# ネットワーク自動化の技術解説およびコンテナネットワーク自動化導入事例の紹介

# Technical Description of Network Automation and Case Studies of Container Network Automation

鈴　木　駿　介

**要　約**　SDN（Software-Defined Network）は登場以来，ネットワーク自動化を実現する要素として注目されてきた．近年では，オーケストレーションツールやコンテナといった自動でスケーリングや設定変更を行うソリューションの利用が増え，自動化というキーワードが製品の選定における重要な要素となってきている．

ユニアデックスは 2018 年，A 社のコンテナ開発基盤において，BCF＋Container 連携の導入を行った．1 年後のヒアリングにて，BCF は問題なく安定して稼働しており，ほとんど触れなくても運用できているとの評価を得た．

**Abstract**　SDN (Software-Defined Network) has been drawing attention as an element that realizes network automation since its appearance. In recent years, the use of solutions that automatically scale and change settings, such as orchestration tools and containers, has increased, and the keyword of network automation has been becoming an important factor in product selection.

In 2018, Uniadex introduced BCF + Container cooperation in the container development platform of Company A. One year later, we got an evaluation that the BCF was working stably without any problems and could be mostly operated without user intervention.

## 1. は　じ　め　に

企業が新しいサービスを顧客に迅速に提供できるかが，ビジネスにおいて重要視される時代となってきた．近年では，急速に変化するニーズに素早く順応するため，ビジネスでパブリッククラウドの利用が増え，オンプレミスとパブリッククラウドの両方にデータを格納するハイブリッドクラウドの利用にも関心が高まっている．これに伴い，オンプレミスもクラウドでの運用を前提とした IT インフラ作りが必須となり，ネットワーク分野にも，迅速な構築と柔軟な運用が求められるようになった．

2010 年代の SDN（Software-Defined Network）の登場以来，ネットワークの自動化というキーワードがソリューションの選定における重要な要素となってきており，今年のコロナ禍におけるビジネスの DX 移行を通じさらにこの動きは加速するであろう．

本稿では，ネットワーク自動化の技術を解説し，導入事例を紹介する．まず 2 章でネットワーク自動化を構成する技術の紹介，3 章でユニアデックス株式会社（以降，ユニアデックス）が検証を行ったネットワーク自動化ソリューションの紹介と，実際の導入事例をもとにした留意点や課題の解説を行う．

## 2. ネットワーク自動化を実現する技術要素の解説

本章ではネットワーク自動化において重要となる技術要素を解説する．ネットワーク自動化とは，ネットワークの「構築」「運用・保守」に関する作業を自動で行うようにすることである．ネットワークはITインフラの中でも，管理する機器の多さや障害時にシステムに与える影響度の大きさなどから，現状の変更に非常に慎重で，設定変更を行う際にも，システム全体に与える影響の検証などに時間を多く要するものであった．2000年代にサーバー仮想化技術によって，サーバー側で構築を迅速に行えるようになったが，ネットワークは，この特有の事情から手動で設定変更を行っている場合も多かった．

近年，DevOpsをはじめとして，システムの開発からサービスインまでのサイクルのスピードが重視されるようになると，オーケストレーションツールやコンテナといった，自動でスケーリングや設定変更を行うソリューションの利用が増えてきた．ネットワーク側の設定変更が手動のままでは，作業のボトルネックとなってしまうため，ネットワークも構築の高速化，自動化が求められるようになっていった．

### 2.1 OpenFlowによるネットワーク制御とSDNの誕生

ネットワーク自動化というキーワードが大きく注目されるようになったのは，SDNの登場以降である．SDNとはSoftware-Defined Networkの略で，ネットワークの煩雑な構築，設定変更をソフトウェアベースで自在にできるようにした技術である．SDN自体に明確な定義というものは存在しないが，2010年代初頭では，主にOpenFlowを用いてネットワークを制御する製品がSDNという認識であった．

OpenFlowはネットワーク統合管理を実現する通信プロトコルであり，実際にSDNを支える重要な要素の一つである．2010年代前半から標準化団体であるONF（Open Networking Foundation）にて標準化が進められ，2012年4月にリリースされたOpenFlow1.3ベースのソフトウェアが広く利用されている．OpenFlowはコントローラーとスイッチ，その間の通信プロトコルの三つの要素から構成され，コントローラーがスイッチに対して設定するフローテーブルに従ってデータ転送処理が行われる仕組みになっている．

