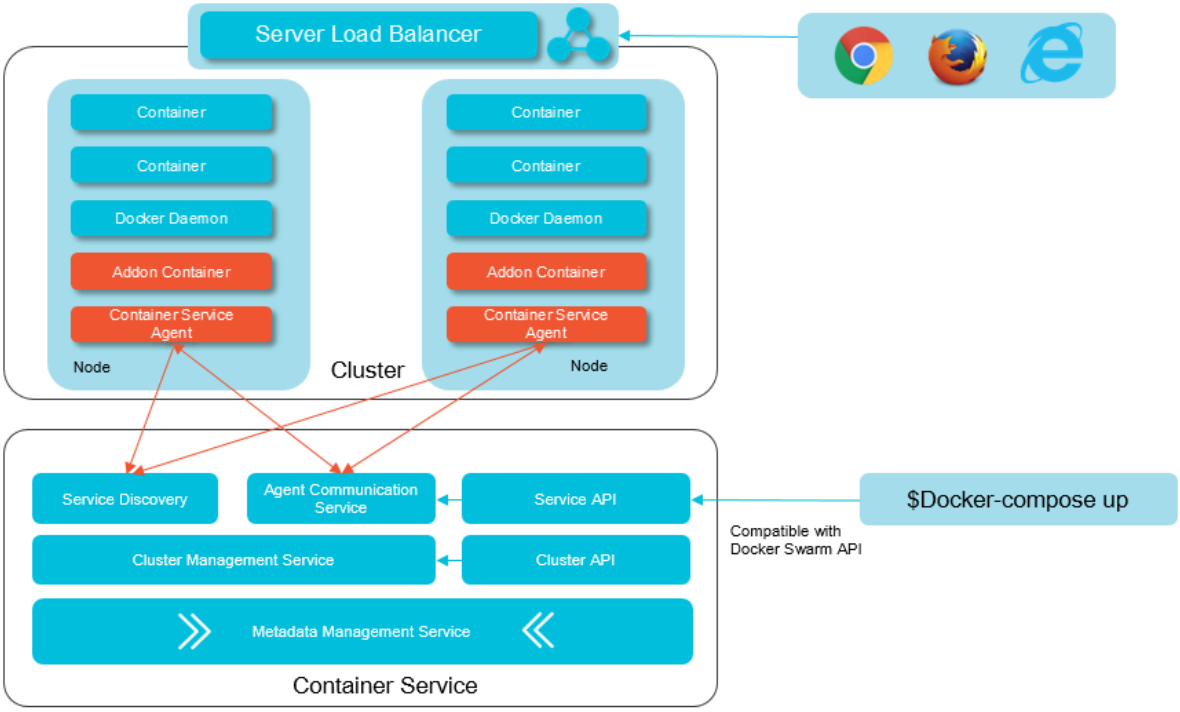
Alibaba Cloud Container Service

プロダクト紹介

Document Version20190629

目次

1 アーキテクチャー



Container Service の基本的なアーキテクチャーは上図とおりで、以下に説明します。

- クラスター管理サービス: Docker クラスター管理とスケジューリングがサポートされています。
- サービスの検出: ストレージのメタデータ (Docker のステータスを含む) がサポートされています。
- エージェント通信サービス: 各ホストとクラスター管理サービス間での通信がサポートされています。
- クラスター API: Alibaba Cloud の 統合 API が提供されています。
- サービス API: Docker Swarm API と互換性のある API が提供されています。

2 利点

使いやすさ

- ワンクリックでコンテナークラスターの作成をサポート
- コンテナーに基づいたワンストップアプリケーションライフサイクル管理
- Alibaba Cloud の仮想化、ストレージ、ネットワークおよびセキュリティに関する機能との統合
- グラフィカルユーザーインターフェイスおよび API のサポート

