが作成されたことを確認します。

```
$ kubectl describe domain –n sample-domain1-ns
```

Operatorは、ドメインCRが作成されたことを認識するとすぐに、domain.yamlでの定義に従ってドメインを立ち上げます。ドメインが起動したことを確認します。

```
$ kubectl get pods –n sample-domain1-ns –o wide
```

管理サーバーと2つの管理対象サーバーが稼働しているのを確認します。

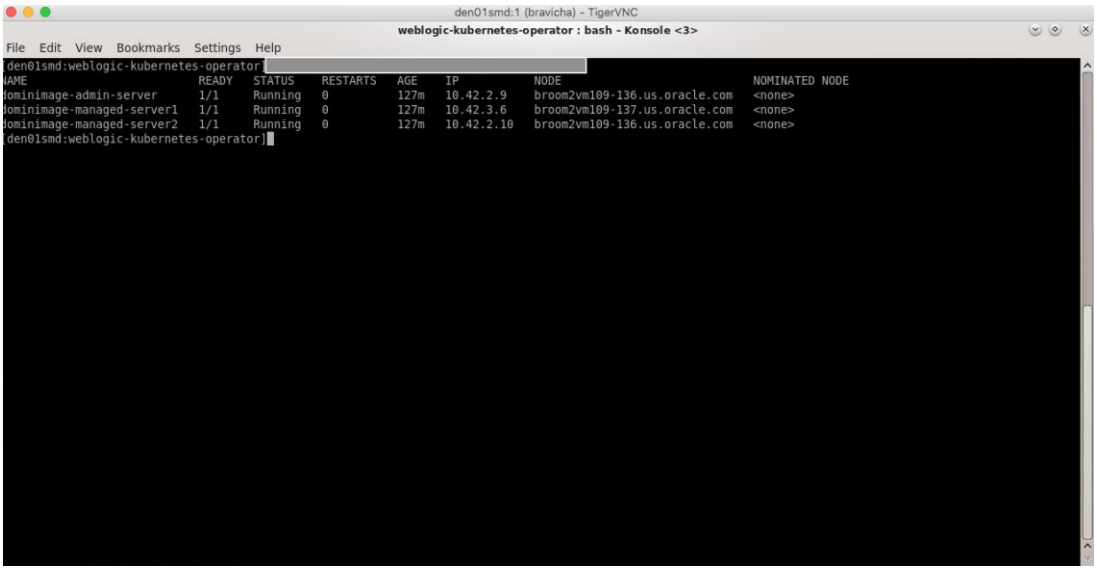

図8：Operatorによって管理されるKubernetes上で稼働するOracle WebLogic Serverドメイン

サンプルのHelmチャートを使用することにより、ドメイン名前空間でドメインのIngressを作成します。

```
$ helm install kubernetes/samples/charts/ingress-per-domain ¥
    --name sample-domain1-ingress ¥
    --namespace sample-domain1-ns ¥
    --set wlsDomain.domainUID=sample-domain1 ¥
    --set traefik.hostname=sample-domain1.org
```