従来はネットワークを制御するコントロールプレーンと，データ転送のテーブルであるデータプレーンが，ネットワーク機器個々に実装されていたため，ネットワーク機器の単位でしかネットワークを制御できなかった．これがOpenFlowにより，ネットワーク全体を管理するコントローラーとOpenFlowに対応したスイッチ群の間を一つのコントロールプレーン，OpenFlowに対応するスイッチ群を一つのデータプレーンのように見せることができ，ネットワーク全体をあたかも一つのネットワーク機器のように，管理，制御することができるようになった．従来のネットワーク機器で1台ずつ設定していたOSインストール，物理ポートの設定，冗長設定をほぼ自動で行える（Zero Touch Provisioning）ようになり，2.4節で述べるREST APIの技術と併せ，ネットワーク自動化実現へ大きな期待を抱かせるものとなった．

OpenFlowプロトコルはコントローラーの機能やネットワークの仕様等の定義はなく，SDNのコントローラーは各社各様の機能実装状況になっている．また，OpenFlowプロトコルを用いていない統合管理型のネットワーク製品もSDN製品として多く認識されており，現在，OpenFlowの実装はSDN製品であるための必要条件ではないと言える．

### 2. 2　VXLAN によるオーバーレイネットワークの実現

仮想化によるサーバー集約統合やクラウド化により，特にデータセンターにおいて複数のシステムを大規模かつ効率的に収容する IT インフラへの要望が出てきた．ネットワークの運用管理性や柔軟性を上げるために，次節で述べる，IP Clos Fabric と組み合わせてネットワークを一つの土管のようにシンプルに見せる技術が登場してきた．

ネットワークでブロードキャストドメイン（L2 ネットワーク）を分離する技術に VLAN がある．VLAN は識別番号となる VLAN ID をイーサネットヘッダーに含めることができ，12 ビットで表記する．VLAN の規格では VLAN ID は 0 と 4095 が予約されており，合計で最大 4094 個の ID しか使用できない．例えば，大量のサーバーを収容する大規模ネットワークにおいて，物理的な設置場所が異なるような場合には，L3 ネットワーク機器を挟み，L2 セグメントを分割することがある．このため大規模ネットワークでは，VLAN による拡張性に限界があり，L2 ネットワークの分割でネットワーク全体も複雑なものになってしまう．

VXLAN は，2014 年の 8 月に RFC7348 にて標準化された，従来の VLAN が抱える課題を解決するために作られた技術で，仮想の L2 ネットワークを実在の L3 ネットワーク上に拡張させるトンネリングプロトコルである．VXLAN では，UDP パケットの中にイーサネットフレームをカプセル化する．これによって，本来ならルーターなどの L3 ネットワーク機器を挟んだネットワークも論理的に同じ L2 セグメントのネットワークに見せることができる．この場合，物理ネットワーク（アンダーレイネットワーク）に対し，論理的な L2 ネットワークをオーバーレイネットワークと呼ぶ．

VXLAN パケットを終端する箇所を Virtual Tunnel End Point（VTEP）といい，VTEP 機能は VTEP 対応スイッチや Linux カーネルなどのソフトウェアでも実現できる．VMware の NSX では，分散仮想スイッチ（vDS）にこのソフトウェア VTEP 機能を持たせ，NSX Controller で vDS の管理，制御を行い OpenFlow 対応の物理スイッチを使用せずにネットワークの統合管理を実現している．このようなコンポーネントを持つ製品を，OpenFlow 対応スイッチを使用するホップバイホップ型 SDN に対して，オーバーレイ型 SDN という．2010 年代後半になると，SDN 製品はホップバイホップ型，オーバーレイ型とも多様化し，両方の形式の特徴を持つ製品（ハイブリッド型）も多く登場した．

### 2. 3　IP Clos Fabric と大規模ネットワークの設計

IP Clos Fabric はデータセンターの大規模なネットワークの設計を目的とした設計モデルで，2016 年 8 月の RFC7938 において定義されている．従来のネットワークは，例えば社内の端末がインターネットへアクセスしやすくするための，North-South 通信[*1] 向けの物理設計が一般的であった．しかし，IT 化が進むにつれて，データセンターネットワークでは，サーバー同士の East-West 通信[*2] において，トラフィックが大量に発生するため，それに対応できる構成が求められるようになった．

IP Clos Fabric は図 1 のようにスパインスイッチ・リーフスイッチの 2 階層で構成し，フルメッシュのファブリック構成で接続する．サーバー，ルーター等の機器は，全てリーフスイッチに接続し，スパインスイッチにはリーフスイッチだけを接続することで，リーフスイッチ間の渡りとして使用する．スイッチ間のネットワーク構成としては，L3 のネットワーク構成をとり，プロトコルにはダイナミックルーティングの eBGP を使用し，等コストマルチパス

