OPENSTACK

コンテナとOpenStackの活用

包括的なレビュー

Contents

1. 導入
2. I. OpenStackにおけるコンテナのハイレベルビュー
3. II. OpenStack コンテナのインテグレーションポイント
4. III. ケーススタディー
5. IV. まとめ
6. V. オープンソースプロジェクト一覧
7. VI. 執筆者
8. Translators

導入

あなたがプライベートクラウドのインフラを、ゼロからすべて構築する任務を負うことを考えてみてください。あなたは限られた予算、小さくも献身的なチーム、そして奇跡を起こすことを求められています。

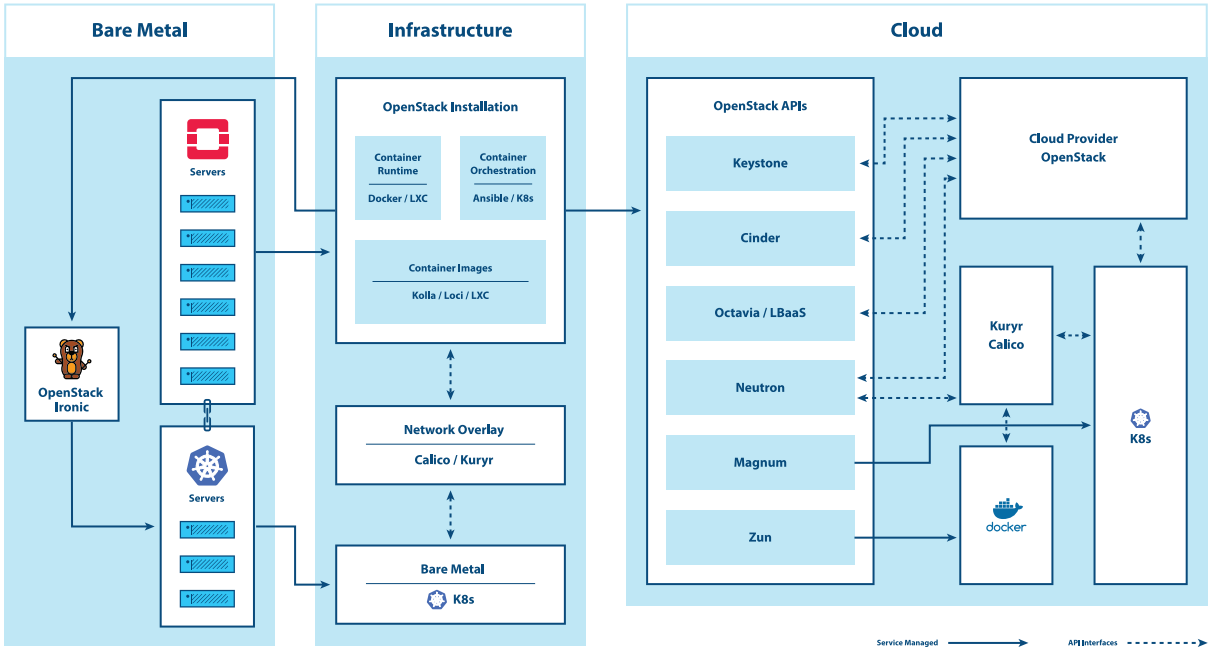
数年前、アプリケーション実行環境を構築するには仮想マシンと、レガシーアプリケーションのため数台のベアメタルマシンを使ったものでした。インフラは進化し、仮想マシンは効率性と迅速性を素晴らしい水準に引き上げましたが、仮想マシン単体ではアプリケーション開発のためのアジャイルアプローチの要求に完全に見合いませんでした。仮想マシンは、引き続き多くのアプリケーション実行の基盤として提供されるでしょう。ですが、斬新なアプリケーション開発と適用のために、開発者はますますコンテナの最新動向を追いかけています。なぜなら、コンテナは迅速性と効率性の水準を向上させるからです。

DockerやKubernetesのようなコンテナ技術は、コンテナ化されたアプリケーションを構築するための先行標準になりつつあります。これらの技術は開発の迅速性を制限する複雑さから組織を開放する助けとなります。コンテナ、コンテナインフラ、そしてコンテナ展開技術は、いくつもの異なった利用例に適用可能で、大変パワフルな抽象化能力があることを、自ら証明してきました。Kubernetesのようなインフラを使うことで、アプリケーションデリバリにコンテナだけを利用するクラウドを実現できます。

しかし、最新のプライベートクラウドはコンテナだけではなく、コンテナはすべてのワークロードや利用方法に向いているわけではありません。今日、多くのプライベートクラウドインフラは、インフラ管理のためのベアメタルマシン、古いアプリケーションのための仮想マシン、そしてより新しいアプリケーションのためのコンテナを取り扱う必要があります。これら三つのアプローチに対応し、管理し、おーけストレートする能力は、運用効率化のカギとなります。

OpenStackは、プライベートクラウドを構築するための現在最高の利用可能なオプションです。ネットワーク、ストレージ、そしてコンピュートインフラをマネージする能力があり、さらに仮想マシン、ベアメタル、そしてコンテナを一つの制御プレーンからサポートします。Kubernetesは、最も人気のあるコンテナオーケストレーターであり、アプリケーションデリバリを変えてきましたが、堅牢なクラウドインフラの可用性に依存しており、OpenStackはアプリケーションをホストするため、もっとも包括的なオープンソースインフラを提供します。OpenStackのマルチテナントなクラウドインフラは、いくつかのインテグレーションポイント、デプロイメントソリューション、そして複数のクラウドをまたがり連携させる能力など、Kubernetesと相性がよいものです。

この文書では、コンテナがどのようにOpenStackと作用するか探索していきます。多様なユースケースの検証や、オープンソースプロジェクトの概要について、OpenStackに限らず、コンテナを適用しやすく、効率化しやすくするその他の技術についても紹介します。



I. OpenStackにおけるコンテナのハイレベルビュー

コンテナとOpenStackが関係する三つの主要なシナリオがあります。

最初のシナリオは、インフラコンテナと呼ばれるもので、オペレータがクラウドインフラの適用や管理、運用をしやすくするため、コンテナを利用します。このシナリオにおいてコンテナは、ベアメタルインフラ上にセットアップされ、ホスト資源への特権アクセスを許可されます。このアクセスはコンピュート、ネットワーク、ストレージといった資源に対する直接的なアドバンテージを、コンテナに許可します。これらの資源は通常、コンテナランタイムにより、ユーザから隠ぺいされるものです。コンテナは、しばしば複雑になるアプリケーションが必要とする依存関係を隔離します。一方で、インフラアプリケーションに対しては、基盤となるシステム資源の管理や操作を直接行うことを、依然として許可します。サービスのアップグレードが必要になったとき、一緒に配置されるサービスを破壊するような依存関係の変更を行うことなく、アップグレードは行われます。

最近のOpenStackのバージョンは、このインフラコンテナモデルを取り込んでいます。そして現在、オーケストレーションツールとコンテナ化されたサービスの組み合わせと共に、OpenStackの全ライフサイクルを管理する一般的な方法になっています。インフラコンテナは、コンテナオーケストレーション技術を使い様々な課題をオペレーターが解決することを実現します。とりわけOpenStackを含む既存のソフトウェアを高速に反復しアップグレードする際に発生する課題です。OpenStackをコンテナの中で実行することは、オペレーターがDay 2に発生するチャレンジを解決するのに役立ちます。これには、サービスのための新しいコンポーネントを追加したり、速やかにソフトウェアバージョンを更新したり、マシンやデータセンターをまたがって高速なローリングアップデートを行うことを含まれます。このアプローチは、OpenStackの展開や更新の問題に対し、コンテナの迅速性をもたらします。

2番目のシナリオは、クラウドインフラ上でコンテナ化されたアプリケーションフレームワークをホスティングする場合に関してです。これは、Docker SwarmやKubernetesといったコンテナオーケストレーションエンジン（COE）、軽量のコンテナにフォーカスしたサービス、そしてサーバレスなアプリケーションプ

