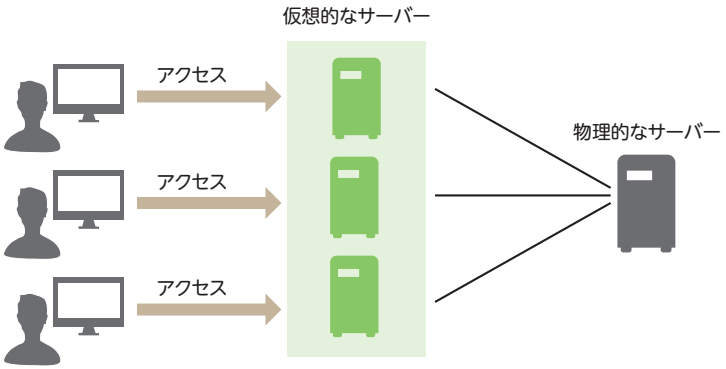
01 仮想化とは
～物理構成にとらわれない柔軟性と可用性

仮想化を使うと、物理的な実体とは異なる構成でサービスを構築できます。たとえば、1台の物理的なサーバー上に、複数の仮想的なサーバーを動作させることが可能です。まずは、仮想化とは何かについて見ていきましょう。

仮想化とは

仮想化とは、**実体のないものをあたかも実在しているかのごとく表現する技術のこと**です。たとえば、1台の物理的なサーバー上に、複数の仮想的なサーバーを構築できます。こうすると、物理的なサーバーを増やすことなく、必要に応じてサーバーを増減することが可能になります。

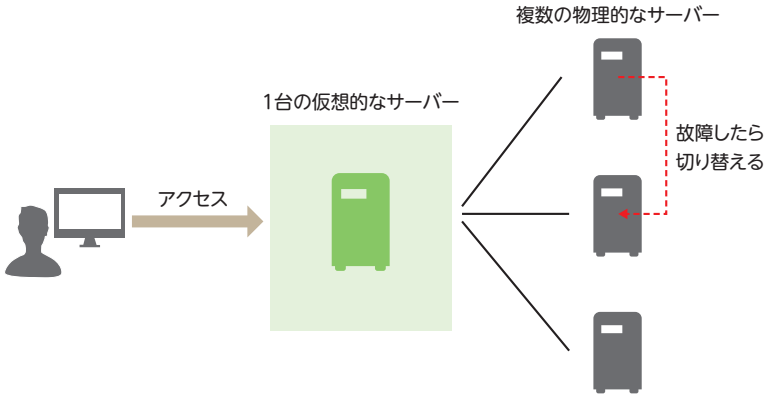
■物理的なサーバーに複数の仮想的なサーバーを構築可能



逆に、仮想化によって、**実際は複数の物理的なサーバーで構成されているのに、ユーザーには1つのサーバーに見せかけることも可能**です。たとえば、オンラインのメールサービス（Google社の「Gmail」など）をイメージしてください。もしあなたが利用しているメールサーバーが故障し、そのサーバーが復旧するまでメールを使えないとしたらとても不便です。しかし実際には1つのメールサーバーが故障しても、ユーザーはそれを知ることもなくサービスの利用を継続することができます。

これらのサービスでは、もともと複数の物理的なサーバーで構成されており、その上で仮想サーバーが動いています。故障したサーバーが出ても自動的に切り離し、他のサーバーに切り替えることで、同じ処理を継続することができるようになっているのです。ユーザーはサーバーの故障を一切知ることもなく、あたかも1台のサーバーが提供するサービスのごとく、いつもと同じアクセス方法で、サービスを受け続けることができます。