安全で制御可能

- Alibaba Cloud Container Service では、コンテナーは独自の ECS (Elastic Compute Service) インスタンス上で実行され、他のユーザーとは共有されません。 そのため、コンテナー間の分離問題は存在しません。
- ネットワークに関しては、コンテナークラスター内の ECS インスタンスおよびコンテナーのアクセスポリシーを定義するセキュリティグループを利用して、コンテナーにアクセスする一部のソースのアドレスの許可または拒否ができます。
- コンテナークラスターの管理 API は相互証明書検証を使用して、不正なユーザーによるインターフェイスへのアクセスを防止します。
- NeuVector などのコンテナーセキュリティに関する特別なソリューションと Alibaba Cloud Container Service が簡単に統合でき、高いレベルのセキュリティ保護を提供します。

プロトコルの互換性

- Swarm と Kubernetes の両方をサポート
- 世界で初めての Kubernetes の適合認証をパスしたバッチ
- クラウドプラットフォームへのアプリケーションのシームレスな移行およびバイブリッドクラウドの管理のサポート

効率と信頼性

- 数秒での大規模なコンテナー起動のサポート
- コンテナーの例外リカバリと自動スケーリングのサポート
- ゾーン間でのコンテナーのスケジューリングのサポート

3 シナリオ

DevOps の連続配信

連続配信処理の最適化

Jenkins を使用することで、Container Service はコードの申請からアプケーションのデプロイまで DevOps の完全なプロセスを自動的に完了し、 自動テストにパスしたコードのみ配信およびデプロイすることを確実にします。また、複雑なデプロイ方法や遅い反復処理といった業界の従来の方法を効率的的に置き換えます。

Container Service は次のような実装が可能です。

- DevOps の自動化

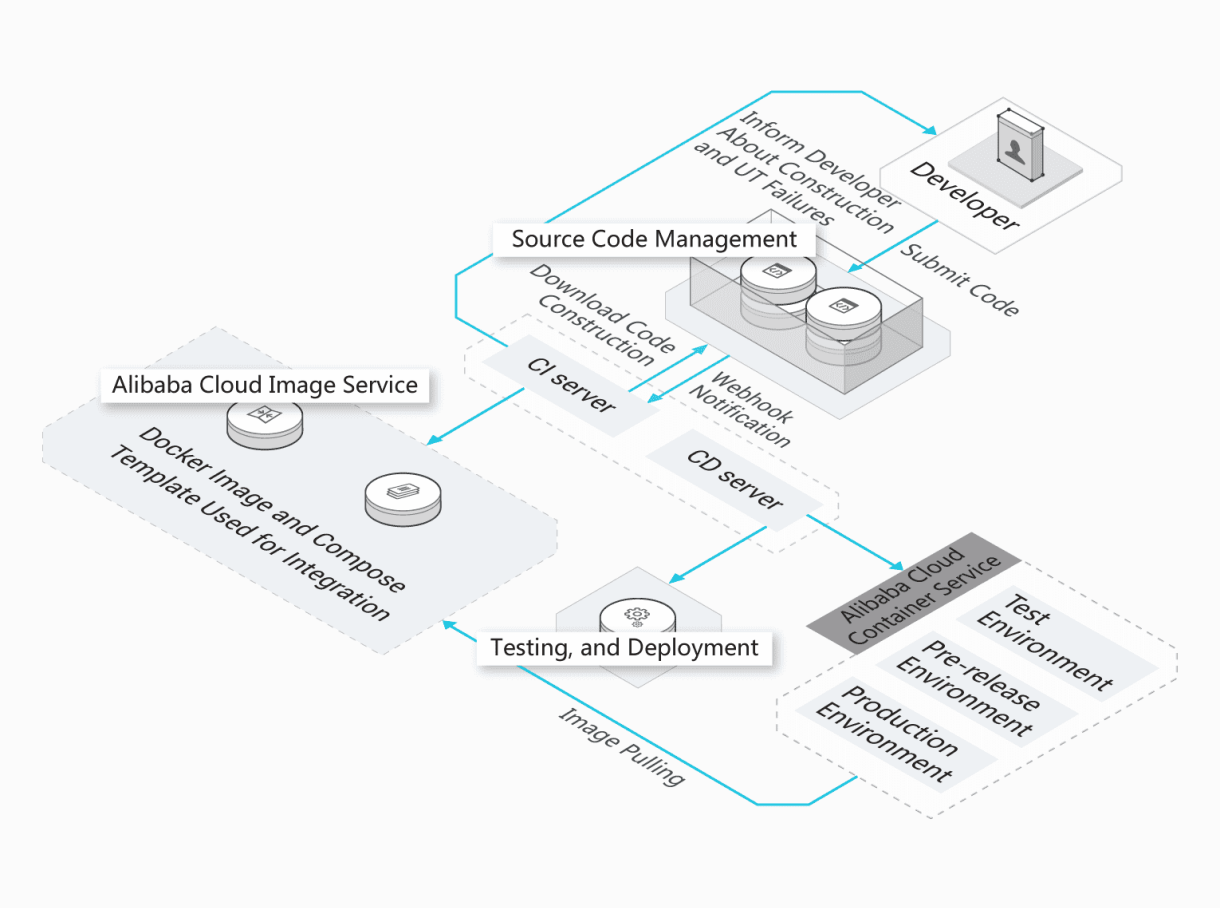
  コード変更から、コードの生成、画像生成、アプリケーションのデプロイまですべての処理を自動化することができます。
- 一貫した環境