ログラミングインターフェイス（API）を含みます。OpenStackコミュニティは、ベアメタルやVMを問わず、セキュアでテナント隔離されたクラウドホストの上で、コンテナ化アプリケーションの提供が可能となるよう努力してきました。このシナリオでは、ストレージやロードバランシング、アイデンティティといったOpenstack APIの利点を、Kubernetesのようなプロジェクトにもたらすドライバによって促進されます。これらの機能によって、開発チームは新しいコンテナ化アプリケーションを書き、速やかにKubernetesクラスタをOpenStackクラウド上に展開できます。これは、彼らのアプリケーションを商用環境へデプロイする強固な自動化を伴う、開発やテストそしてコードのデバッグに必要なリソースを提供する、完全なアプリケーションライフサイクルを提供するソリューションです。

最後のシナリオは、独立したOpenStackとCOEのデプロイの間の相互作用を考えます。このホワイトペーパーでは特に、Kuberunetesクラスターのことについて記述します。OpenStackとKuberunetes双方にまたがるAPIの一致制と相互運用性は、このシナリオにおける主要な成功の源泉です。例えば、Kuberunetesに OpenStack Cinderがホストするボリュームを直接アタッチすることは可能ですし、OpenStack Keystoneを認証と認可のバックエンドとして利用することも可能です。さらに、OpenStack NeutronをOpenStack Kuryrと共に使って接続させ、ネットワークオーバーレイとして利用することも可能です。逆に言えば、Openstackクラウドにとって、Calicoのようなプロジェクト用のNeutronドライバーを使い、Kuberunetesクラスタに対して、同じネットワークオーバーレイを共有することが可能なのです。3番目のシナリオは、クラウドサービスの提供方法（KuberunetesやOpenStack）にさほど注目せず、いかに独立したサービスが相互作用するかをより注目します。

II. OpenStack コンテナのインテグレーションポイント

コンテナ上へのOpenStackインフラのデプロイ

最初にお伝えしておきたいことは、OpenStackはコンテナの普及にともない大きな変化をしてきたということです。なぜならば、コンテナはインフラストラクチャコードの管理に対し、新しいアプローチを可能にしたからです。以前の管理ストラテジーには、重い黄金のマシンイメージの作成と管理の双方が必要でしたし、もろい状態維持のためのコンフィグ管理システムが必要でした。難易度を上げていたものの正体は、サービス集合の管理です。これはどれも、リリースとリリースの間に変化する、自分自身の依存関係を必要としました。アプリケーション隔離のいくつかのやり方なしには、不可能でないにしても依存関係の解決は難しくなります。

インフラストラクチャコンテナは、OpenStackデプロイメントの新しいプロジェクトが、エレガントにこの依存関係の問題を解決するという両立を実現可能にしました。軽量、独立、自己完結、そして特徴的なこととしてステートレスであるアプリケーションを使うことで、クラウドオペレーターは複雑な制御プレーンを展開する際に、ものすごい柔軟性を得ることが出来ます。コンテナランタイムとオーケストレーションエンジンを結合されることで、インフラストラクチャコンテナは、複雑で高い可用性を持つインフラの迅速な展開、保守、そして更新を可能にします。

OpenStackクラスターの構築において、デプロイメント技術の選択にはいくつか考慮点があります。オペレーターは Linux Containers（LXC）やDockerを彼らのベースコンテナとして選択し、事前構築済みかカスタム構築されたアプリケーションコンテナを利用し、オーケストレーションのための伝統的なコンフィグ管理システムか、あるいはKuberunetesのようなよりモダンなアプローチを選択することが出来ます。表1には、既存のOpenStackデプロイメントプロジェクトと、それらの基礎的な技術を要約しています。

Project	Container Type	Supported Containers	Project
OpenStack-Ansible	LXC	OSA LXC Containers	Ansible
Kolla-Ansible	Docker	Kolla Containers	Ansible
Triple-O	Docker	Kolla Containers	Ansible
OpenStack-Helm	Docker	Kolla Containers Loci Containers	Kubernetes and Helm

Table 1

基礎的なデプロイメントシステムはそれぞれ、OpenStackコードの一組のコンテナを構築するため、そしてサービスをサポートするために、異なるアプローチをとっています。 OpenStack Ansible（OSA）とKollaプロジェクトは、LOCIがプロジェクトアプリケーションコンテナの構築に焦点を当てる一方、特定のオーケストレーションシステムを考慮することなく、彼ら独自のプロジェクトがホストする構築システムを持っています。ハイレベルな観点からの差分は:

1. OSAは、低レベルのLXCコンテナを利用するという点で特異なものです。そしてLXCアプリケーションコンテナの作成のため、カスタム構築システムを持っています。
2. Kolla構築システムは各サービスごとに一つDockerコンテナを生成し、OpenStack展開の初期化と管理を担うコンテナのサポートを前に進めます。Kollaコンテナは、基礎になるオペレーティングシステム、ソースやパッケージのインストール、そして更なるカスタマイズのためのテンプレートエンジンの選択を行うことで、高度に設定可能です。
3. OpenStackアプリケーションコンテナを構築する最後の選択肢は、LOCIです。LOCIはDockerコンテナを構築し、それぞれのプロジェクトに対てコンテナを一つずつ提供することもしています。LOCIは、コンパクトでセキュアなコンテナを迅速に生成することに焦点を当てています。これは、すべての共通ディストリビューションのため、デプロイメントシステムによる構築の、基礎的な利用を期待されているからです。

ベアメタルインフラストラクチャ - OpenStack とブートストラップ問題の解決

すべてのクラウドの基礎には、インフラストラクチャサービスを提供するベアメタルサーバのデータセンターが存在します。「サーバーレスコンピューティング」であってもデータセンター内のハードウェア上にあるクラウドの上で、ソフトウェアを実行しています。ハードウェアインフラストラクチャの起動法に関する問題は、致命的な問題です。OpenStackソフトウェアは、クラウド的な品質をベアメタル管理に与える方法を取り扱うため、比類なく選出されるものです。

OpenStack Ironicは、bare-metal as a serviceを提供します。スタンドアロンなサービスとして、Ironicはベアメタルノードを見つけられ、それらを管理データベース上でカタログにし、すべてのサーバライフサイクル、すなわちエンロール、プロビジョン、保守、そして廃止を含んだ管理を行います。

OpenStack Novaのドライバとして使うとき、そしてOpenStackサービスの一式と紐づけられるとき、Novaはすべてのベアメタルインフラストラクチャを管理するため、パワフルでクラウド的なサービスを実現します。

このことは課題を提起します：どのようにして、一つのブートストラップOpenStackサービスは、ベアメタルインフラストラクチャを管理するのでしょうか？ひとつの典型的なソリューションは、種を設置するため、前節で記したコンテナベースのインストールツールと同じようなものを利用するというものです。この種は、しばしば「アンダークラウド」と呼ばれますが、あたかも仮想化クラウドであるかのように、ベアメタルクラスターの管理を完全に自動化するために利用できます。

これは、単にOpenStackによる仮想化をベアメタルクラウド上で実行するだけに留まらず、ベアメタル環境にKuberunetesだけのインストールを実行するのにも使えます。これにより、アイデンティティ、ストレージ、ネットワーク、そして他のOpenStackサービスを通じ利用可能なクラウドAPIのすべての利点を使うことが出来ます。

OpenStack上でのコンテナベースアプリケーションの提供

コンテナとベアメタルインフラストラクチャの、どちらのインフラストラクチャも重要です。しかし、多くの人はコンテナのことを考えたとき、アプリケーションコンテナのことを考えているようです。隔離、カプセル化、保守の容易さなどコンテナがもたらす特徴は、アプリケーションを供給するにあたり、アプリケーションコンテナを理想的なソリューションとするでしょう。しかしながら、ベアメタル、パブリッククラウド、プライベートクラウドのいずれかから、ホストサーバのプラットフォームを提供するために、コンテナは依然として必要です。

Kuberunetesはアプリケーションを提供するプラットフォームであり、永続ストレージやロードバランサー、ネットワーク、そして動的配置のコンピュートノードのような、クリティカルなインフラの提供を自動化する、クラウドAPIと共によく作用します。OpenStackは、オンプレミス型のプライベートクラウド、あるいは利用可能なパブリックもしくはマネージド型のOpenStackクラウドを通じ、クラウドインフラを提供します。