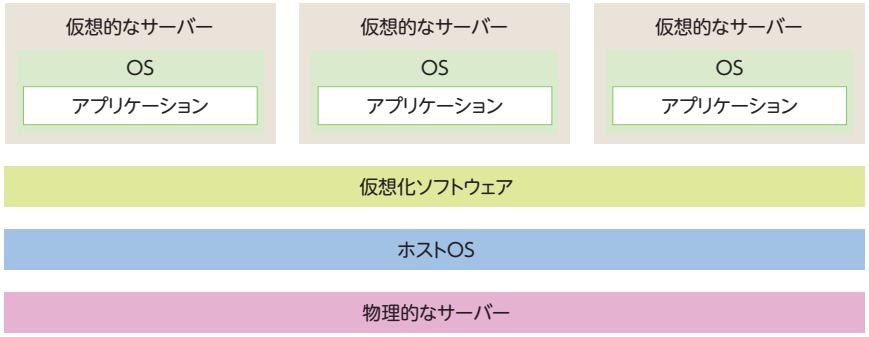
■複数のサーバーが1台のサーバーであるかのように見せる



仮想化の概念

仮想化の概念を理解するには、次ページの図を見ていただくのがよいでしょう。この図は、物理的なサーバー上で仮想的なサーバーを3台動作させている様子を表しています。まず、「物理的なサーバー」が最下位に存在します。その上の「ホストOS」とは、物理的なサーバー上にインストールされているOS（オペレーティングシステム。コンピューターを動かすためのソフトウェア）のことです。さらにその上には、**仮想化ソフトウェア**があります。これは、仮想的なサーバーを動作させるうえで必要となるソフトウェアです。この仮想化ソフトウェアこそが、仮想化を実現するものです。

■ サーバーの仮想化



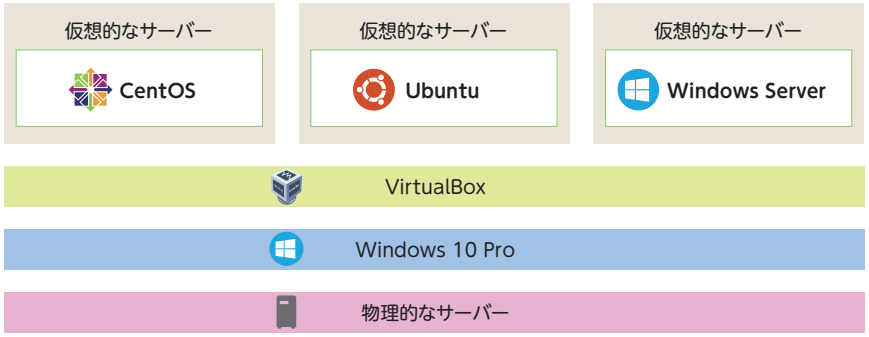
SF映画などで、主人公が非常にリアルな仮想現実（バーチャルリアリティー）に没入して、現実と区別ができなくなる物語があります。プログラムの場合も同じで、実際のハードウェアとまったく同じ反応を返す仮想化ソフトウェアがあれば、実ハードウェアと同じようにプログラムが動作します。これが仮想化の基本的な原理です。

柔軟に環境を構築できる

仮想化によって、物理的な環境の制約にとらわれず、柔軟に環境を構築することができます。これが仮想化の持つ**柔軟性**です。物理的なサーバーを新たに購入しなくとも、あたかも物理的なサーバーが存在するかのごとく、新たなサーバーを仮想的に用意することが可能です。

たとえば、物理的なサーバーのOSを、Windows 10 Proとします。Windows 10 Proに、「VirtualBox」というOracle社の仮想化ソフトウェアをインストールすることで、仮想的なサーバーを構築できます。仮想的なサーバーでは、CentOSやUbuntu、Windows Serverといった、ホストOSとは別のOSを動作させることも可能です。1台のサーバーで3台分の仮想的なサーバーを構築できるので、コストや設置スペースの削減にもつながります。

■ 仮想的なサーバーの構築例



可用性が向上する

可用性とは、「システムが故障せずに継続して利用できるかどうか」を示す指標です。仮想化を用いれば、作成した仮想環境をその環境ごとファイルとしてバックアップしておくことが可能なため、もし仮想環境が故障しても、バックアップから復元することですぐに以前の状態に戻すことができます。これは、システムの可用性を向上させます。ただし、仮想的なサーバーは、物理的なサーバーをシャットダウンしてしまえば、使用することはできません。

まとめ

- **仮想化は実体のないものをあたかも実在しているかのごとく表現する技術**
- **仮想化によって1台の物理的なサーバー上に複数の仮想的なサーバーを構築できる**
- **仮想的なサーバーはバックアップすることが可能なので障害時に以前の状態に戻すのが容易である**