  Container Service によりコードのみならず、不変のアーキテクチャを基にした実行環境も配信することができます。
- 継続的なフィードバック

  統合結果やデプロイの結果がリアルタイムでフィードバックされます。

推奨する使用例

ECS (Elastic Compute Service) とContainer Service の同時利用を推奨します。



マイクロサービスアーキテクチャ

企業のビジネス活動を加速させる迅速な開発とデプロイの実行

企業での本番環境において、マイクロサービスは適切に分割され、それぞれのマイクロサービスアプリケーションは Alibaba Cloud のイメージリポジトリに保存されます。 マイクロサービスアプリケーションを繰り返すだけで、Alibaba Cloud によりスケージューリング、オーケストレーション、デプロイおよびゲート起動が提供されます。

Container Service は次のような実装が可能です。

- Server Load Balancer とサービスの検出

  レイヤー 4 およびレイヤー 7 のリクエスト転送とバックエンドバインディングをサポートします。
- スケジューリングと例外リカバリの複数のポリシー

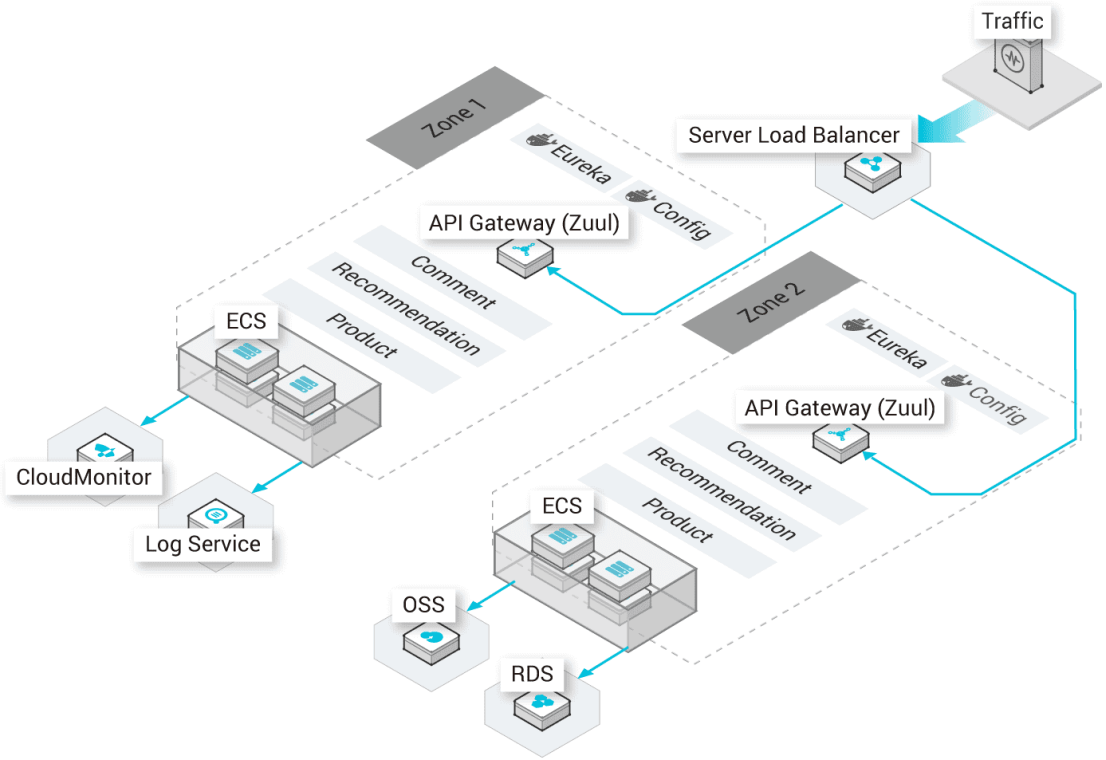
  サービスレベルのアフィニティスケジューリングをサポートします。 クロスゾーンの高可用性とディザスタリカバリをサポートします。