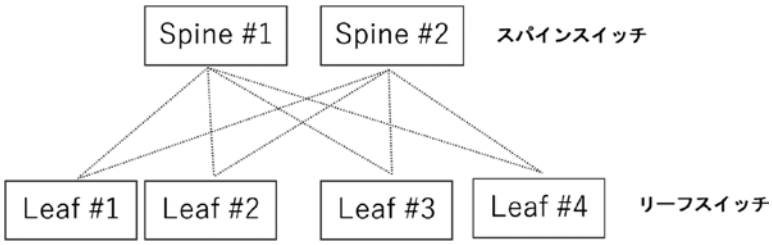


**図 1　IP Clos Fabric の構成**

（ECMP）により，スイッチ間で全ての経路を使用できるようにする．

これにより，どのリーフスイッチからでも必ず 1 ホップで別のリーフスイッチに到達でき，East-West 通信に強い構成であると同時に，従来の North-South 通信向けのネットワーク構成と異なり，ループ構成によるブロックポートは存在せず大量のトラフィックに対応できるネットワークとなっている．このように，縦横無尽に張り巡らされた経路を柔軟に使用でき，ファブリック全体を一つの大きな土管のようにシンプルに使用できるので，データセンターネットワークやオーバーレイネットワークを使用する SDN 製品のアンダーレイネットワークの構成として推奨されており，次世代のネットワーク製品と非常に相性が良い．現在，IP Clos Fabric はデータセンターネットワークだけではなく，安価なスイッチを使用して，キャンパスネットワークへの導入も進んでいる．

また上記のような RFC7938 で定義されている構成の他に，SDN 製品によっては，スイッチ間の接続に OpenFlow によるフローテーブルに基づいたデータ転送を利用することによって，より簡易的に IP Clos Fabric を実装しているものもある．

### 2. 4　REST API によるネットワーク制御と製品連携

SDN は前節までに述べた各要素を組み合わせ，ネットワークの統合管理，簡素化，自動化を実現している．また，ネットワークの自動化を実現する重要な要素として本節の REST API による制御も関わってくる．REST API とは，文字通り REST（Representational State Transfer）の思想によって設計された API のことで，Web API の一つである．REST の設計思想とは，大きく分けて以下の 3 点である．図 2 に REST API の概略を示す．

1）　リソース（情報資源）を一意な URI で表現する
2）　HTTP 技術をベースとし，HTTP メソッドで操作方法を表現する
3）　処理結果を，XML，JSON，YAML 等のプログラムに使用される汎用的なデータ形式で返す

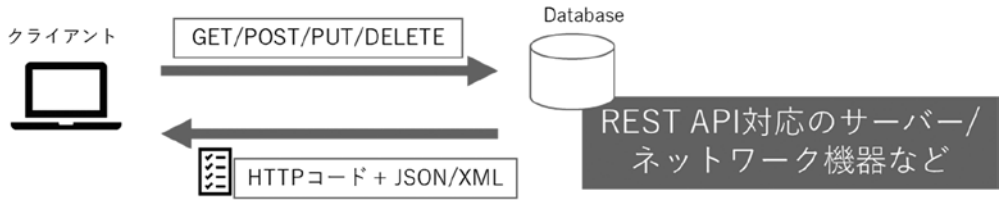


**図 2　REST API の概略図**

上記の設計思想は，人が理解しやすくプログラム化しやすいため，多くの製品に実装されている．SDN 以外のネットワーク機器でも REST API に対応しているものがあるが，特に SDN と REST API は相性が良く，ネットワークの制御を行う窓口がコントローラー一つであるため，比較的容易にプログラムに実装できる．これは SDN がネットワーク自動化を容易にする理由の一つである．例えば，curl 等のコマンドでコントローラーへ設定を行うスクリプトを記述できれば，その知識だけで，ネットワークを自動設定するプログラムが実装できる[*3]．

また，多くの仮想化製品やオーケストレーションツールが REST API に対応しているため，それらの製品同士で，製品間の連携ができる．例えば，サーバー仮想化プラットフォーム上で，新しく仮想マシンを作成したら，その設定変更は API を経由してネットワーク機器に伝達され，自動で設定を更新できる．

この SDN 製品との連携機能によって，データセンターの IT インフラをソフトウェア制御するソリューションである SDDC（Software Defined Data Center）や，IT インフラ全てをソフトウェア制御する SDx（Software Defined X）や SDI（Software Defined Infrastructure）という概念が生まれ，一つの操作によって，ネットワークを含む IT インフラが自動で設定される IT インフラ自動化が注目されるきっかけとなった．

この頃から，SDN は製品単体ではなく，SDI を実現するソリューションの構成要素の一つとして，他の製品と組み合わされるようになった．

### 2. 5　SDN 技術のパブリッククラウド連携とハイブリット環境でのネットワーク管理

2018 年頃になると，ビジネスにおいてパブリッククラウドの利用が前提となってきた．しかし，企業のコンプライアンス上，全ての業務データをパブリッククラウドへ移行するのは難しい場合も多く，パブリッククラウドとプライベートクラウドをシームレスに接続して利用する，ハイブリッドクラウドという運用方法が注目を集めてきた．