06 サーバー仮想化のしくみ
～物理サーバーに複数の仮想サーバーを構築

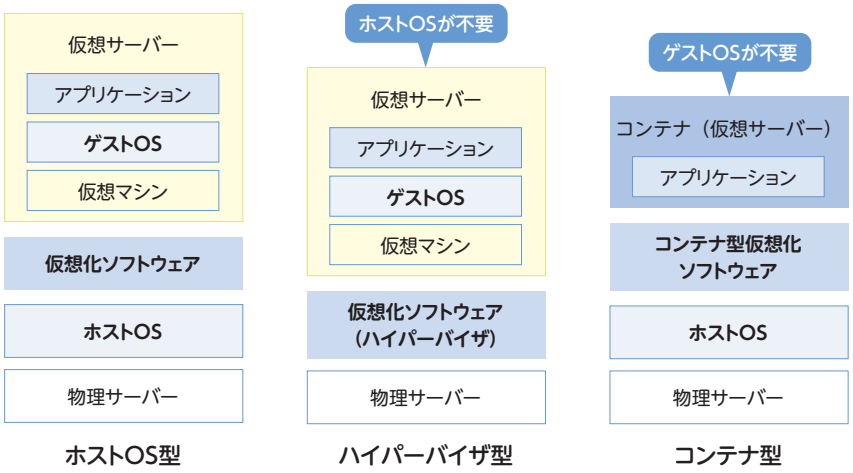
本章では、仮想化のしくみや技術について詳しく説明します。まずは、第1章でも紹介した「サーバー仮想化」について見ていきましょう。「サーバー仮想化」には、大きく3つの種類があります。

サーバー仮想化とは

サーバー仮想化とは、物理サーバー上に、仮想的なサーバーを構築する技術のことです。第1章で説明したサーバー仮想化は、実は**ホストOS型仮想化**といって、サーバー仮想化の種類の1つです。サーバー仮想化にはほかに、**ハイパーバイザ型仮想化**と**コンテナ型仮想化**があります。

この3種類のサーバー仮想化について大まかな構成を見ると、ホストOSとゲストOSの有無がそれぞれ異なります。**ホストOS**は物理サーバーに直接インストールするOSを指し、**ゲストOS**は仮想サーバーにインストールするOSを指します。

■ 3種類のサーバー仮想化



サーバー仮想化の種類

3種類のサーバー仮想化については次節以降で解説していきますが、ここでは概要を紹介します。

1つ目のホストOS型は、ホストOS上で仮想サーバーを動かします。Windows上でLinuxを利用する、もしくはMac上でWindowsを利用するといった、サーバー構築以外の用途でも使われます。

2つ目のハイパーバイザ型は、ホストOSの一部機能を仮想化ソフトウェアが代行することにより、ホストOSを省いてメンテナンス性を高めたものです。

3つ目のコンテナ型は、使用リソースが低く柔軟性が高いため、Webアプリケーションの開発・運用環境として広く利用されています。その反面、コンテナを活かすためには、「アプリケーション単位で仮想化する」という考え方やそのメリットを理解する必要があります。

■ サーバー仮想化の種類

種類	概要
ホストOS型	物理サーバーにインストールされたホストOS上で仮想サーバーを動作させる。代表的なソフトウェアはVirtualBox
ハイパーバイザ型	ホストOSなしで動作するハイパーバイザと呼ばれる仮想化ソフトウェアを利用する。ホストOSがない分、ホストOS型よりメンテナンス性が高いとされる。代表的なソフトウェアはHyper-V
コンテナ型	コンテナという、アプリケーションと実行環境をまとめて隔離するしくみを利用し、OS単位ではなくアプリケーション単位で仮想化する。仮想マシンやゲストOSがなくなり、コンテナ同士を組み合わせて利用する。代表的なソフトウェアはDocker

まとめ

- **サーバー仮想化は物理サーバー上に仮想的なサーバーを構築する技術**
- **サーバー仮想化にはホストOS型仮想化、ハイパーバイザ型仮想化、コンテナ型仮想化の3つがある**

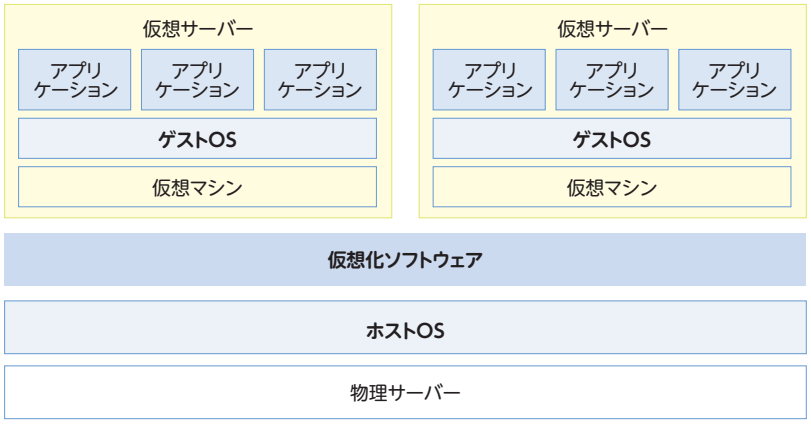
07 ホストOS型仮想化
～ホストOS上で複数のゲストOSが動作

サーバー仮想化のうちの1つである「ホストOS型仮想化」は、ホストOS上で仮想サーバーを動作させます。仮想化としてはもっともイメージしやすく、かつかんたんに試すことができます。