OpenStackは、Kuberunetesのための最初の上流クラウドプロバイダでした。これは、活動的な開発チームが"Kubernetes/Cloud Provider OpenStack"プラグインを保守していたことによるものです。このプラグインにより、KuberunetesはCinderブロックストレージやNeutron、Octaviaロードバランサー、そしてNovaを使ったコンピュート資源の直接的な管理といった利便性を活用することが出来ます。Kubernetes環境にドライバーをデプロイするのと同じくらいシンプルに、プロバイダーを用いることで、ドライバをロードするようフラグを設定し、ローカルユーザのクラウド認証を提供します。

KuberunetesやほかのアプリケーションフレームワークをOpenStack上にインストールするソリューションはいくつかあります。コンテナフレームワークを実現するもっとも簡単な方法の一つは、Magnumを使うことです。これは、Kuberunetesを含む、いくつかのアプリケーションプラットフォームの選択肢を背景にした、完全なマネージドクラスタをデプロイするシンプルなAPIを提供します。これはOpenStack APIと、クラウドプロバイダプラグインを利用するKuberunetes展開の例です。例えば、200以上の独立、または連携したCERNのOpenStackオンサイトクラウドや、パートナークラウド上にもあるKuberunetes環境において使用中です。もしあなたがお好みのOpenStackクラウドでMagnum APIを利用可能でなければ、あなたは他のKuberunetesインストールツールを利用することが出来ます。それは、 kubeadm、Kubernetes Anywhere、Cross-Cloud、またはKubesprayといったツールを、あなたのKuberunetesクラスタをOpenStack上にインストールしたり管理するため使うことが出来ます。これらはそれぞれ標準Kuberunetesを使うため、ストレージやロードバランスの利便性を得るためクラウドプロバイダインターフェイスを有効にするのも簡単です。

Zunは、もう一つのOpenStackプロジェクトであり、サーバやクラスタの管理をすることなく、個々のコンテナを管理する軽量コンテナサービスAPIを提供します。OpenStackにホストされたKuberunetesクラスタは、弾力的です。なぜなら、Nova APIを直接通じてクラウドリソースをクラスタに追加、あるいは削除することにより動的に大きさを変えるからです。その他にも、KuberunetesはOpenStack Zunに対しコンテナバックエンドとして使われ、Podインフラストラクチャの管理をZunで行います。軽量でマルチテナントなコンテナサービスAPIを、直接サーバを作る必要なくコンテナを稼働するために提供します。NeutronとCinderの直接インテグレーションは、ネットワーキングとボリュームを個々のコンテナのために提供するため使われます。

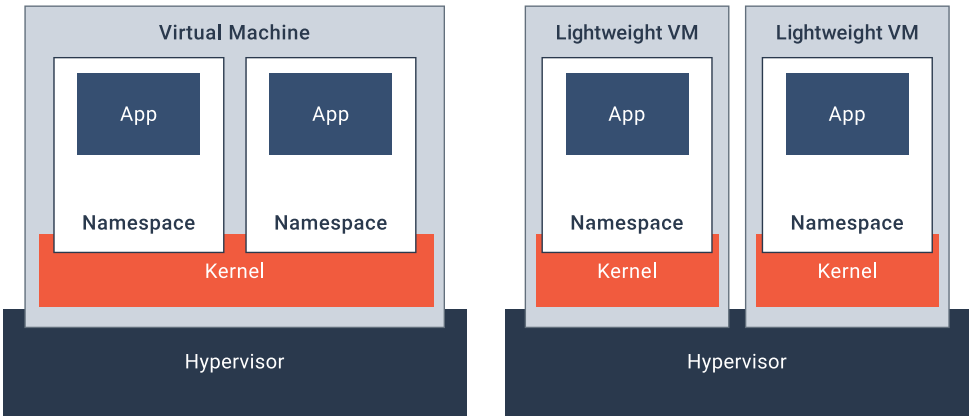
最後に、QinlingプロジェクトはLambda、Azure Functions、Google Cloud Functionsと似たようなサーバレス機能をサポートする、プラットフォームの実現に焦点を当てた「Function as a Service」を提供します。これはコンテナの管理をさらに抽象化し、イベント駆動、オンデマンドにスケールするサーバレスコンピュートの体験と共に、ユーザの開発を加速します。Qinlingは、KuberunetesやDocker Swarmといった、異なるコンテナオーケストレーションのためのバックエンド、多様な人気のあるファンクションパッケージ、ローカルストレージやOpenStack Swiftのようなストレージバックエンドをサポートします。

Kata Containers - 仮想化を通じたセキュアなアプリケーション

Kata コンテナは新しいオープンソースプロジェクトであり、この新規実装は、コンテナエコシステムの中でシームレスに統合された軽量仮想マシンの新しい実装です。Kata コンテナは、コンテナのように軽量で速く、コンテナ管理層と共に統合されます。この中には、DockerやKubernetes (k8s) といった、よく利用されるオーケストレーションツールがあります。さらに、VMのセキュリティ優位性もまた提供されます。Kataコンテナは、Open Container Initiative (OCI)標準に従っています。この標準は、OpenStackファウンデーションがアクティブに活動しているものです。Kataコンテナは, OpenStackファウンデーションにホストされてますが、OpenStackプロジェクトから分離されたプロジェクトとして、自治の仕組みと自身のコミュニティを持ちます。

業界のコンテナへの以降は、信頼されたワークロードと信頼されないワークロードが混在するマルチテナント環境の中で、ユーザのワークロードを確保するという独自のチャレンジが存在します。Kata Containersは、ハードウェアを背景にした隔離を、各コンテナやポッド内のコンテナの集合のための境界として利用します。このアプローチは、伝統的なコンテナアーキテクチャにおける、共有カーネルに関するセキュリティの関心に焦点を当てたものです。

Kata Containersは、長時間実行されるWebサーバーアプリケーションだけでなく、継続的インテグレーション／継続的デリバリーなどイベントベースとオンデマンドのデプロイメントに最適です。 また、従来の仮想化された環境のコンテナへの移行も簡単に行えます。従来のゲストカーネルやデバイスパススルー機能をサポートしています。 Kata Containersは、セキュリティ、スケーラビリティ、リソース使用率の向上を実現すると同時に、全体的に単純化されたスタックにつながります。



Containers in Cloud Today
(Shared kernel, isolation within namespace)

Kata Containers
A lightweight virtual machine isolates each container/pod and provides a separate kernel for each container/pod.

OpenStackとKubernetesの並列したインテグレーション

オープンソースプラットフォームを選ぶ主要な利益の一つは、これらのプラットフォームの標準デプロイメントにまたがる、インターフェイスの安定性にあります。OpenStackファウンデーションとCloud Native Computing Foundation (CNCF) の双方は、OpenStackクラウドとKuberunetesクラスタの間に相互運用性標準を維持しています。これには、ライブラリやアプリケーション、そしてドライバがすべてのプラットフォームをまたがり、どこにデプロイされたかに関わらずに動作することを保証します。これにより、並列したインテグレーションの機会を作り、OpenStackとKuberunetesの双方にお互いに提供されるリソースのアドバンテージをもたらします。

KuberunetesコミュニティのOpenStack Special Interest Group （SIG-OpenStack）は、クラウドプロバイダOpenStackプラグインを保守しています。さらにOpenStack上にKuberunetesを稼働させるクラウドプロバイダインターフェイスの追加により、Kuberunetesが個々のOpenStackサービスのアドバンテージを利用するいくつかのドライバを保守します。このドライバには以下のものが含まれます。

- 二つのスタンドアロンなCinderドライバ。フレックスボリュームドライバは、ドライバに接続するために実行ベースのモデルを使います。そしてコンテナストレージインターフェイス（CSI）ドライバは、コンテナオーケストレーションシステムのために標準インターフェイスを使い、任意のストレージシステムをコンテナワークロード向けに提供します。70以上のストレージドライバのサポートにより、これらのドライバは、単一のCinder APIを通じて、バトルテストを経たプロプライエタリとオープンソースストレージの資産に接することが出来ます。
- Keystoneに向けのWebhookベースの認可と認証インターフェイス。各モード、認可と認証、はそれぞれと独立して構成することが出来ます。作業は進行中ですが、このインターフェイスは、OpenStack Keystoneを使いKuberunetes RBACを支援する、緩やかなマルチテナンシをサポートします。

