クラウドとKubernetes

3.1 はじめに

クラウド関連の情報をウォッチしていると、毎日のようにコンテナ技術にまつわるニュースが流れていることにお気付きでしょうか。キーワードとしてdockerやKubernetes(クバネテスと発音されることが多い)[*1]、CNCFなどを含むニュースであれば、それに類する情報であると考えて間違いありません。更に、今ではdocker、Kubernetesを基盤とするさまざまなプロダクトが生み出され、エコシステムが拡大を続けているため、一見してそれと気付かなくても実はコンテナ系技術の話題であることも珍しくありません。

今後しばらくはメガクラウドベンダを中心にクラウド業界のルールが整えられていくであろうことはある程度否定できませんが、コンテナ技術の興隆を見るにつけ、思ったよりも早く次のトレンドが業界のルールを塗り替えていくのではと思わずにはいられません。もしかしたら、IaaSの登場よりももっと大きなインパクトをIT業界へ与えるイベントが進行しているのかもしれません。

IIJにおいてもこの新しい技術を、ビジネスの迅速な展開や大規模システムの効率的で高品質な運用、ポータビリティの高いソフトウェアの開発、インフラの効率的な利用によるコスト最適化など、幅広く生かすべく活用を始めています。社内で利用されているIKE(IIJ Container Engine for Kubernetes)についてはまた後で触れますが、まずはなぜこんなにも急激にコンテナ技術が注目を集めるようになったのか、そしてこの技術がクラウドへどのような影響を与えると考えられるのかを解説します。

3.2 dockerとKubernetes

コンテナ技術を取り巻くサービスやプロダクトは雨後の竹の子のごとく登場していますが、その中心にあるのはたった2つのプロダクト、dockerとKubernetesです。日進月歩のコンテナ界隈ですが、この2つのプロダクトの動向をキャッチアップすることで、主要な流れを見極めることができるでしょう。

両者の関係は少々複雑ですが、なるべく単純化して表現するとdockerはプログラムをコンテナにくるんで起動するコンテナエンジン、Kubernetesは複数のコンテナエンジンを束ねて制御するコンテナオーケストレータです。一般的にオーケストレータとは連携する複数システムを協調させ、全体を1つの系として統合させるコントローラを示す言葉ですが、Kubernetesのようなコンテナオーケストレータとはコンテナエンジン(≒docker)が稼働する多数のホストノードを束ねて、大きなひとつのリソースプールとして効率的かつ自律的に制御することを意味しています。製品としてのdockerはKubernetesと競合するところもあるのですが、ひとまずコンテナエンジンとコンテナオーケストレータの関係と理解した方が混乱はないでしょう(図-1)。

とはいえ、常に両者がペアで使われるわけではありません。docker単体での利便性は既に広く認知され大いに活用されているものの、Kubernetesがそこまで実環境で利用されているとは言えません。それは、Kubernetesがそれなりの規模で構成されたコンテナクラスタを制御するプラットフォームであるため、試すにしても一定のハードルがあるのに対して、dockerは手元の作業環境を便利にしてくれるユーティリティ

---

*1 dockerに代表されるコンテナエンジンを制御し、多数のノードから構成されるコンテナクラスタを管理するコンテナオーケストレータの1つ。googleによって生み出され、現在はCNCF(Cloud Native Computing Foundation)にホストされるオープンソースソフトウェアである。googleの社内システムであるborgをベースに開発されたとされる。コンテナ化されたアプリケーションのためのランタイム環境であり、インフラに依存しないポータブルなアプリケーションのパッケージング、プロビジョニング、オペレーションが可能になる。クラウド時代のOSと称されることもあり、マルチクラウド、ハイブリッドクラウドのための統一されたオペレーションインタフェースとしても期待される。