ホストOS型仮想化とは

ホストOS型仮想化は、**物理サーバーにOSがインストールされており、その上で仮想サーバーを動作させる技術**です。そのOSに対して、ホストOS型仮想化ソフトウェアをインストールします。たとえば、ホストOSをLinuxとし、その仮想化ソフトウェア上に、ゲストOSとしてWindowsを動作させることが可能です。どのようなゲストOSが動作できるかは、ホストOSにインストールしたホストOS型仮想化ソフトウェアの種類によります。

■ ホストOS型仮想化



ホストOS型仮想化ソフトウェア

ホストOS型を実現する仮想化ソフトウェアには、いくつか種類があります。ここでは、主なホストOS型仮想化ソフトウェアを紹介します。

■ 主なホストOS型仮想化ソフトウェア

製品名	提供元	プラットフォーム	概要
Hyper-V	Microsoft	Windows	Windows 8以降のWindows OSのProにプレインストールされている。ハイパーバイザ型にも同じ名前の製品があるが、Windows Server製品でないものはホストOS型
VMware Workstation Player	VMware	Windows、Linux	商用利用でなければ無償で利用可能
VMware Horizon	VMware	Windows、Linux	有償だが、オンプレミスにもクラウドにも対応している
VMware Fusion	VMware	Mac	VMwareのMac版。ゲストOSにWindowsを実行可能。通常版とProfessional版があるが、どちらも有償
VirtualBox	Oracle	Windows、Linux、Mac	Open Source Software（OSS）として提供。無償で商用利用も可能だが、低スペックのマシンであれば、上述のVMWare Workstation Playerのほうが快適に動作する
Parallels Desktop	Parallels	Mac	MacでWindowsを実行する場合に利用

まとめ

- **ホストOS型仮想化は物理サーバーにOSがインストールされており、その上で仮想サーバーを動作させる技術**
- **どのようなゲストOSが動作できるかはホストOSにインストールしたホストOS型仮想化ソフトウェアの種類による**

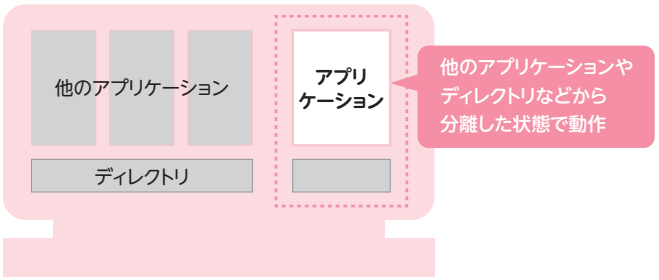
19 コンテナ技術の歴史
～意外と古いコンテナ技術の起源

サーバー仮想化の1つであるコンテナ技術は、サービスの開発手法として大変人気が高く、重要な技術となっています。そのため本章からは、コンテナ技術の詳細を解説していきます。まずは、コンテナ技術の歴史から見ていきましょう。

コンテナ技術の歴史

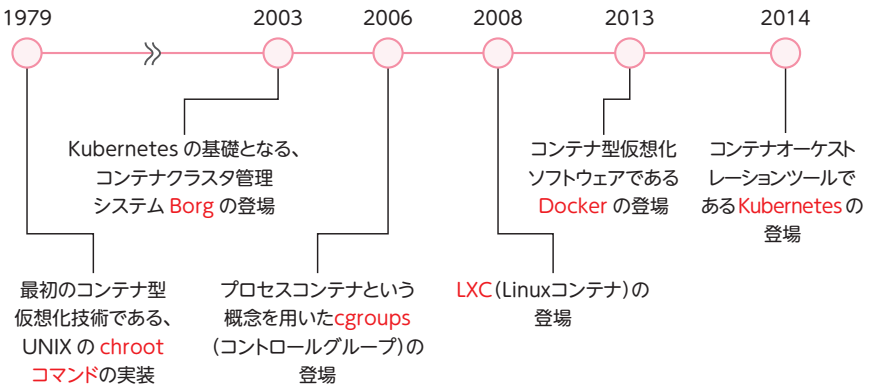
コンテナ技術の歴史は意外に古く、最初のコンテナ技術は、1979年に登場した、UNIXの**「chroot」コマンド**です。その働きは、アプリケーションによるアクセスを指定したディレクトリに限定するというものでした。そのディレクトリ上において、プロセスは分離して利用されるため、あるアプリケーションのプロセスが他のアプリケーションのプロセスに影響を与えることはありません。このように、コンテナ型の仮想化は**アプリケーションを隔離する技術**ともいえます。