・ マイクロサービスのモニタリングとオートスケーリング

マイクロサービスおよびコンテナーレベルでのモニタリングをサポートします。 マイクロサービスのオートスケーリングをサポートします。

推奨する使用例

ECS、RDS (Relational Database Service)、OSS (Object Storage Service) および Container Service の同時使用を推奨します。



ハイブリッドクラウドアーキテクチャ

多様なクラウドリソースの O&M (Operation and Maintenance) の統合

Container Service コンソールで、クラウド上とクラウド外にあるリソースを、複数のクラウドコンソール間で切り替えることなく同時に管理できます。 コンテナーインフラストラクチャーに関連付けられていない特性を基にした同一のイメージとオーケストレーションを使用して、クラウド上とクラウド外のアプリケーションを同時にデプロイすることができます。

Container Service は以下の項目をサポートします。

・ クラウド上のアプリケーションのスケールインおよびスケールアウト

業務のピーク時に業務トラフィックをクラウドに割り当てるためのクラウド容量の迅速な拡張

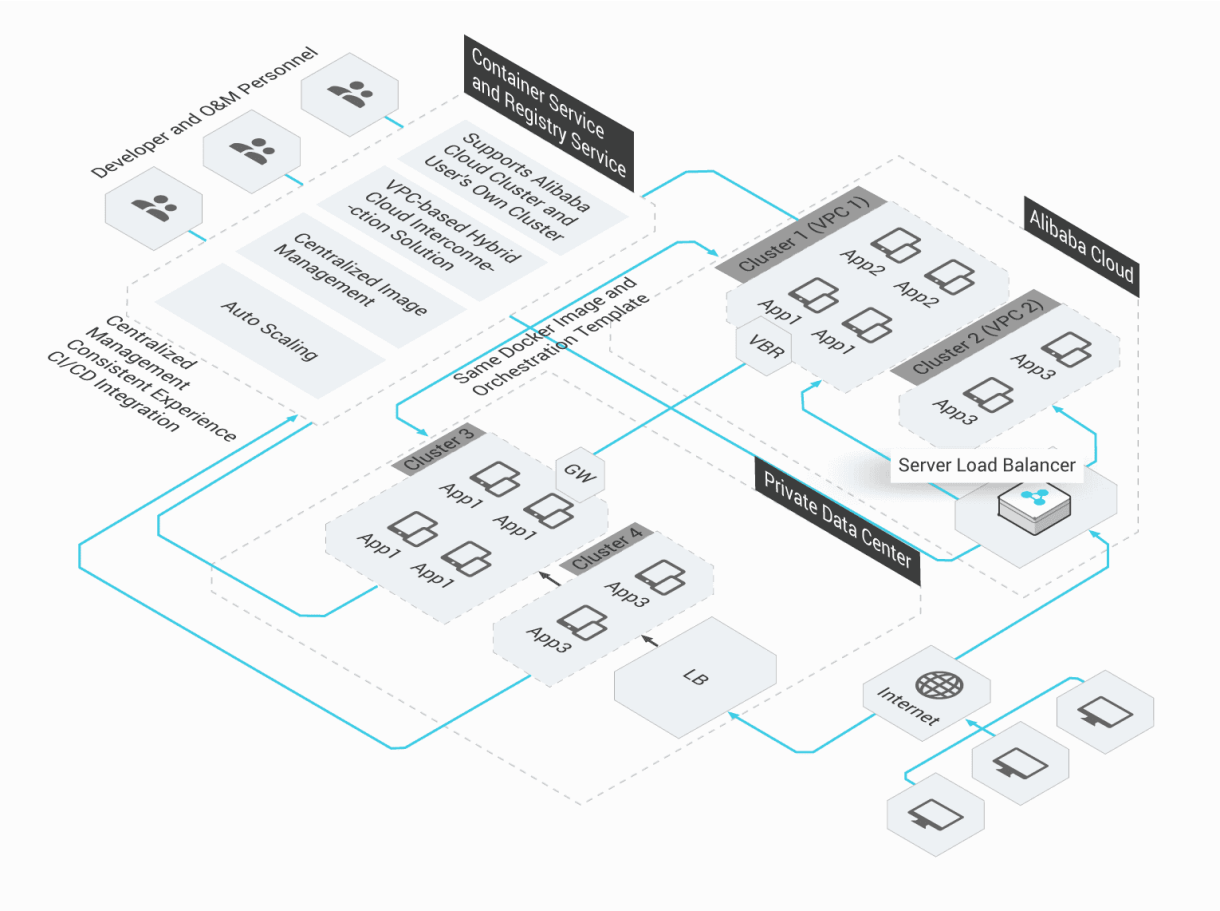
- クラウド上でのディザスタリカバリ

  クラウド上とクラウド外の業務システムを同時にデプロイすることで、クラウド外のサービスおよびクラウド上のディザスタリカバリを提供します。
- クラウド外での開発およびテスト

  クラウド外で行われた開発およびテスト後に、クラウド上でアプリケーションをシームレスなにリリースします。

推奨する使用例

ECS、VPC (Virtual Private Cloud) および Express Connect の同時使用を推奨します。



オートスケーリングアーキテクチャ

業務トラフィックに応じた業務の自動的な拡張と縮小

Container Service により手動で介入することなく、業務トラフィックに応じて自動的に拡張または縮小を行うことができます。 これにより、突発的なトラフィックの増加によるシステムダウンや、タイミングの悪い拡張、過剰なアイドルリソースによる無駄なリソース消費を避けることができます。