Kubernetesを利用するにはインフラに合わせたネットワークドライバやストレージドライバ、トラフィックマネージャなどが必要になるが、Kubernetesにそのような実装は基本的に含まれていない。またKubernetesを環境として整えるには、アプリケーション管理やアカウント管理を行うポータルやモニタリングツールなども事実上必須となるが、そのようなツール類も別途そろえる必要がある。そのため、Kubernetesに周辺の環境を加え、インストーラなどを整備したパッケージであるKubernetesディストリビューションが登場しつつある。本稿で触れているIKE(IIJ Container Engine for Kubernetes)もそうしたKubernetesディストリビューションの1つである。

でもあるので、ちょっとしたユースケースでも分かりやすく便利だからです。おそらく多くのエンジニアがdockerをテスト環境を整えるツールとして、またソフトウェアの配布手段として、便利に利用していることでしょう。dockerは既にエンジニア必携のツールとなりつつあるのです。

ですが、今IT業界を賑わしているのは、どちらかと言えばKubernetesの方でしょう。それは、コンテナ技術がもたらす効果は便利なユーティリティにとどまるものではなく、Kubernetesを活用することでサーバサイドシステムの在り方を大きく変えることが期待されているからです。

まだ成熟しているとは言い難いKubernetesがそこまで注目を浴びているのは、Kubernetesのオリジナルが10年以上に渡ってgoogleのシステムを支え続けているborgだということでしょう。googleの社内システムが詳細に語られることはほとんどありませんが、borgがどのように利用されているのかはSRE(Site Reliability Engineer)Bookを通して一部が明らかにされ、その驚くべき実態が垣間見えるようになりました。これがきっかけとなりコンテナ技術に注目するようになった人も少なくないでしょう。googleのシステムに仮想マシンはなく、基本的にすべてのプロセスがコンテナとして起動されている話は、特にクラウドビジネスに携わるエンジニアに大きなインパクトを与えました。KubernetesはborgのOSS版であるとされていますが、実際にどれほどの共通性があるのかは分かりません。しかし、まだ登場して日が浅く、実績に乏しいと感じられることもあるKubernetesも、そのデザインにはgoogleにおいて長期間利用され、熟成されてきたベストプラクティスが詰め込まれているであろうことは想像できます。

3.3 IaaSを使いこなすための ベストプラクティス

それでは、Kubernetesが変えるサーバサイドシステムの在り方とは何でしょう。曰く、ポータビリティが高く、インフラに依存しないデプロイメントが可能になる。曰く、大規模なクラスタの管理能力を備え、コンピューティングリソースをダイナミックに活用するスケーラビリティに優れる。様々な言葉で表現されますが、曖昧で具体性に欠けるように感じられないでしょうか。それはある意味仕方のないことで、Kubernetesの役割はコンピュータを管理するOSのようなものなのです。OSが何かご存じない方にOSとは何かを説明すれば、やはりとりとめがなく断片的な話になるか、極めてテクニカルなディテールの話になるかどちらかでしょう。

実際のところ、Kubernetesはクラウド時代のOSに例えられることがあります。インフラの差異をドライバで吸収し、ネットワークやストレージのコンフィグレーションを仮想化し、Kubernetes上にデプロイされるシステムへ統一されたインタフェースを提供します。Kubernetesを利用することでIaaS固有のインタフェースに頼らず、統一されたシステム設計、構