■ コンテナの基本的な原理



その後コンテナ技術の成長期を経て、2013年に、現在も代表的なコンテナ型仮想化ソフトウェアである**Docker**が登場します。そのおかげで、ホストOS型やハイパーバイザ型と比べて**手軽にサーバー仮想化を実現**できるようになりました。しかし、利用頻度が高まり大量のコンテナを同時に使用するようになるにつれ、コンテナの管理が煩雑になるという課題が浮き彫りとなりました。その後、この課題を解消するソフトウェアとして**Kubernetes**が開発されました。Kubernetesは**コンテナオーケストレーションツール**と呼ばれ、多くのコンテナを指揮（オーケストレーション）することができます。

■ コンテナ技術の歴史



まとめ

- **最初のコンテナ技術であるUNIXの「chroot」コマンドはアプリケーションによるアクセスを指定したディレクトリに限定することができた**
- **コンテナ型仮想化ソフトウェアであるDockerの登場によって手軽にサーバー仮想化を実現できるようになった**
- **コンテナオーケストレーションツールであるKubernetesが登場したことでコンテナの管理がしやすくなった**

20 コンテナ技術のメリット
～仮想サーバーを手軽に構築できる

第2章で、サーバー仮想化としてホストOS型、ハイパーバイザ型、コンテナ型の3つを説明しました。本節では、コンテナ型をほかのサーバー仮想化と比較した場合のメリットについて紹介します。

コンテナ技術のメリット

コンテナ技術は、仮想マシンやゲストOSを必要としません。そのため、コンテナを作成したらすぐにアプリケーションを導入して、仮想サーバーを構築することができます。この手軽さが最大のメリットといえるでしょう。代表的なコンテナ型仮想化ソフトウェアのDockerでは、主要なアプリケーションやライブラリのコンテナイメージを集めたDocker Hubというサービスが提供されています。そのため、かんたんなサーバーならコマンドを1つ実行するだけで、仮想サーバーのコンテナを立ち上げることができます。

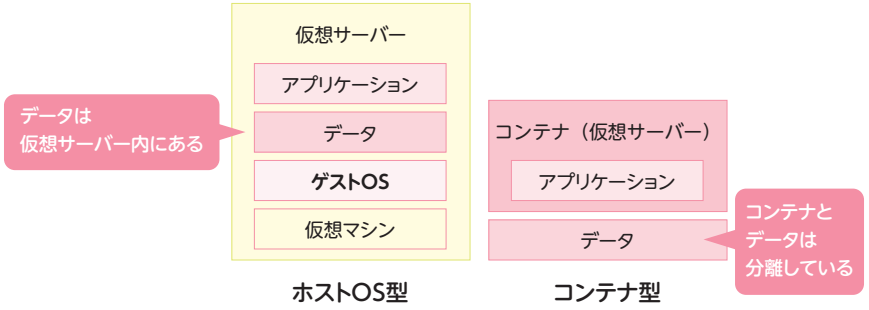
■ ホストOS型とコンテナ型の比較

ゲストOSを持たないということは、コンテナ内のアプリケーションもホストOSの機能（カーネル）を利用して動いていることになります。しかし、ファイルシステムが独立しているため、ホストOS内にインストールされたほかのアプリケーションやファイルなどの影響を受けません。

データはコンテナに含めない

コンテナ技術では、可能な限りデータをコンテナに含めないのが基本です。対してホストOS型やハイパーバイザ型では、アプリケーションが記録するデータは仮想サーバー内にあります。そのため、仮想サーバーがクラッシュするとデータも失われることがあります。コンテナ技術においてコンテナにデータを含めることは可能ですが、一般的にアプリケーション用のコンテナとは分離します。また、データの保管先としてホストOSのディレクトリを指定することもできます。そのため、アプリケーション用のコンテナを破棄しても失われることはなく、データを新しいコンテナで使用することもできます。