管理コンソールを起動するには、ブラウザに移動し、CRで定義した管理サーバーのホストとNodePortを使用し、URLを入力します。

http:// ${myhost}:<Node Port>/console

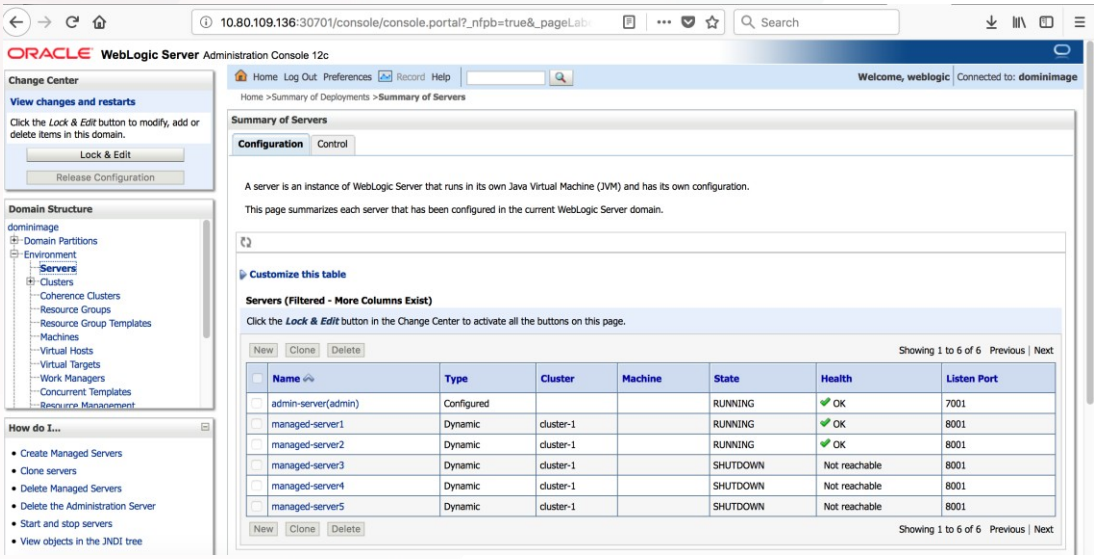

図9：Oracle WebLogic Serverの管理コンソール、サーバー・ページ

Oracle WebLogic Serverクラスタのスケーリング

Oracle WebLogic Serverクラスタは、以下の3つの方法で拡張/縮小できます。

1- domain.yamlを編集し、レプリカの数を変更します。たとえば次のように指定します。
clusters:
clusterName: cluster-1
serverStartState:"RUNNING"
replicas:3

次に、以下を実行します。

$ kubectl apply –f domain.yaml

Operatorは、CRでの変更を認識するとすぐ、稼働している管理対象サーバーで新規Podを起動します。

2- WebLogic診断フレームワーク（WLDF）でルールを設定することもできます。ルールの条件を満たすと、Operatorはスケーリング・アクションを実行するよう指令されます。Operatorは、そこで稼働している管理対象サーバーで新規Podを起動します。ブログ「Automatic Scaling of WebLogic Clusters in Kubernetes」を参照してください。

3- Monitoring Exporterにより、Oracle WebLogic Serverのメトリックを Prometheusにエクスポートできます。Prometheusではルールを設定でき、ルールの条件を満たすと、PrometheusはOperatorを呼び出してOracle WebLogic Serverクラスタをスケーリングします。

ライフサイクル管理/操作

WebLogic Kubernetes Operatorは、LivenessとReadinessの2つのプローブを使用します。Livenessプローブにより WebLogic Kubernetes Operatorは、Podで稼働しているOracle WebLogic Serverインスタンに異常が発生したことを認識します。OperatorはPodを再起動し、確実にサーバーの可用性が高く維持されるようにします。Readinessプローブによりoperatorは、アプリケーションが、単に起動しているというだけでなく、トラフィックの受信準備もできていることを認識します。Readinessプローブによってアプリケーションの”準備ができている"ことが示されると、Operatorは管理対象サーバーのエンドポイントでのKubernetesクラスタ・サービスに変更を加え、Ingressがその準備ができているアプリケーションへのトラフィックのルーティングを開始します。

WebLogic Kubernetes Operatorは、サーバー/クラスタ/ドメインの起動/停止/再起動などのユーザーが開始するライフサイクル操作と、ドメインのローリング再起動の開始にも対応します。documentation about lifecycle operations を参照してください。

これらのライフサイクル操作はCRを介して実行されます。たとえば、Operatorにドメインのシャットダウンを指示するには、domain.yamlを編集してserverStartPolicyをNEVERに設定してから、kubectl –f apply domain.yamlを実行します。Operatorは、CRでserverStartPolicyが変更されたのを認識すると、ドメインをシャットダウンします。

ローリング再起動の場合は、restartVersionを何らかのstring versionに設定すると、Operatorがドメインのローリング再起動を開始します。string versionに基づいてOperatorは、ドメイン内のどのサーバーがすでに再起動しており、まだ再起動する必要があるサーバーはどれかを認識します。

```
# serverStartPolicyに適している値は"NEVER"、"IF_NEEDED"、"ADMIN_ONLY"です
    # これにより、ドメインの検出時に、OperatorがどのOracle WebLogic Serversドメインを起動するかを決定します
    # - "NEVER"はドメイン内のいずれのサーバーも起動しません
    # - "ADMIN_ONLY"は管理サーバーのみを起動します（管理対象サーバーは起動しません）
    # - "IF_NEEDED"は非クラスタ化サーバー、管理サーバー、クラスタ化サーバーを、レプリカ数に達するまですべて起動します

    serverStartPolicy:"NEVER"
    restartVersion:"RollDomain1"
```

Kubernetes内のOracle WebLogic Serverドメインの監視

Oracle WebLogic Serverドメインは、Oracle WebLogic Monitoring Exporterを使用し、Prometheusによって読取り可能で、Grafanaダッシュボードで表示可能な形式ですべてのメトリックをエクスポートすることによって、Kubernetesで監視できます。PrometheusとGrafanaをデプロイしてドメインのメトリックを表示する方法については、エンド・ツー・エンドのサンプルを参照してください。