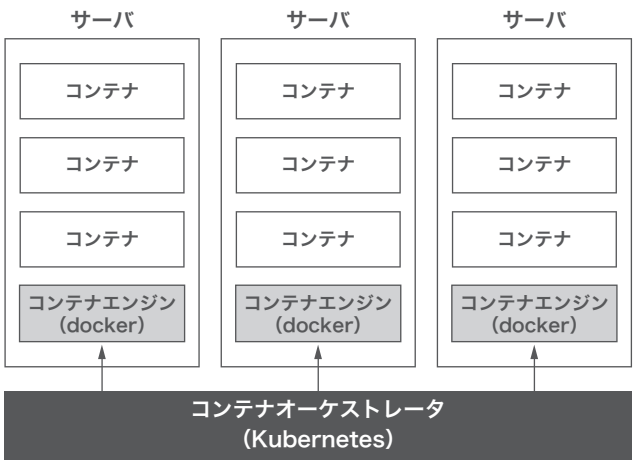

図-1 コンテナエンジンとコンテナオーケストレータ

築、運用が可能になるのです。ただ、KubernetesはOSではないので、アプリケーションインタフェースを提供するわけではありません。Kubernetes上で動くのはあくまでも通常のLinuxやWindowsのアプリケーションです。

似て非なる存在であるOSとKubernetesですが、大きく違うのは前者が物理的な箱に収まった1台のコンピュータを管理するのに対して、後者はネットワークで接続された複数のコンピュータを大きな1つのリソースプールとみなして管理することです。OSによって管理されるプロセスは箱の外へ出ることができませんが、Kubernetesクラスタで動くプロセス(≒コンテナ)はクラスタを構成するどこかのノードで動いてさえいれば問題ありません。だから、リソースが不足すればノードを増強してコンテナを収容替えするだけで済みますし(これは自動的に行われます)、あるノードに障害が発生してコンテナが停止しても、他のノードでコンテナを再起動するだけで復旧します(これも自動で行われます)。

多くの場合、クラウドサービスは止まらない安定性に優れたシステムとみなされています。しかし、実際にはクラウドサービスも様々で、特にIaaSのコンピューティングリソースは冗長化されているわけではありませんから、障害があれば停止しますし、メンテナンスのために計画的に停止されることもあります。IaaSが普及したことでシステムリソースの調達と構築、それにハードウェア障害からの復旧は劇的に短時間で可能になりましたが、多くの運用に関してはさほど変化がありませんでした。仮想化されていようがいまいが、扱う対象が結局はサーバ、ストレージ、ネットワークであれば、オペレーションは大きくは変わりません。IaaSを利用するのは難しくありませんが、そのメリットとされるユーティリティコンピューティングの特性を生かすには、多くの労力を必要とするのが現実です。

そこでKubernetesの出番です。必要なときに必要なだけリソースを確保し、使っただけのコストが発生するIaaSの特性はKubernetesと親和性が高く、動的にリソースを管理することでリソースを無駄なく使ったり、スケーラビリティを生かしたりすることが容易です。また、複数ノードを組み合わせて可用性を保つだけでなく、障害を起こしたノードの復旧をKubernetesへ任せることで、システム全体として影響がない範囲の障害であれば人手を介さずに自動で復旧させることが可能です(図-2)。

コンテナを利用すると従来利用されていたハイパーバイザや仮想マシンといった分厚い管理層が無くなり、単一のOS上にコンテナ化されたプロセスが直接乗るため管理層が薄く、軽くなると説明されることが多いのですが、ただそれだけではコンテナ技術の効果は極めて限定的です。仮想マシンよりもコンテナの方が多くの場合効率的なのは事実ですが、それは手段の話でしかありません。多数のサーバを束ねて大きなリソー