■ データはコンテナに含めない



まとめ

- **コンテナ技術を使うと仮想サーバーの構築が容易になる**
- **コンテナ技術では可能な限りデータをコンテナに含めない**

25 Dockerとは
～コンテナ技術のデファクトスタンダード

ここまで仮想化やコンテナの基礎知識について解説してきました。本章では、コンテナ技術のデファクトスタンダードであるDockerについて解説します。まずは、Dockerの概要や特徴を見ていきましょう。

Dockerとは

Docker（https://www.docker.com/）とは、**コンテナの実行やコンテナイメージ（コンテナを実行するためのテンプレート）の作成・配布を行うためのプラットフォーム**です。Dockerは、アプリケーションを実行環境ごとパッケージ化したコンテナイメージを作成します。そのコンテナイメージは、コンテナレジストリを介して、開発環境から運用環境へ配布できます。**実行環境ごとパッケージ化するため、開発環境でも運用環境でも同じようにアプリケーションを動作させることができます。**

Dockerは柔軟性、疎結合などのコンセプトの上に築かれています。Dockerのしくみを学ぶためには、これらのコンセプトを理解することが重要です。

■ Dockerのコンセプト

柔軟性	疎結合
プログラミング言語などが制約されない	システムを独立したコンポーネントへ分解できる
軽量	**スケーラブル**
効率的にリソースを活用する	需要に応じてリソースを増減できる
ポータブル	**セキュア**
異なる実行環境への移行が容易	コンテナ同士を分離できる

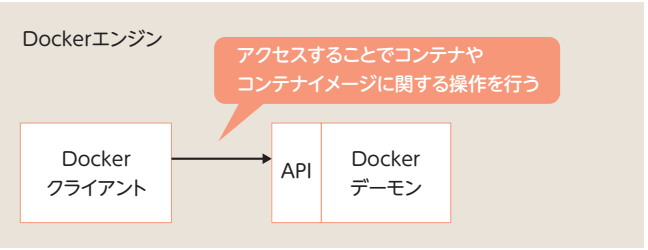
Dockerによって実現できること

Dockerを使うことによって、アプリケーションの高速かつ問題の少ないリリースやアプリケーションの容易なデプロイ（プログラムを配備し、システムを利用可能な状態にすること）、スケールを行うことができます。これらの課題を解決できるのはDockerだけではありませんが、Dockerはこれらの課題を解決するための1つのアプローチを提供します。

Dockerエンジンとは

Dockerエンジンとは、**コンテナやコンテナイメージを管理するためのアプリケーション**です。Dockerエンジンはクライアント・サーバー型のアプリケーションであり、クライアントである「Dockerクライアント」からサーバーである「Dockerデーモン」のAPIへアクセスすることによって、コンテナやコンテナイメージに関するさまざまな操作を行えます。Dockerクライアントから Dockerデーモンの APIへアクセスするためには、Dockerコマンド（P.109参照）を実行します。このDockerエンジンやDockerクライアントを指して「Docker」と呼ぶ場合もあります。

■ Dockerエンジンのしくみ



まとめ

- **Dockerはコンテナの実行やコンテナイメージの作成・配布を行うためのプラットフォーム**

26 Docker が注目される理由
～ Docker の歴史と発展を振り返る

ほかにもコンテナ技術がある中で、Dockerはなぜこれほど普及するに至ったのでしょうか？　本節では、Dockerの歴史と発展を振り返るとともに、Dockerが注目される理由について見ていきましょう。

Dockerの歴史

Dockerは、2013年3月のPython Conferenceのライトニングトークで Solomon Hykes氏によって紹介されました。2014年6月にはDockerエンジンのバージョン1.0が一般に利用可能となり、同年12月にはDockerのダウンロード回数が1億回を突破しました。その後、2016年6月にリリースされたDockerのバージョン1.12では、Docker標準のコンテナオーケストレーション機能であるSwarmモードが追加され、2017年10月にはコンテナオーケストレーションツールのデファクトスタンダードであるKubernetesがDocker Enterpriseに統合されました。さらに、2018年4月にはDocker Enterprise 2.0がアナウンスされ、2019年4月にはDocker Enterprise 3.0がアナウンスされました。Dockerは現在も進化を続けています。