OpenStackとKuberunetesの双方は極めて動的なネットワークモデルをサポートします。これは様々なドライバに支援されたものです。これらの標準的なネットワークインターフェイスにより、スタンドアロンのOpenStackとKuberunetesクラスタを強固なネットワークのインテグレーションと共に構築することが容易になります。OpenStackの中では、Kuryrプロジェクトが共通ネットワークインターフェイス（CNI）を生成し、NeutronとDocker、Kuberunetesを接続します。一方で、Calicoのような外部のプロジェクトは、標準的なNeutron APIを通じ人気のあるKuberunetesネットワークオーバーレイへの直接的なアクセスを提供します。

III. ケーススタディー

OpenStackコミュニティの多くのメンバーは、コンテナに関係するさまざまなOpenStackプロジェクトに新しいコードを提供し、コンテナの意味と利点を評価し、生産中のコンテナを使用して課題を解決し、新しい機能をリリースしています。 このセクションでは、最も興味深い事例をいくつか紹介します。

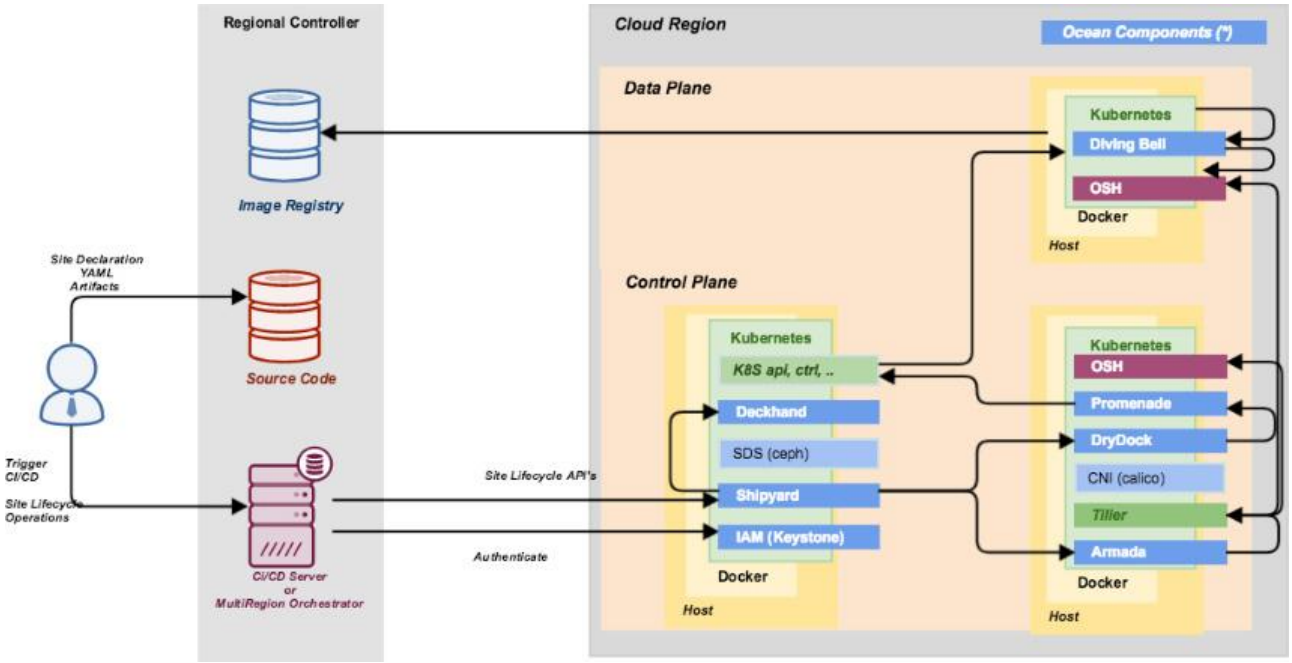
Select a Case Study

- AT&T
- Cern
- SK Telecom
- 超流動

AT&T

AT&T、世界最大の通信会社の一つは、コンテナOpenStackの上で自分の5Gのインフラを構築することを目的として、シンプルさと効率性を生成するためのインフラストラクチャのコンテナに頼って、OpenStackの自分自身を展開し、管理するためのコンテナ技術を活用しています。

彼らの目標を達成するために、AT&Tは、OpenStack-Helmプロジェクトを使用して、Kubernetesクラスターの LOCIベースのOpenStackイメージを編成し、Kubernetes、Docker、およびコアとなるOpenStackサービスを活用しています。 彼らは、Bandit、Tempest、Patrole、その他多くのOpenStackプロジェクトも使用しています。 AT&Tはまた、コミュニティで協力して、 Airshipと呼ばれるundercloudのプロジェクトを開始しました。これは、ベアメタルからOpenStackワークロードを実行しているプロダクショングレードのKubernetesにクラウドを提供します。



AT&Tはコンテナ化が伝統的なデプロイメント活動を左よりにし、CI/CDを使って正当化することを発見しています。Kuberunetesはまた、きわめて大規模なスケーラビリティと回復力を提供します。さらにOpenStack-Helmを使い宣言的に運用行動を設定したり、コンフィグを挿入したり、ローリングアップグレードやテナントのワークロードに影響を与えることなく、更新をかけたりすることも出来ます。

OpenStackのデプロイと運営のためにコンテナ技術の活用することは、テナントに明らかな影響をもたらしません。コンテナ技術がより高い回復力のあるプラットフォームを持ったり、クラウドの特徴を最小限の割り込みと共に頻繁に得ることが出来る、といった例外を除いては。AT&Tの運用チームのこの新しい経験は、彼らの努力を宣言的なサイト向けに使うコンフィグ定義や、コンフィグをデプロイするKuberunetes志向の自動化に導きました。

AT&Tはこのアーキテクチャを利用し、消費者とビジネスに焦点を当てた製品やサービスのバックボーンを仮想化を用いて強化することを目指しています。AT&Tのコンテナ化されたネットワーククラウドの最初のユースケースは、新しい5Gネットワークのための VNFを初期展開することです。OpenStackはこれまでも、現在も、そしてこれからもAT&TのVNFに焦点を当てたクラウドのユースケースとして最適なものになっています。コンテナ化は、AT&TがOpenStackインフラの展開、管理、および拡張の信頼性を高め、迅速かつゼロタッチな方法で行うことを可能にする進化です。

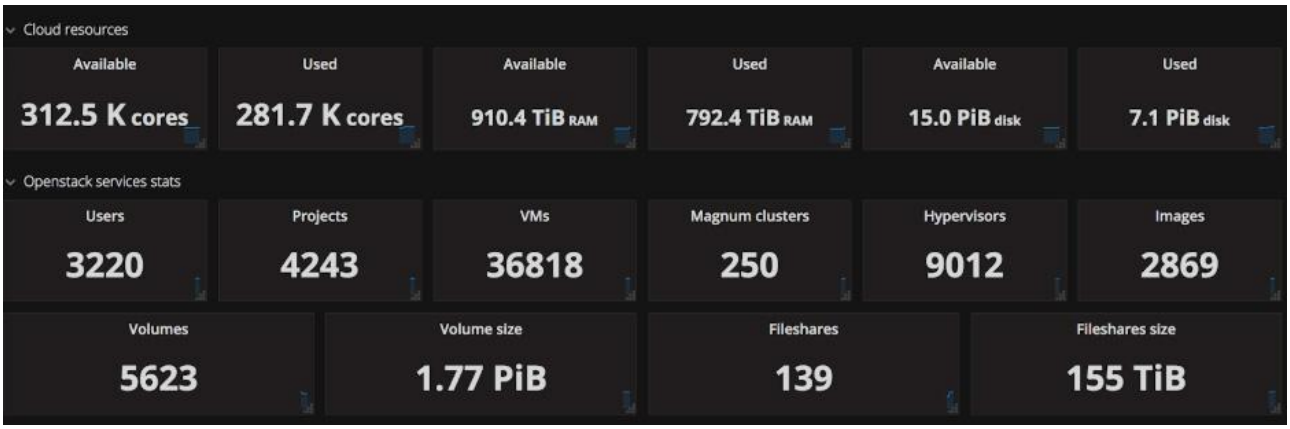
運用にあたって、AT&Tはこのアプローチを検証している段階ですが、年末までに5Gの商用サービス開始を宣言しています。OpenStackとコンテナ技術はこのサービスのバックボーンを形成し、何百人ものAT&Tユーザにとって戦略的に重要なものとなります。5Gサービスの展開は大規模に分散している本番環境でのOpenStackとコンテナ技術の妥当性が実証されます。