Oracle WebLogic Monitoring Exporterは、Oracle WebLogic Serverクラスタのサーバーにデプロイされるwebアプリケーションです。このエクスポータ、wls-exporter.war は、Monitoring ExporterのGitHubプロジェクト・リリースからダウンロードしてください。Oracle WebLogic Monitoring ExporterでのPrometheusおよびGrafanaのセットアップおよび実行方法について詳しくは、ブログ（Using Prometheus and Grafana to Monitor WebLogic Server on Kubernetes）を参照してください。

wls-exporterというWebアプリケーションは、domain-home-in-image:12.2.1.3イメージに作成されたドメインにデプロイする必要があります。ここでは、Image Tool、およびWDTとの統合を使用してエクスポータをデプロイします。

エクスポータを保存したアーカイブを作成し、アプリケーション・デプロイメント構成でyamlモデルを作成します。wls-exporter.warは、次のアーカイブ・ディレクトリ形式に従う必要があります。

wlsdeploy/applications

たとえば、wls-exporter.warをデプロイする場合のarchive.zipディレクトリ構造は次のようになります。

```
Archive:          archive.zip
      creating:META-INF/
    inflating:META-INF/MANIFEST.MF
      creating: wlsdeploy/
      creating: wlsdeploy/applications/
    inflating: wlsdeploy/applications/wls-exporter.war
```

デプロイ対象のエクスポータについて記述したメタデータ・モデルを作成します。この例の場合、メタデータ・モデル・ファイルはsimple-topology.yamlと呼ばれ、wls-exporter.warを指定します。

```
appDeployments:Application:
                'wls-exporter':
                        SourcePath: 'wlsdeploy/applications/wls-exporter.war'
                        Target: '@@PROP:CLUSTER_NAME@@'
                        ModuleType: war
                        StagingMode: nostage
                        PlanStagingMode: nostage
```

@@PROP:CLUSTER_NAME@@の値はdomain.propertiesファイルに格納され、エクスポータとアプリケーションのデプロイ先となるOracle WebLogic Serverクラスタを参照します。

Image Toolのupdateコマンドで指定可能なオプションは、作成されるイメージの名前、更新されるイメージ、上記で作成されたWDTドメインのyamlモデル、そのモデルに渡されるプロパティ、アプリケーションが保存されているアーカイブ、WDTの更新操作です。imagetoolコマンドで使用するオプションの説明が必要な場合は、次のコマンドを実行します。

```
imagetool update –h
```

注：この例において、ドメイン・ホームは次のようにする必要があります。
/u01/oracle/user_projects/domains/sample-domain1

```
$ imagetool update --tag domain-home-in-image-monitor:12.2.1.3 ¥
--fromImage=domain-home-in-image:12.2.1.3 ¥
--wdtArchive=<Directory>/archive.zip ¥
--wdtModel=<Directory>/simple-topology.yaml ¥
--wdtDomainHome=/u01/oracle/user_projects/domains/sample-domain1 ¥
--wdtVariables=<Directory>/domain.properties ¥
--wdtOperation=DEPLOY --wdtVersion=1.1.1
```

更新されたイメージは、Oracle Private Cloud Appliance内のKubernetesクラスタの各ノードにプッシュする必要があります。

ローカル・ボックスからtarファイルにイメージを保存します。

```
$ docker save domain-home-in-image-monitor:12.2.1.3 -o domain-home-in-
image-monitor.tar
```

Oracle Private Cloud Applianceの各VMにtar化されたイメージをコピーします。

$ scp -i ~/.ssh.id_rsa domain-home-in-image-monitor.tar root@broom2vm109-137.us.oracle.com:/tmp/domain-home-in- image.tar

$ scp -i ~/.ssh.id_rsa domain-home-in-image-monitor.tar root@broom2vm109-136.us.oracle.com:/tmp/domain-home-in- image.tar

次に、ワーカー・ノードのそれぞれにssh接続し、そのイメージを各ノードにロードします。

$ docker load -i /tmp/domain-home-in-image-monitor.tar

imagetoolを編集して新しく更新されたイメージを追加します。

ドメインの実行に使用されるDockerイメージの名前でこれを更新してください：
image: "domain-home-in-image-monitor:12.2.1.3"

domain.yamlを適用してCRを更新します。

$ kubectl apply –f domain.yaml

Operatorは、新しく更新されたイメージ名によってCRでの変更を認識すると、Oracle WebLogic Serverドメインをローリングし、新規イメージに基づいて新しいPodを起動します。これでエクスポータがデプロイされます。次のステップは、PrometheusとGrafanaのデプロイです。