■ Dockerの歴史

- 2013年03月　Python Conferenceのライトニングトークで Solomon Hykes 氏によって紹介
- 2014年06月　Docker エンジンのバージョン 1.0 が一般に利用可能
- 2014年12月　Docker のダウンロード回数が 1 億回を突破
- 2016年06月　コンテナオーケストレーション機能である Swarm モードが追加
- 2017年10月　コンテナオーケストレーションツールの Kubernetes が Docker Enterprise に統合
- 2018年04月　Docker Enterprise 2.0 がアナウンス
- 2019年04月　Docker Enterprise 3.0 がアナウンス

Dockerの注目度

Google Trends（Googleでどれだけ検索されているかを確認できるツール）ですべての国を対象に「Docker」をキーワードとして検索すると、多少のばらつきはあるものの、2013年以降おおむね右肩上がりに注目度が推移していることがわかります。このことから、Dockerへの関心が世界的に高まっているといえます。なお、日本のみを対象に「Docker」をキーワードとして検索すると、すべての国を対象とした場合と同様に、2013年以降おおむね右肩上がりに注目度が推移しています。国内においても、Dockerへの関心が高まっていると推測できます。

■「すべての国」を対象にした検索結果

■「日本」を対象にした検索結果

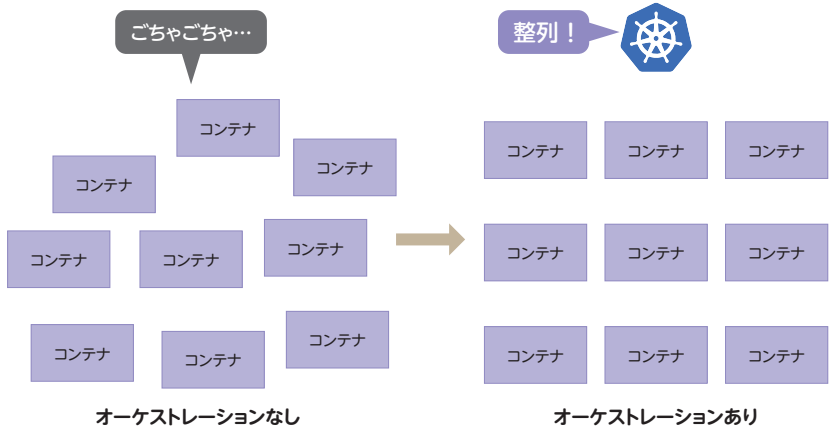
37 コンテナオーケストレーション ～コンテナを管理する際に必要な作業とは

コンテナやコンテナを起動するマシンが増えれば増えるほど、手動で管理するのは大変です。ここでは、コンテナの管理・運用を自動化する「オーケストレーション」の必要性について見ていきましょう。

オーケストレーションとは

オーケストレーション（orchestration）は「指揮」や「編成」を意味する単語であり、コンテナ技術においては**コンテナの管理・運用を自動化すること**を表します。ここで「コンテナの管理・運用」と一言で述べましたが、実際にコンテナを管理・運用する際はどのような作業が必要となるのでしょうか？　複数のコンテナ（Webサーバー、APサーバー、DBサーバー）から構成されるWebアプリケーションを例に挙げながら、「負荷分散」「死活監視」「スケーリング」の3点について、どのような作業を行うかを見ていきましょう。

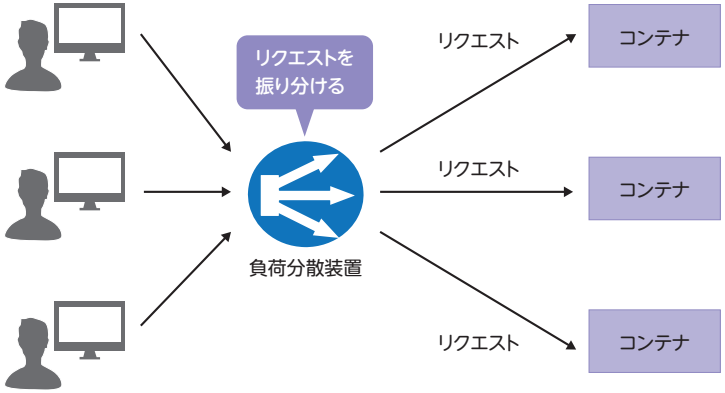
■ オーケストレーション



負荷分散

コンテナ技術における**負荷分散**とは、**複数のコンテナへリクエストを振り分けて負荷を分散すること**です。負荷分散を行う手段の1つとして、リクエストを振り分ける負荷分散装置を、リクエストを処理するコンテナの前段に設ける方法があります。先ほどのWebアプリケーションの場合、Webサーバー、APサーバー、DBサーバーの各層の前段に負荷分散装置を設けます。負荷分散装置でコンテナ1台あたりの負荷を軽減することで、処理時間を短縮したり、単位時間あたりに処理できるリクエストの数を増やしたり、システムの性能を向上させたりすることができます。