Cern

CERN、ヨーロッパの核研究組織は、物理学者やエンジニアは、世界最大かつ最も複雑な科学機器を使用して、物質の基本成分、すなわち基本粒子を研究することによって、宇宙の基本構造の研究を可能にしています。

CERNは2013年からOpenStackを実運用しており、現在は単一のクラウド内で仮想マシン、ベアメタル、コンテナのサービスを提供しています。コンテナは、OpenStack Magnumを介してプロビジョニングされた、使用例に応じて仮想マシンまたはベアメタルのいずれかで実行されます。 Kubernetes、Docker Swarm、DC / OSなどさまざまなコンテナ技術が利用可能です。

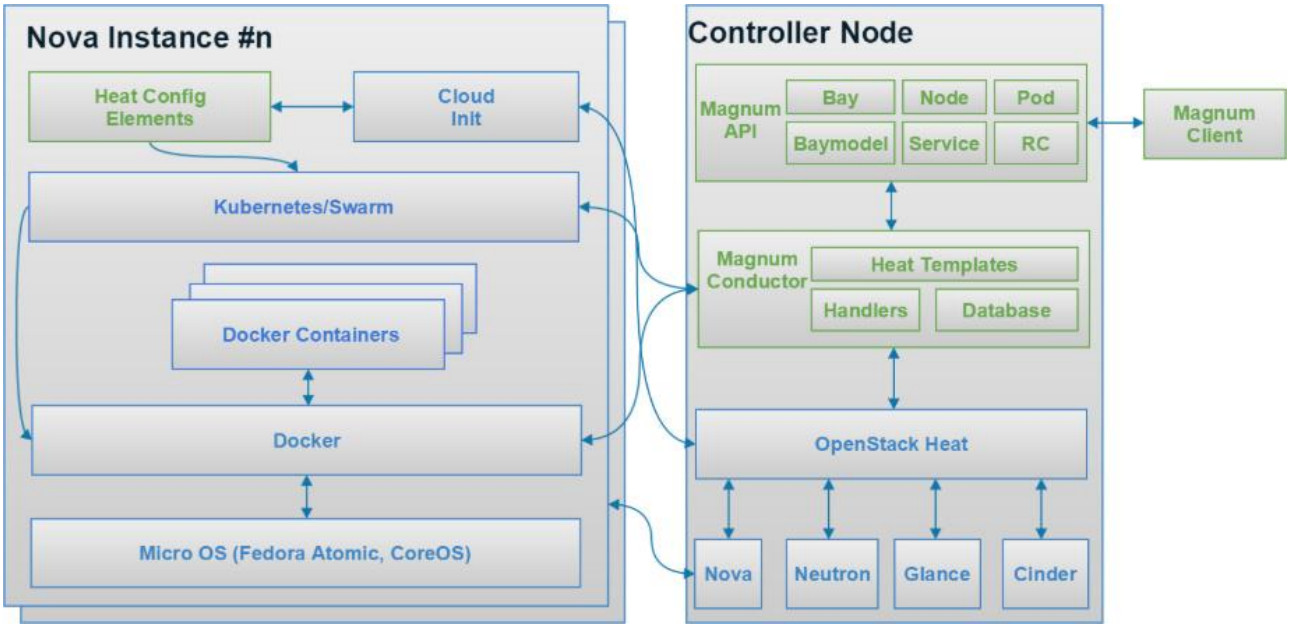
CERNは、現在、OpenStackの上にMagnumを通じてプロビジョニングされた250のコンテナクラスターを実行しています。



CERNのOpenStackクラウドは、ユーザーにいくつかのコマンドまたはWeb GUIを使用して、設定されたコンテナエンジンを要求するセルフサービスアクセスを提供します。 これにより、技術の迅速な利用が可能になり、必要に応じて1000sのノードまで拡張できます。 ベストプラクティス構成は、CERNのストレージと認証サービスへの統合された監視と統合によって利用できます。

このリソースプールを効率的に稼働させるには、余分な操作人員を必要とせずにスケーリングし、一貫した管理プロセスとツールが必要です。 Magnumを介してOpenStack上にコンテナを追加することで、ハードウェア修復プロセスや一貫性のある認証モデルなど、以前に開発したオートメーションを使用しながら、ユーザーのニーズに応じてリソースの迅速な再割り当てをサポートすることができました。

公的資金で設立された研究所として、 KubernetesやOpenStackなどのオープンソースに対し、他の組織と協力してコミュニティに還元するためのフレームワークを提供しています。CERNは CERN openlab framework を通して、RackscaleやHuaweiなどの多くのベンダーとともにMagunumやfederationといった機能を大規模に提供しています。これらの知見はOpenStack Special Interest Groups, the Square Kilometer Array (SKA), Kubecon Europe, OpenStack in Productionなどで共有されています。



CERNでは、いくつかのワークロードがMagnumによってプロビジョニングされたコンテナ内で実行されます。

- Reana/Recast
  - これらのツールは、高エネルギー物理学の分野において、再利用可能なワークフローを実行するためのワークフローを提供します。コンテナは、分析ソフトウェアとデータをパッケージする機能を提供します。この機能は、単独かつ容易に共有でき、さらにオンプレミスでも外部資源を利用しても容易にスケールアウトできます。仕事は、分析とデータ保護をサポートするYadageワークフローに基づき、Kuberunetesジョブとしてスケジュールされます。
- Spark as a Service
  - 最近、KubernetesはSparkのリソースマネージャに追加されました。Sparkは、driverとexecutorをpodとして生成することができ、Kubernetesはそれらのスケジューリングとライフサイクルに責任を持ちます。CERNのIT部門のチームは、ユーザがKubernetesクラスタを欲しい時にOpenStack Magnumを使って作り、SparkをKubernetes上に展開します。そして、CERNの特性ファイルシステムとデータ源を安全な方法で使いながら、必要となるすべてのインテグレーションが提供されます。ユーザはいくつかのコマンドを使い、Spark環境を要求したサイズだけ効率的に作成することが出来ます。彼らの環境を実行中にスケールアップまたはダウンするオプションも付属しています。
- 大型ハドロン衝突型加速器（LHC）のアップグレードのための、LHC実験検出器トリガーシミュレーション
  - LHCは2020年代の間により高いルミノシティにアップグレードする必要がある。これは衝突をフィルタする実験トリガーファームにおける、重要な発展に必要となる。大規模Kuberunetesクラスタは、ATLAS実験のための異なるアプローチをシミュレートするために作られ、設計を確認します。結果的にKubernetesとOpenStackコンポーネントのファインチューンをもたらします。
- Gitlab継続的インテグレーションランナー
  - Gitlabを使いユーザは、CI/CDジョブの構築とそれらの実行を共通、あるいはプロジェクト固有のランナー上で行います。CERNユーザは、CERNコンテナサービスを使い、ソフトウェアのテストと構築を有効にすることができ、コンテナイメージやドキュメントを構築や発行し、