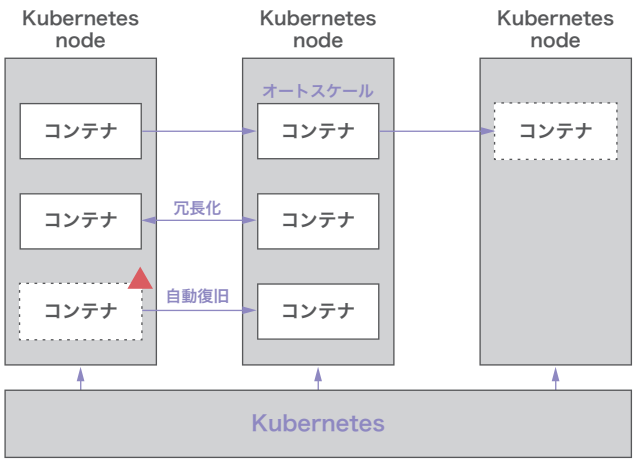

図-2 IaaSを生かすKubernetes

スプールとして構成し、構成情報の管理をKubernetesに任せて自動化することで、初めて本来の効果が発揮されるのです。実際のところ、現時点ではおそらく多くのコンテナが仮想マシンとして実装されているIaaS上で動いていることでしょう。Kubernetesとは、IaaSをより良く使いこなすためのベストプラクティスがつまったパッケージと考えて良いでしょう。

3.4 ハイブリッドクラウドを実現するKubernetes

KubernetesがIaaSの利点をより良く使いこなすためのプラットフォームであるのは事実ですが、IaaS上でしか使えないわけではありません。それどころか、今後を考えるむしろオンプレミスな環境が重視されるワークロードでこそKubernetesに注目すべきです。これまで10年近くに渡りお客様の声を聴き、クラウド関連のレポートを見てきましたが、クラウドが広く普及し、100%オンプレは現実的ではないと考えられるようになっても、100%クラウド化もやはり難しいと回答される方が大多数を占めています。多くのエンジニアがIaaSを手足のように使いこなすようになった今このような意見が多数を占めるということは、今後の在り方が見えてきたのかもしれません。

となると、IaaSとオンプレ環境を適切に使い分ける、ハイブリッドクラウドを本気で考える必要があるということですが、言うまでもなくそれは簡単なことではありません。IaaSはクローズドなシステムですから、オンプレ環境で使うことはできません。仮に使うことができても複雑なIaaSシステムの運用をオンプレ環境で行っては、もはやなんのためにIaaSを利用しているのか分からず、本末転倒な事態になりかねません。プライベートな環境にIaaSと同等の環境を持つにふさわしいワークロードももちろん存在しますが、多くのトレードオフを考慮する必要はあるでしょう。

ですが、KubernetesでIaaSとオンプレ環境の両方をラップすることで、それが現実的な解となるかもしれません。ハイブリッドクラウドを実現するには大きく2つの課題があります。IaaSとオンプレの統一的な管理運用システムとアプリケーションやデータのポータビリティです(図-3)。

コンプライアンス上の理由からIaaSには出せないワークロードやデータ、それにリソースの増減なく長期的に一定の大規模リソースを利用するワークロードはオンプレ環境に置きたいものです。そして、それ以外のワークロードはIaaSに置きたいと考えるかもしれません。ある程度最初からどちらが適しているかはっきりしていれば問題ないのですが、ビジネスの立ち上げ時期はIaaSに、安定時期にはオンプレにといった具合に、時期によって適した環境が変わっていくことも珍しくはありません。前述した2つの課題をKubernetesが効果的に解決することができれば、一気にこの市場が広がる可能性を秘めています。