■ 負荷分散



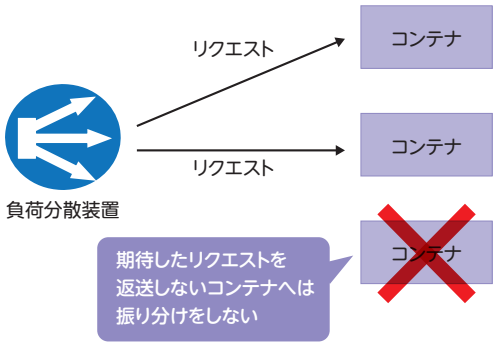
死活監視

コンテナ技術における**死活監視**とは、**システムを構成するコンテナが正常に稼働しているかどうかを監視すること**です。死活監視を行う手段の1つとして、コンテナへリクエストを送信し、それに対してコンテナが期待したレスポンスを返送するのかをチェックする方法があります。

先ほどのWebアプリケーションの場合、Webサーバー、APサーバー、DBサーバーの各層の前段に設けた負荷分散装置から、それぞれの層に含まれるコンテナへリクエストを送信し、期待したレスポンスを返送しない場合は「正常に稼働していない」と判断します。負荷分散と死活監視を組み合わせ、正常に稼働

していないコンテナへの振り分けを負荷分散装置で停止することで、システムが正常に稼働している時間を長くできます。

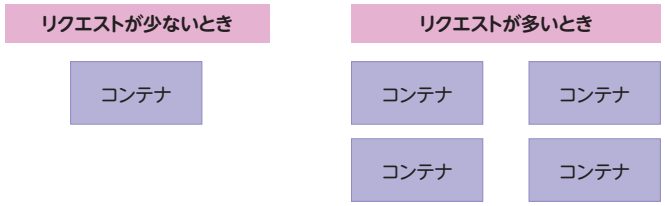
■死活監視



スケーリング

コンテナ技術における**スケーリング**とは、**リクエストの規模に応じてコンテナを増減させること**です。スケーリングを行う手段の1つとして、リクエストの件数などの指標が一定のしきい値を上回る／下回る場合に、リソースの増減を行う方法があります。先ほどのWebアプリケーションの場合、たとえば「直前の1分間にWebサーバー層が受信したリクエスト件数が、コンテナ1台あたり500件を超えるとき」に「Webサーバー層のコンテナを1台増やす」ように設定できます。スケーリングは、リソースの消費量を最適化するとともに、リソース不足によってシステムがダウンするなどの事態を防ぎます。

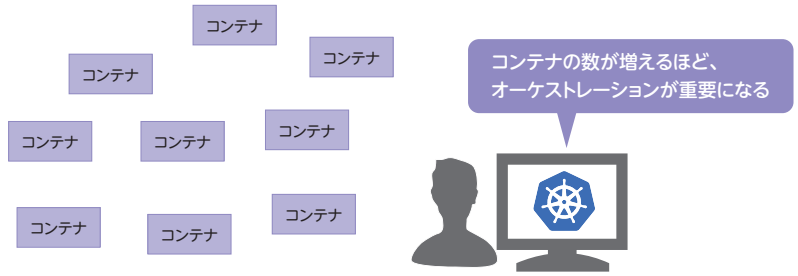
■スケーリング



オーケストレーションの重要性はますます高まる

これまで述べた負荷分散や死活監視の作業は手動で行えますが、管理・運用するコンテナの数が増えれば増えるほど、対応することは困難です。今後、コンテナを活用したアプリケーションの普及や規模拡大とともに、コンテナの管理・運用に共通の作業を自動化するオーケストレーションの重要性は、ますます高まることが予想されます。

■オーケストレーションの重要性



まとめ

- **コンテナオーケストレーションはコンテナの管理・運用を自動化すること**
- **コンテナの管理・運用には負荷分散や死活監視、スケーリングなどの作業が必要である**
- **コンテナを活用したアプリケーションの普及や規模拡大とともにコンテナオーケストレーションの重要性はますます高まる**