Container Service は次のような実装が可能です。

- 迅速な対応

  ビジネストラフィックが拡張を必要とするインジケーターに達した際、数秒でのコンテナー拡張を実行します。

- 完全自動化

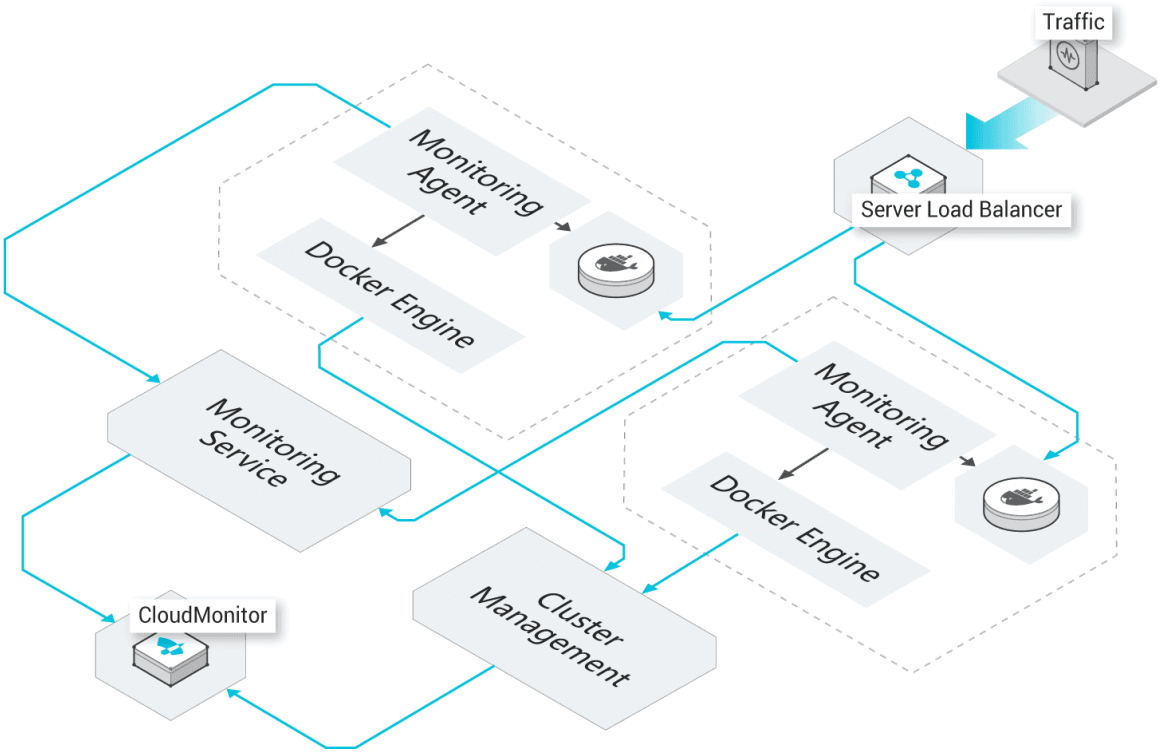
  手動での介入がいらない、拡張および縮小処理が完全に自動化されます。

- 低コスト

  トラフィック減少時、無駄なリソース消費を避けるため自動的に容量の縮小を行います。

推奨する使用例

ECS と Cloud Monitor の同時使用を推奨します。



4 用語集

基本的な用語

クラスター

クラウドリソースの集まりで、コンテナーを実行する際に必要です。 複数のサーバーノード、Server Load Balancer インスタンス、VPC (Virtual Private Cloud) およびその他のクラウドリソースと関連付けられます。

ノード

Docker Engine がイントールされたサーバー (ECS (Elastic Compute Service) インスタンスまたは物理サーバーのいずれか) で、コンテナーのデプロイや管理に使用されます。 Container Service のエージェントプログラムはノード内にインストールされ、クラスターに登録されます。 クラスター内のノード数は、スケーラブルに変更できます。

コンテナー

Docker イメージを使用して作成されたランタイムインスタンスです。 1 つのノードで複数のコンテナーを実行できます。

イメージ

Docker におけるコンテナーアプリケーションの標準パッケージ形式です。 イメージを指定して、コンテナーアプリケーションをデプロイできます。 イメージは、Docker Hub、Alibaba Cloud Container Hub、またはユーザーのプライベートレジストリから取得できます。 イメージ ID は イメージリポジトリおよびイメージタグ (デフォルトでは最新) の URI によって 一意に識別されます。

オーケストレーションテンプレート

Container Service のグループとグループの相互通信関係の定義を含むテンプレートタイプです。複数のコンテナーアプリケーションのデプロイと管理に使用されます。 Container Service は Docker Compose テンプレートの仕様をサポートおよび拡張します。