もしくは完全なアプリケーションライフサイクルを管理する複雑なパイプラインをセットします。これには、異なる環境への自動デプロイメントを含みます。

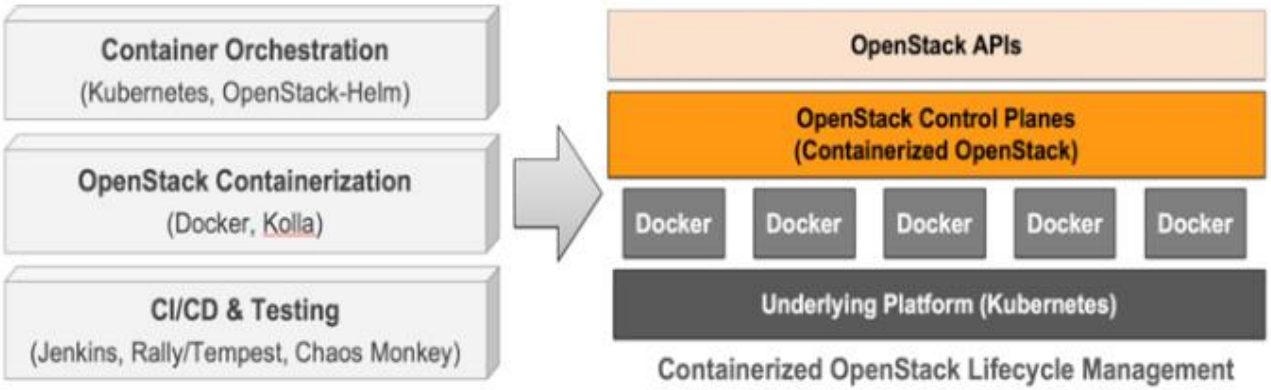
- 外部クラウドと連携したKubernetesコンピュートファーム
  - CERNは、マルチクラウドの運用をサポートするため、Kuberunetesクラスタのフェデレーションを利用しています。複数のクラスタは、AWSやGCE、そしてCERNやKubecon 2018でデモされたT-Systems Open Telecom CloudといったOpenStackクラウドのように、様々な技術のクラウドにまたがってシームレスに構築されています。

仮想マシンとコンテナエンジン、そしてベアメタルを一つのフレームワークの下で統合することは、アカウンティングやオーナ権限、クォータを簡単に確認するために提供されます。KuberunetesのためのManilaストレージドライバは、ファイル共有のための透過的なプロビジョニングをもたらします。これは、彼らのワーキンググループ向けの優先度定義のために行われる、容量計画とリソース調整経験の双方において、IT部門をサポートします。スタッフの離職に伴うリソースの再調整や失効といったリソース管理ポリシーは、矛盾のないワークフロー上で執り行われます。

SK Telecom

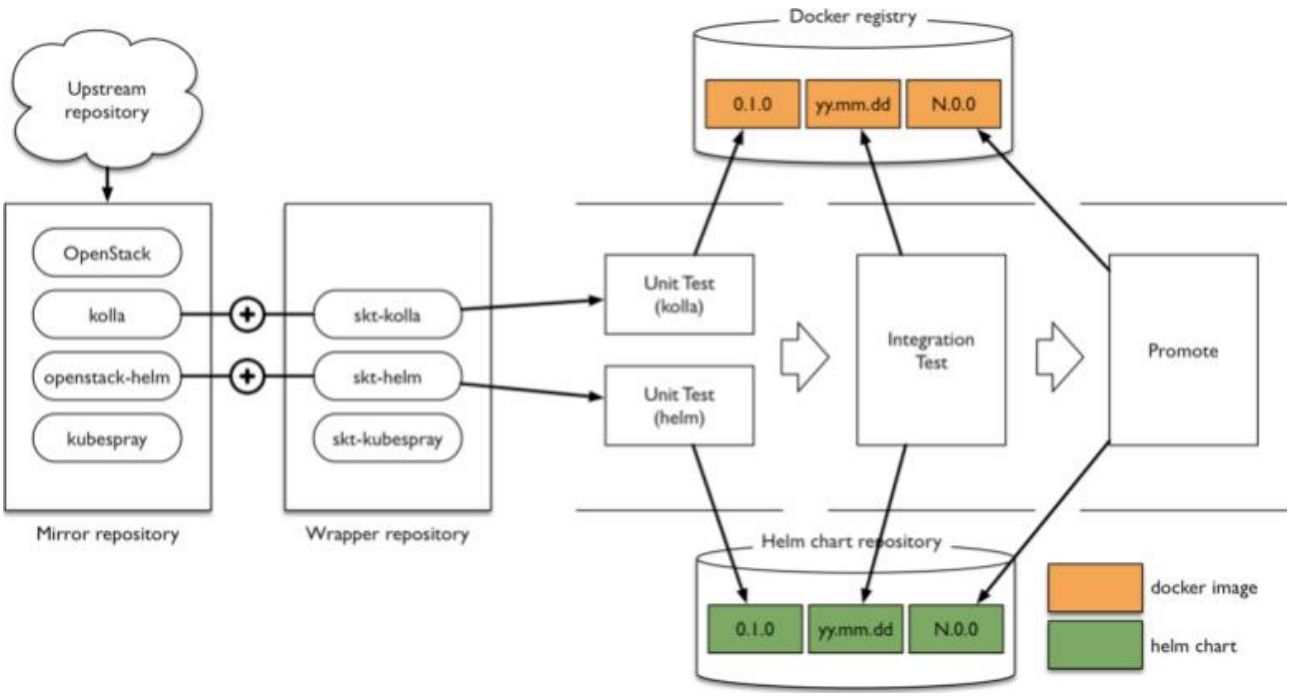
SK Telecom (SKT), South Korea's largest telecommunications operator, has been exploring optimized approaches for deploying OpenStack on Kubernetes with the aim of putting core business functions on containerized OpenStack by the end of 2018. SKT leverages Kolla and Openstack-Helm. with deployments automated by Kubespray. SKT devotes nearly 100% of it's development efforts to OpenStack-Helm, and works closely with AT&T to make OpenStack-Helm successful.

SKTは、Kubernetes上に展開するOpenStackに関する彼らの努力のため、ほかのツールとも協力しています。ロギング、モニタリング、そしてアラームのために、彼らはPrometheusや Elasticsearch、Fluent-bit、そして Kibanaを使っており、これらはすべてOpenStack-Helmコミュニティの中でデフォルトの参照ツールになっています。SKTはこれらをまとめ、TACO（SKT All Container OpenStack）と呼ばれる単一の内製統合されたソリューションに結合しています。



SKTは特に、コンテナ化されたKubernetes上のOpenStackについて、自動化された継続的インテグレーション/継続的デリバリ（CI/CD）パイプラインに精力的に取り組んでいます。SKTのCIシステムには、Jenkins、Rally、Tempest、Dockerレジストリ、そしてJiraやBigbucketも使われています。SKTはまた、 Cookiemonsterと呼ばれるオープンソースツールも開発しています。これは、chaos-monkeyのようなKuberunetes環境のための耐障害性テストツールであり、彼らのCIパイプラインのために耐障害性テストを行うものです。

すべての変更ごとに、SKTは自動的にOpenStackコンテナとHelm chartの双方をビルドし、テストします。日々、彼らは自動的に高可用性を備えたOpenStack環境をインストールます。これは、3台のコントロールノードと2台のコンピュートノードを備え、400のテストケースをTempestからその環境へ実行し、サービスの確認を行います。そして最後に耐障害性テストをCookiemonsterとRallyを使って実行します。完全なCIシステムは次の図に描かれています。



SKTは、このデプロイメントをAirshipの配下プロジェクトであるArmadaを使って自動化しています。これはAT&Tにより始められた新しいオープンソースインフラストラクチャプロジェクトであるコミュニティの中で導入されたものです。SKTは、彼らの商用利用に基づいたプロジェクトに向けて、増進を提供するためにコミュニティと協調しています。

実際のところ、SKTはKubernetes上にOpenStackをデプロイすることで、以下のような多くの利益をすでに得ています。

- 簡潔で簡便なインストール
- クラスタのオートヒーリング
- 稼働中のサービスへの影響を最小化するOpenStackのアップグレードやアップデート機能
- ブルーグリーンデプロイメントやカナリアリリースを含む、発展型リリース手法の高速な適用
- コンテナ隔離を通じたPython依存性の完全に自動化された管理
- 安全な認証情報と構成管理
- 高速でフレキシブルなクラスタアップグレードのロールアウト

SKTはこのアプローチを依然としてテストしてますが、彼らのOpenStack-Helm環境を商用で稼働するのに向け活発に進んでいます。この年末に、SKTは最低三つの商用クラスタを持つでしょう。さらに2019年には最大となる4番目のクラスタがオンラインになる予定です。このユースケースには以下のようなものが含まれます。

- ビッグデータプラットフォーム（2018年Q4に稼働する計画）
- 仮想デスクトップインフラストラクチャプラットフォーム（2018年Q4に商用適用可能）
- 汎用インターナルプライベートクラウド（2018年Q3に稼働する計画）

- 仮想ネットワークファンクション上に構築された通信事業者向けのネットワークインフラストラクチャ（2019年内に開設予定）

SKT is also trying to improve automation on telecom infrastructure operation by utilizing containerized VNFs and leveraging containers' auto healing and fast scale-out features. In order to allow interaction between virtual machine based VNFs and containerized VNFs, Simplified Overlay Network Architecture(SONA), which is a virtual network solution for OpenStack, will support communication between VMs and containers. SONA uses the Kuryr project for integration of OpenStack and Kubernetes, and it optimizes network performance using software defined networking technologies.