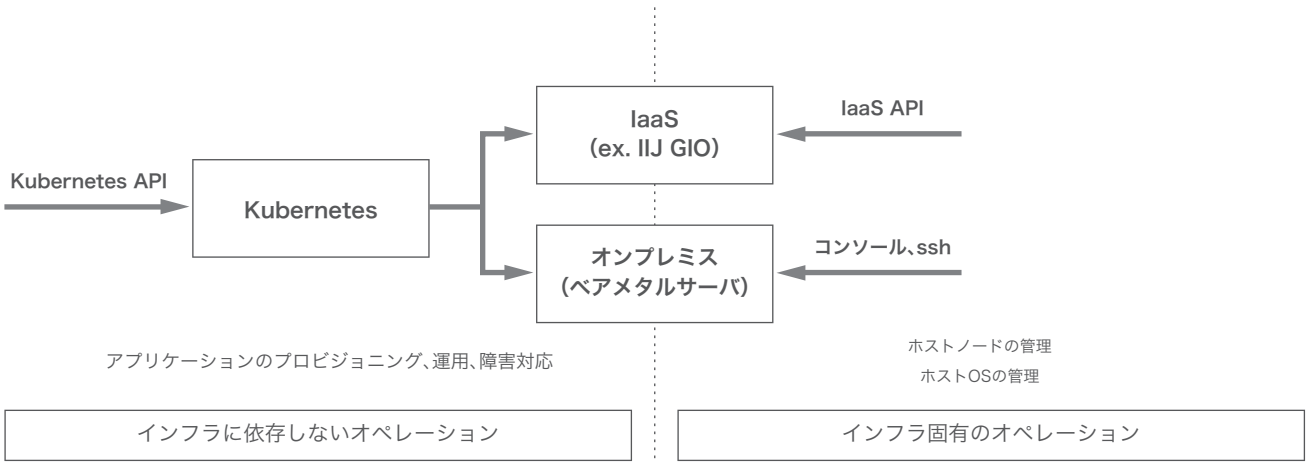

図-3 ハイブリッドクラウドを実現するKubernetse

ただし、現時点ではKubernetesを利用したハイブリッドクラウドはもちろん、オンプレでKubernetesを活用する環境は、IaaS上のそれに比べてまだ成熟しているとは言い難い状態です。あらゆるリソースをAPIを通じてソフトウェアで制御できるIaaSに比べて、オンプレ環境は統一的なソフトウェア制御ができませんから、環境を整えていくにはKubernetesとデバイス双方の歩み寄りと協力が必要でしょう。また、SDN(Software Defined Network)、SDS(Software Defined Storage)といった技術がオンプレ環境で使いやすくなると、オンプレ環境におけるKubernetesの普及に弾みがつきそうです。

ハイブリッドクラウドを実現するには、IaaSとオンプレを管理する共通システムが必要です。Kubernetesがその有力候補であることは間違いありません。

3.5 IKE(IIJ Container Engine for Kubernetes)

このように今後のサーバサイドシステムの設計や運用、アプリケーションの流通とプロビジョニングを根底から変え、劇的に効率を向上させる可能性を秘めたKubernetesですが、IIJにおいてもコンテナクラスタシステムが開発され、その活用は始まっています。IKE(IIJ Container Engine for Kubernetes)と名付けられたこのシステムは、サービスの共通基盤として、また社内システムの運用環境として生み出されたものです。

IKEのようにKubernetesを中心としてコンテナクラスタの環境を整えるパッケージをKubernetesディストリビューションと呼びます。先にKubernetesをOSに例えましたが、KubernetesはOSのカーネルのような存在です。OSがカーネルだけでは何もできず、各種ツールやデバイスドライバを整えたディストリビューション(例:RedHat、Ubuntu)があって初めて役に立つように、Kubernetesをただインストールしても何もできません。インフラに応じたネットワークドライバとストレージドライバが最低限必要ですし、より快適なコンテナクラスタを実現するにはKubernetesを制御するマネージメントツールと、Kubernetes上にデプロイされたアプリケーションをモニタリングしたり、アラートを通知したり、ログを収集したり、運用を支援する環境の整備が欠かせません。Kubernetesクラスタを構成するエコシステムを整え、そうした環境を指定されたインフラに合わせてインストールする機能を備えたパッケージがKubernetesディストリビューションであり、IKEはその1つというわけです。

IKEはお客様へのサービス提供を目的としたものではないため、ある程度動作環境を限定して設計されていますが、自社クラウドサービスであるIIJ GIOはもちろんのこと、オンプレ環境へのインストールが可能です。また、今後は他社IaaS上にもインストールできるようにして、どのようなインフラであっても共通の環境を提供できるようにする計画です。