Prometheusのデプロイ

PrometheusとGrafanaの両Podとサービスを実行する名前空間を"monitoring”という名前で作成します。

$ kubectl create namespace monitoring

Prometheusは、Helmチャートを使用してインストールされます。サンプルSetting up Prometheusの指示に従ってPrometheusをデプロイしてください。

新しいOracle WebLogic Serverメトリックを取得するためのPrometheus構成の更新は、動的に実行することができます。PrometheusのHelmチャートを使用してインストールする場合には、Prometheusサーバー用とConfigMapでの変更の検出用の2つのコンテナがPrometheusサーバーPodで実行されています。変更するには、PrometheusサーバーのConfigMapに最新の構成を適用してください。サンプルには、Prometheusを更新して新しいメトリックを取得する方法が示されています。

Grafanaのデプロイ
GrafanaはHelmチャートを使用してインストールします。サンプルSetting up Grafanaの指示に従ってGrafanaをデプロイしてください。

デプロイしたままのGrafanaのダッシュボードで、データソースに関する監視情報をドリルダウンすることができます。現在容量、有効接続数、合計接続数などの情報を追跡し、グラフ形式で表示できます。

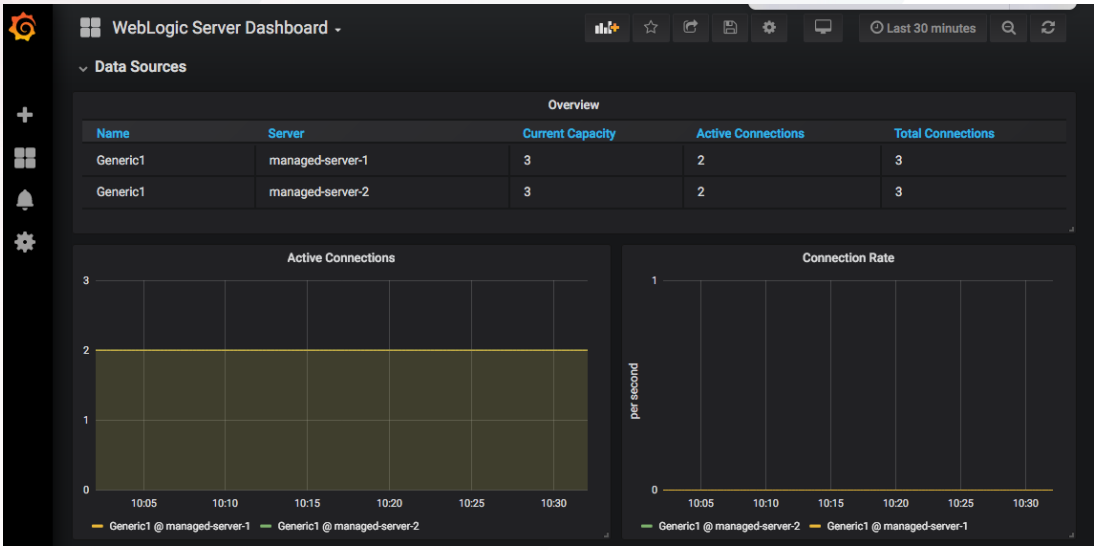

図10：Oracle WebLogic Serverのデータソースを表示したGrafanaのダッシュボード

Prometheusアラート・マネージャ

Prometheusは、アラート・ルールを設定するように構成可能です。アラート条件が満たされると、Prometheusサーバーはアラートを生成し、電子メール、Slack、webhookなどのさまざまなレシーバに通知を送信できます。また、WebLogic Kubernetes Operatorに対して、Oracle WebLogic Serverクラスタをスケーリングするように通知することもできます。サンプルには、Prometheusでアラート・マネージャをデプロイおよび構成する方法と、Webhookのセットアップ方法またはアラートの生成方法が示されています。

Oracle WebLogic ServerのKubernetesロギング

Oracle WebLogic Serverには、サーバー・ログを保持するためのさまざまなオプションが用意されています。サーバー・ログは、Kubernetes上の永続ボリュームに永続的に保持したり、LogstashまたはFluentDを使用してElastic Stackに送信したり、Oracle WebLogic Logging ExporterによってElastic Stackに直接送信したりすることができます。