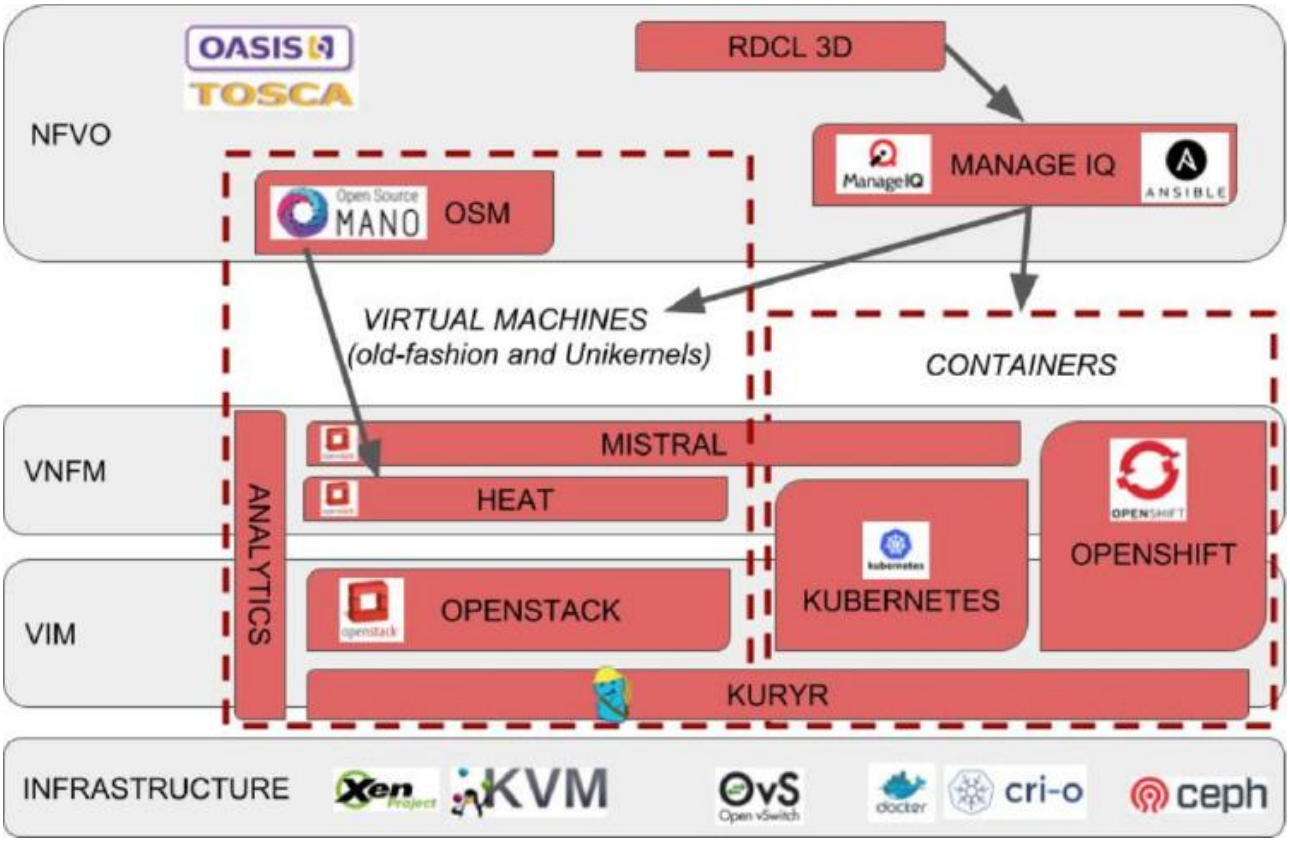
まとめとして、SKTはKubernetesがOpenStackのデプロイと運用のいくつもの複雑さを解決すると見出だしています。OpenStackを単純化することで5G時代に向けた高度なインフラストラクチャのイノベーションを提供する、パワフルなアプローチをSKTに提供します。Kubernetes上のOpenStack に集中的に努力することは、彼らの内部機能をコンテナ内のマイクロサービスに向けて進化させ、人工知能や物のインターネット、そして機械学習をデリバリする重要なインフラストラクチャに劇的に向上させます。

超流動

超流動プロジェクトは、ヨーロッパにある12の国に存在する、18のパートナーにより構成されています。このプロジェクトはサービスの立ち上げを素早く、ネットワーク上のどこ（コア、アグリゲーション、エッジj）でも起動させ、そしてそれらを透過的に違う場所に移動させる能力を発展させます。SUPERFLUIDITYはヨーロッパの研究プロジェクトであり（Horizon 2020）、5Gネットワークのために基本的なインフラストラクチャブロックを、よく知られたオープンソースの活用や拡張により、構築する挑戦を行っています。SUPERFLUIDITYは、コンバージドクラウドに基づいた5Gコンセプトを提供する予定です。これは、モバイルエッジの革新的なユースケースを実現するものであり、新規ビジネスモデルを力づけ、投資や運用コストを削減するものです。

これらのゴールを追求するために、プロジェクトコンソーシアムはレガシーなVMベースのアプリケーションから、Cloud Native様式のコンテナ化されたアプリケーションに移動させつつあります。KuryrはOpenStackの仮想マシンとKubernetesとOpenShiftのコンテナ化されたサービスの間を取り持つ、ブリッジとして提供されます。

このプロジェクトは、ManageIQを中央のネットワークファンクション仮想化オーケストレータ（NFVO）として利用し、Ansibleをアプリケーションデプロイメントとライフサイクル管理に使い、Heat、Neutron、Octaviaを含んだOpenStackサービスと、OpenShiftを通じたKubernetesをVMとコンテナの統合に使っています。



ManageIQアプライアンスから実行されるAnsibleプレイブックを活用することにより、SUPERFLUIDITYはアプリケーションをデプロイする共通の方法を提供します。これらのアプリケーションは、OpenStack HeatのテンプレートとOpenShiftのテンプレートにより提供されるクラウドオーケストレーション機能の利用し、次々と有効になります。

このコンソーシアムは5Gのクラウド無線アクセスネットワーク（CRAN）と、モバイルエッジコンピューティング（MEC）のコンポーネントを、コンテナの中にデプロイします。これはまた、分散インフラストラクチャ上に存在するビデオストリーミングのような、高スループットアプリケーションもデプロイします。

アプリケーションデリバリをCloud Nativeのアプローチにシフトしていくことは、高速で耐障害性をSUPERFLUIDITYのインストールにもたらします。これはVMベースのアプリケーションとコンポーネントをコンテナにスムーズに移行することを可能にし、一方で特定のアプリケーションにとってVM利用を可能にする多様性を維持します。これらのアプリケーションの例は、特別なセキュリティ保護や、シングルルートインプット/アウトプット仮想化（SRIOV）により必要とされるネットワーク高速化が上げられます。