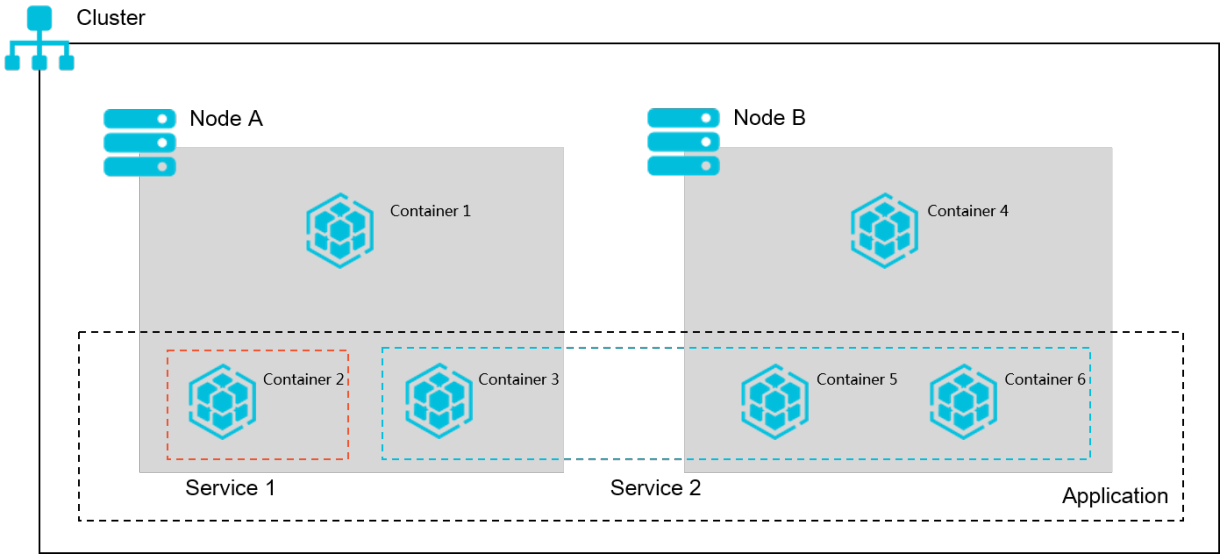
アプリケーション

アプリケーションソフトウェアは、イメージまたはオーケストレーションテンプレートを使用して作成できます。 1 つのアプリケーションに、1 つ以上のサービスを含めることができます。

サービス

同じイメージと設定に基づいて定義されたコンテナーのグループです。 スケーラブルなマイクロサービスとして使用されます。

関連性



Kubernetes に関する用語

Kubernetes は、Google によるオープンソースの大規模コンテナーオーケストレーションおよびスケジューリングシステムです。コンテナー化されたアプリケーションのデプロイ、拡張、管理を自動的に行うために使用され、移植性、スケーラビリティ、自動スケジューリングなどの特性を持ちます。

ノード

Kubernetes クラスターのノードは、コンピューティングパワーを備え、すべてのポッドが実行されているアクティブなサーバーです。 アクティブなサーバーは物理マシンまたは仮想マシンです。kubelet 管理ノードで実行されるコンテナーは、アクティブなサーバーで実行されなければなりません。

名前空間

名前空間は Kubernetes クラスターを仮想的に分離します。 デフォルトでは、Kubernetes クラスターには 3 つの名前空間があります。デフォルトの名前空間である "default"、システムの名前空間である "kube-system" と "kube-public" です。 管理者は要件に適した新しい名前空間を作成することができます。

ポッド

Kubernetes の最小限の基本ユニットで、アプリケーションやサービスのデプロイに使われます。 ポッドは 1 つ、または複数のコンテナー、ストレージリソース、独立したネットワーク IP

アドレス、 および実行しているコンテナーのメソッドの管理および制御するポリシーオプションをカプセル化することができます。

レプリケーションコントローラー

RC (Replication Controller) により、実行中のポッドをモニタリングすることで、いつでも Kubernetes クラスター上で指定した数のポッドレプリカが実行されていることを確認できます。 1 つ以上のポッドレプリカを指定できます。 指定した数よりもポッドレプリカの数が小さい場合、RC により新しいポッドレプリカが実行されます。 ポッドレプリカが指定した数を超えた場合、冗長なポッドレプリカを停止させます。