IIJがKubernetesディストリビューションを実装した理由はいくつかありますが、IaaSの活用だけが目的ではありません。一番の目的はビジネスのスピードアップ、二番目が運用の専門性を高めて高度化、複雑化するシステムに対応することです。こうしてまとめてしまうと抽象的で目的が曖昧に思えるでしょうが、実現したいのは例えばサービスを開発する部

門は開発に、運用部門は運用に専念できる環境を整えることです。これを実現できれば前述の目的はおのずと達成されていくことでしょう(図-4)。

それにしても、仮想マシンのごとく1つのコンテナにサーバ環境をすべて詰め込むのではなく、プロセスを1つずつコンテナに包むようにしただけで、インフラに依存しないサーバサイドシステムのディストリビューションと汎用的な運用システムが実現してしまい、IT業界全体へ影響を及ぼすようなプロダクトが誕生するのですから、この世界は本当に面白いものです。おそらくこれはまだ始まりにすぎず、これからも思いもよらぬアイディアが登場するのだろうと思うとワクワクしますね。

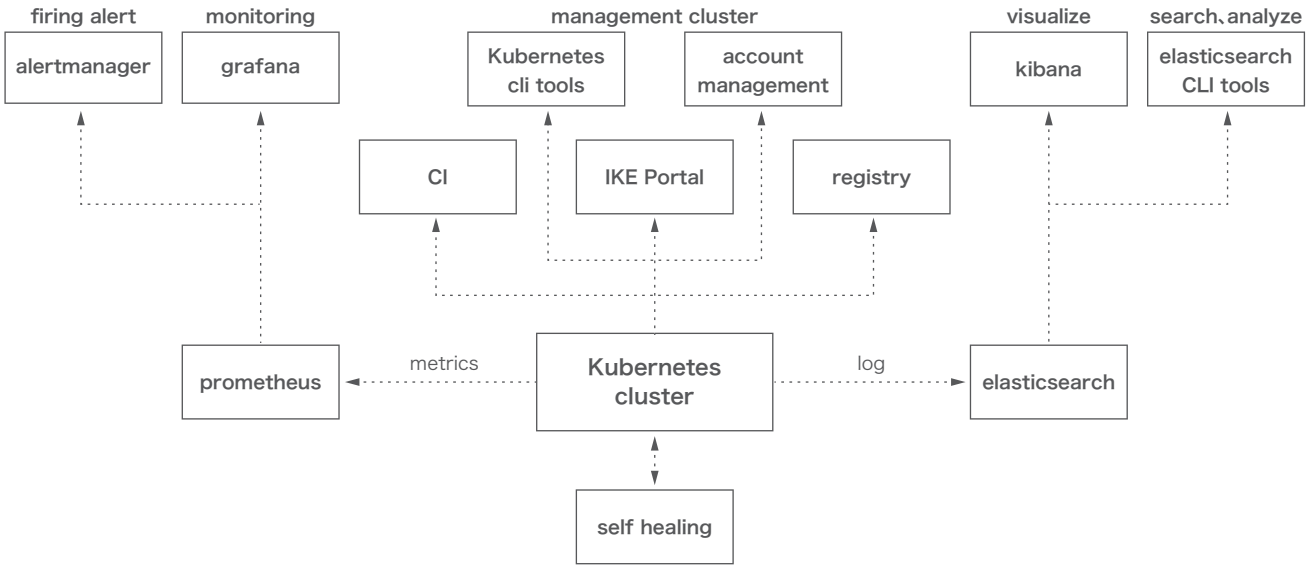

図-4 IKE

執筆者:
田口 景介(たぐち けいすけ)
IIJ サービス統括本部 技術戦略室。
社会人生活の半分をフリーランス、半分をIIJで過ごすエンジニア。元々はアプリケーション屋だったはずが、クラウドと出会ったばかりに半身をインフラ屋に売り渡す羽目に。現在はコンテナ技術に傾倒中だが語りだすと長いので割愛。タグをつけるならコンテナ、クラウド、ロードバイク、うどん。