スケール性能の試験において、SUPERFLUIDITYは約1000ポッドを秒速22ポッド（作成から稼働までの時間を計測）のレートで起動させることが出来ました。この驚くべき性能は、OpenStack上で管理されたVM上で稼働するOpenShiftによって達成されました。この環境では、Kuryrをポッドネットワークドライバとして使い、二重カプセル化による性能ヒットを避けました。

Select Another Case Study

IV. まとめ

過去数年間、コンテナ化は開発者と組織の双方に重要なツールとなってきています。OpenStackは、モジュラー設計と拡大するコミュニティを活用し、コンテナ技術をいくつものレイヤで統合してきました。これはコンテナとOpenStackを商用環境に持ち込もうとするいくつもの組織と、コンテナに新しい機能をもたらそうとするプロジェクトの数、双方に見ることが出来ます。OpenStack FoundationはOpenStack内で新技術を統合し、活用できるようにするために努力しており、コンテナは、その約束の重要な例です。

より学習するために、Containers Landing Page を訪れましょう。ここには、ドキュメントのコピーと共に、OpenStackとコンテナの統合にフォーカスした多くの映像へのリンクがあります。Kubernetes SIG-OpenStackはSlackチャンネル、メーリングリストがあり、もしあなたがKubernetesとOpenStackを統合するコミュニティと直接関わるならば、週次ミーティングもあります。

V. オープンソースプロジェクト一覧

Airship

Airshipは、相互運用可能で疎結合なオープンソースツールです。これは自動化されたクラウドプロビジョニングと管理を宣言的な方法で提供します。Airshipは、アプリケーションプラットフォームとしてKubernetesに基づいています。

Ansible

Ansibleは、しばしば利用されるオーケストレーションツールであり、OpenStack環境のデプロイや管理に使われてきました。

Cinder

OpenStack Cinderはブロックストレージをサービスとしてもたらし、70以上の異なった有力なストレージドライバにより、単一のAPIバックエンドを提供します。

Cloud Provider OpenStack

Cloud Provider OpenStackは、Kubernetes クラウドプロバイダーインターフェイスの実装です。これはOpenStackにホストされたKubernetesクラスタに、OpenStackに備えられたストレージとロードバランサ資源に直接アクセスすることを提供します。

Calico

CalicoはKubernetesとOpenStack双方に利用できるドライバによるネットワークオーバーレイであり、L3だけのルーティングを特徴とします。

Cyborg

Cyborgは、FPGAやGPU、ASIC、その他といったハードウェア高速化を管理する一般管理フレームワークです。POD向けの汎用ハードウェアインターフェイスを実現するため活動中です。

Docker

Dockerはオープンソースコンテナの仮想化フレームワークであり、ホストコンテナ化アプリケーションに使われてきました。

Helm

HelmはKuberunetesのための公式パッケージマネージャです。アプリケーションデプロイメントはHelm-Chartsによって記述され、Kuberunetesクラスタ上に自動的にデプロイし管理されます。

Ironic

IronicはOpenStackベアメタルサービスです。スタンドアロンサービスとしても、OpenStack Novaのドライバとしても稼働し、ベアメタルシステムの完全なライフサイクルを管理することが出来ます。子のライフサイクルには、エンロール、プロビジョニング、メンテナンス、そしてデコミッションが含まれます。

Loci

LOCIは、OpenStack環境のために、軽量でOCI準拠のコンテナを構築するOpenStackプロジェクトです。

LXC

LXCは、Linux kernelのnamespace隔離やほかの技術が隔離されたLinuxランタイムを作るための利点を持つ、低レベルのコンテナ仮想化インターフェイスです。

Kata Containers

Kataは、コンテナのように感じられ実行される一方で、VMのワークロード隔離やセキュリティの利点も提供する、軽量な仮想マシン（VM）の標準実装です。

Keystone

Keystoneは認証とユーザアカウントの管理、ロール情報の手段を主にOpenStackクラウド環境向け提供する、OpenStack Identityサービスです。一方で、Kuberunetesを含むほかの環境のプラグインも提供します。

Kolla (Containers)

Kolla (Containers)は、各OpenStackサービスのコンテナを構築するプロジェクトです。これにはよく考えられた構築と、テンプレートシステムが含まれ、ソースと数多くのホストOS上のパッケージの双方からコンテナを構築することが可能です。

Kolla Ansible

Kolla Ansibleは、Kollaコンテナを使い完全なOpenStack環境をデプロイし、保全するAnsibleを使ったOpenStackプロジェクトです。

Kubernetes

Kuberunetesは、クラウドインフラストラクチャ上に強固で高可用性なアプリケーションをデリバリする、コンテナオーケストレーションシステムです。

Kuryr

Kuryrは、DockerやKubernetesといったコンテナランタイム向けにNeutronネットワークオーバーレイを提供するOpenstackプロジェクトです。これは、コンテナとVMネットワークのための「統合ブリッジ（integration bridge）」を狙っています。

Magnum

Magnumは、container platforms as a serviceの管理機能を提供するOpenStackプロジェクトです。この中には、KuberunetesやDocker Swarm、MesosやDC/OSプラットフォームが含まれます。テナント分割されたアプリケーションを、シンプルなユーザ向けAPIを通じて作成することが出来ます。

Neutron

Neutronは、OpenStack software-defined networking サービスです。多くのネットワークドライバを背景に、動的なネットワークインフラストラクチャを提供する単一APIを提供します。

OpenStack Ansible

OpenStack Ansibleは、LXCコンテナにOpenStackサービスを構築するプロジェクトです。そして、これらのコンテナ化されたサービスと共に、OpenStack環境をデプロイし、管理します。

OpenStack Helm

OpenStack Helmは、OpenStackのライフサイクルをデプロイや管理し、Kuberunetes上のインフラストラクチャを管理します（例：CephやMariaDB）。さらに商用利用可能なデプロイを提供し、小さなエッジデプロイメントから、大きなセントラルフオフィスに至るまで、様々なユースケースに適用できます。また、Helmパッケージ管理システムも利用できます。OpenStack Helmはベアメタル（Ironic）と仮想（Nova/KVM）の両ワークロード管理をサポートし、LOCIとKollaコンテナのサポートによりイメージにとらわれません。

Qinling

Qinlingは、Functions as a Serviceを提供するOpenStackプロジェクトです。Qinlingはコンテナのための異なるオーケストレーションプラットフォームをサポートします。KubernetesやDocker Swarmだけでなく、ローカルのファイルストアやOpenStack Swift、そしてS3のような異なるストレージバックエンドの機能パッケージもサポートします。

Triple-O

TripleOは、インストール、更新、そしてOpenStackクラウドの運用を目的としたプロジェクトです。Nova、Ironic、Neutron、Heat、そして自動クラウド管理のためのAnsibleの上に構築される基礎として、OpenStackクラウドサービスを利用します。

Zun

ZunはOpenStackのコンテナサービスです。Zunは、サーバやクラスタを管理することなく、アプリケーションコンテナの実効に役立つAPIサービスを提供することを目的とします。

VI. 執筆者

OpenStack Kubernetes SIG コミュニティーのメンバー

- Jaesuk Ahn, SK Telecom
- Christian Berendt, Betacloud Solutions GmbH
- Anne Bertucio, OpenStack Foundation
- Pete Birley, AT&T
- Chris Hoge, OpenStack Foundation
- Lingxian Kong, Catalyst Cloud
- Hongbin Lu, Huawei
- Daniel Mellado, Red Hat, Inc.
- Allison Price, OpenStack Foundation
- David Rabel, B1 Systems GmbH
- Sangho Shin, SK Telecom
- Davanum Srinivas, Huawei
- Luis Tomás, Red Hat, Inc.
- Sam Yaple, Verizon Digital Media Services
- Mikhail Fedosin, Red Hat, Inc.
- Flavio Percoco, Red Hat, Inc.

エディター

- Brian E Whitaker, Zettabyte Content LLC

Translators

- Akihiro Motoki, <amotoki@gmail.com>
- Hajime Miyamoto, <ha-miyamoto@kddi.com>
- Shunichi Kuroyanagi, <sh-kuroyanagi@kddi.com>
- Hajime Miyamoto, <ha-miyamoto@kddi.com>