レプリカセット

RC のアップグレードされたバージョンです。 RS (Replica Set) と RC の違いは、セレクターへのサポートのみです。 RS はより多くの適合したモードのタイプに対応しています。 一般的に、RS オブジェクトは独立して使用されず、 理想的な状況下でデプロイに関するパラメータとして使用されます。

デプロイ

デプロイとは、ユーザーによる Kubernetes クラスターに関するアップデート操作を指します。 RS よりも広いアプリケーションの範囲を持ち、サービスの作成、サービスのアップデート、サービスのローリングアップデートを行うことができます。 ローリングアップデートの実行は、実際に新しい RS を作成し、新しい RS 上で理想的な状態までレプリカ数を段階的に追加、 および古い RS を段階的に 0 にするまで削減します。 このような複合操作は RS による十分な解説はありませんが、 より一般的なデプロイにより解説されます。 手動での管理やデプロイにより作成された RS の使用は推奨しません。

サービス

Kubernetes の基本操作ユニットです。 実際のアプリケーションサービスの概念では、それぞれのサービスに多くのコンテナーがあり、サポートする機能を提供します。 Kube-Proxy のポートおよびサービスセレクターによりバックエンドコンテナーに渡すサービスリクエストが決定されます。また、1 つのアクセスインターフェイスが外部で表示されます。 外部機能に対し、バックエンドがどのように機能するかを通知する必要がなく、これにより有効なバックエンドの拡張と管理が行えます。

ラベル

基本的にキーとバリューのペアのコレクションがリソースオブジェクトに対して付与されます。 ラベルは、ユーザーに対して有効なオブジェクトの属性を指定するために使用されます。ただし、カーネルシステムに関して直接的な意味付けをすることはできません。 ラベルは、オブジェ

クトの作成時に直接付与するか、いつでも編集することができます。 1 つのオブジェクトに対して複数のラベルを付与することができますが、 キー、バリューは固有のものでなければなりません。

ボリューム

Kubernetes クラスターにおけるボリュームは Docker ボリュームと同様のものになります。 違いとしては、Docker ボリュームの範囲はコンテナーであり、 Kubernetes ボリュームのライフサイクルおよび範囲はポッドになります。 それぞれのポッドにおいて設定されたボリュームは、そのポッド内の全てのコンテナーにより共有されます。 PVC (Persistent Volume Claim) 理論ボリュームを使用でき、バックエンドでの実際のストレージ技術を気にする必要はありません。 PV (Persistent Volume) に関する特定の設定はストレージ管理者が完了させます。

PV および PVC

PV および PVC により、 Kubernetes に対しストレージの抽象理論機能を設定できます。PV コンフィギュレータが設定を行うため、ポッドの設定ロジックにおける実際のバックエンドストレージ技術の設定を気にする必要がありません。 ストレージにおける、PV と PVC の関係性は、コンピューティングにおけるノードとポッドの関係性と同様のものです。 PV およびノードはリソースプロバイダーであり、クラスターインフラストラクチャーにより変更され、Kubernetes クラスター管理者により設定されます。 PVC および ポッドはリソースユーザーであり、業務要件およびサービス要件により変更され、 Kubernetes クラスターユーザー、つまりサービス管理者により設定されます。

Ingress

クラスターに対するインバウンドアクセスの認証ルールのコレクションです。 Ingress の設定を使用し、外部アクセス可能な URL、Server Load Balancer、SSL および名前ベースのバーチャルホストを提供できます。 Ingress リソースを API サーバーに送信することで、 Ingress をリクエストできます。 Ingress コントローラーにより、一般的に Server Load Balancer を利用して Ingress を実装され、 エッジルーターおよび他のフロントエンドを設定できます。これにより、HA メソッドを利用したトラフィック操作が行えます。

関連ドキュメント

- *Decker* に関する用語
- *Kubernetes* の概念