Elastic Stackのコアは、Elasticsearch、Logstash、Kibanaから成る3つのオープン・ソース製品です。Elasticsearchは検索分析エンジンです。Logstashはサーバー側のデータ処理パイプラインで、複数ソースからのデータを同時に取り込み、変換して、Elasticsearchなどの"stash"に送信します。Kibanaによりユーザーは、Elasticsearchでチャートやグラフを使用してデータを可視化することができます。

Oracle WebLogic Logging Exporterは、Elasticsearch REST APIを使用することによってOracle WebLogic ServerのログをKubernetesのElastic Stackに直接統合できるように、ログ・イベント・ハンドラをOracle WebLogic Serverに追加します。Oracle WebLogic Logging Exporterには、Oracle WebLogic Logging Exporterのインストール方法に関するステップ・バイ・ステップの手順を含む、独自のGitHubプロジェクトが組み込まれています。

移行結果の要約

上記の移行プロセスが完了すれば、以下の点で、Oracle Private Cloud ApplianceのKubernetesでOracle WebLogic Serverアプリケーションを実行する基盤を据えたことになります。

- 本番でのアプリケーションの監視
- アプリケーションのライフサイクル管理

- ワークロードの要求を満たすために必要とされるスケーリング構成
- 本番の問題の分析
- アプリケーションの開発および管理におけるCI/CDおよびDevOpsの広範な導入
- 本番アプリケーションへの継続的な更新の有効化
- 本番システムの継続的なパッチ適用の有効化
- 追加アプリケーションの環境への移行およびデプロイメント
- Oracle CoherenceおよびOracle Application Development Frameworkアプリケーションの移行およびデプロイメント
- 新規アプリケーションの開発とデプロイメント
- Oracle Fusion Middleware製品アプリケーションの将来のデプロイメント

お客様は、オラクルが全面的にサポートする、Kubernetesで提供される新機能により、既存のアプリケーション投資を活用し続けることができます。

また、クラウド・デプロイメント用の標準プラットフォームとして登場したクラウド・ネイティブ・インフラストラクチャに移行したことになり、Oracle Cloud Infrastructureなどのパブリック・クラウドを含め、クラウド間でのアプリケーションの移植性を高め、統合することができます。これにより、アプリケーション開発のための新しいマイクロサービス・テクノロジーの導入が可能になります。たとえば、オラクルによって支援およびサポートされるオープン・ソース・プロジェクトであるHelidonの使用を検討可能です。HelidonはJavaライブラリのセットで、既存アプリケーションの拡張、または新規のネット・アプリケーションの開発のためのマイクロサービスに適しており、Oracle WebLogic ServerおよびOracle Coherenceとの統合をサポートしています。オラクルはまた、これらと追加テクノロジーにおよびハイブリッド・アプリケーションの統合管理をサポートし、ハイブリッド・プライベート/パブリック・クラウド・デプロイメントに及ぶ管理もサポートする、ツールの作成にも取り組んでいます。ビジネスのニーズを満たすようにアプリケーション・ポートフォリオを進化させるための多くのオプションが選択可能になります。

結論

オラクルは、プライベート・クラウド・デプロイメント向けのOracle Private Cloud ApplianceまたはOracle Private Cloud at Customerとともに、これらのシステム上のKubernetesクラスタを作成および管理するためのツールを提供します。またオラクルでは、Oracle Exalogic Elastic Cloudシステム上のアプリケーションを含む既存のOracle WebLogic Serverアプリケーションを、オラクルがサポートするオープン・クラウド・ネイティブ・インフラストラクチャ上のこれらのプライベート・クラウド・システムに移行できるようにするツールも用意しています。そうすることにより、Oracle WebLogic Serverアプリケーションでの投資を保持し、新しいKubernetesとテクノロジーを利用してこれらのアプリケーションの機能を強化し、ビジネスの要件に従って、組織が、進化するアプリケーションに対処できるようにすることができます。

ORACLE CORPORATION

Worldwide Headquarters
500 Oracle Parkway, Redwood Shores, CA 94065 USA

海外からのお問い合わせ窓口
電話 + 1.650.506.7000 + 1.800.ORACLE1
FAX + 1.650.506.7200
oracle.com

CONNECT WITH US

+1.800.ORACLE1までご連絡いただくか、oracle.comをご覧ください。北米以外の地域では、oracle.com/contactで最寄りの営業所をご確認いただけます。

blogs.oracle.com/oracle facebook.com/oracle twitter.com/oracle

Integrated Cloud Applications & Platform Services

ホワイト・ペーパー Oracle Private Cloud ApplianceおよびKubernetes上のOracle WebLogic Server
2019年8月
著者：Monica Ricelli
共著者：[適宜入力]

Oracle is committed to developing practices and products that help protect the environment