ハイブリッドクラウドでは，パブリッククラウドとプライベートクラウドを同時に利用しているため，企業のシステム管理者はこの両方の基盤の運用方法や，それらを統合する仕組みの知識を問われる．システム管理者の負担を減らすため，パブリッククラウドとオンプレミスのプライベートクラウドを同じ製品のポリシーで管理することが望ましい．

各ネットワークベンダーも，主要パブリッククラウドの AWS（Amazon Web Service），Microsoft Azure，GCP（Google Cloud Platform）を中心に，パブリッククラウドとの連携ソリューションをリリース，もしくはロードマップに加えている．パブリッククラウド連携の主な機能としては，SDN ベンダーが持っているオーケストレーションツールや SDN コントローラーと API を通して，パブリッククラウドの操作，AWS の Auto Scaling のような自動スケーリングサービスの動作に応じたネットワーク設定の自動変更や，同じ製品の UI でのオンプレミスとパブリッククラウドのネットワーク管理などである．

## 3.　ユニアデックスのネットワーク自動化検証と顧客導入事例

ユニアデックスでは，2010 年代前半から SDN 製品の検証を行い知見を蓄積してきた．Big Switch Networks 社（2020 年 2 月から，Arista Networks 社）が提供する，Big Cloud Fabric（BCF）は，最も早い時期から検証と導入を行ってきた製品のひとつである．特に，前章の REST API によるネットワーク制御と製品連携については，オンプレミス環境のネットワー

ク自動化製品のサービス化を目指して，BCF，Cisco ACI などの SDN 製品と VMware vSphere や OpenStack，Kubernetes との連携検証を多数行ってきた．その中で実際に，SDN 製品と REST API を使用した製品連携の自動化ソリューションは数件の導入実績がある．近年では，VMware 社の製品を中心にハイブリッドクラウド環境の運用を目指した，IT インフラのオーケストレーションツールやパブリッククラウド連携についても検証を始めている．

本章では，ユニアデックスのネットワーク自動化の取り組みの一例として，BCF のコンテナネットワーク自動化ソリューションの技術解説と実際の顧客導入事例について紹介する．

### 3. 1　BCF の概要

本節では BCF について紹介する．BCF はコントローラーとスイッチという基本構成をとり，コントローラーに実装したソフトウェアによってスイッチを管理・制御する製品である．BCF の物理構成を図 3 に示す．

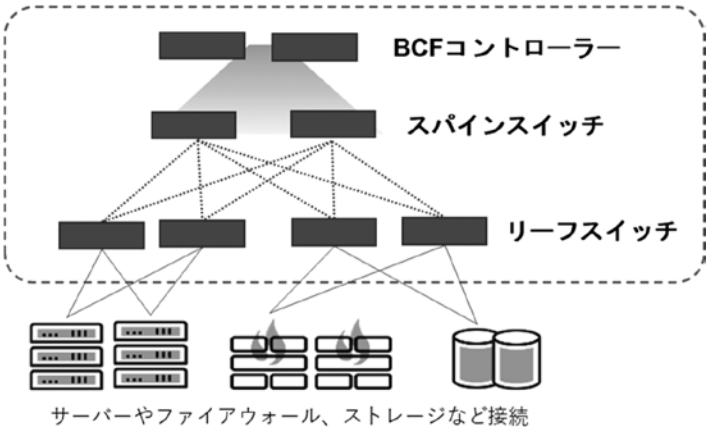


**図 3　BCF の物理構成**

ネットワーク全体を管理・制御するコントローラーとスイッチ部分はスパインスイッチ・リーフスイッチの 2 層の IP Clos Fabric の構成をとる．BCF はコントローラーへスイッチを登録すると，コントローラーから OS を自動的にダウンロードし，ネットワークの設定を取得する．ユーザーは，コントローラーからネットワーク設定を行い，各スイッチに設定を入れることなく，1 台のスイッチとして扱うことができる．BCF は主にデータセンターネットワークの自動化，運用簡素化に重きを置いている製品であり，多数の仮想化，コンテナプラットフォームと連携できる．

### 3. 2　BCF＋Container 連携の解説と導入事例の紹介

本節では，実際に導入実績のあるコンテナネットワークの自動化を実現する BCF＋Container 連携を一例として紹介する．

#### 3. 2. 1　コンテナの解説

はじめに，コンテナ技術の概要と Docker，Kubernetes について述べる．仮想化技術の中で，

現在主流のハイパーバイザー型仮想化では，仮想マシン毎にメモリ，ディスクを確保しなければならず，リソースが逼迫したり，アプリケーションの実行環境の構築まで待たされたりすることがあった．その課題を解決するために，コンテナ技術が考案された．コンテナ技術は元々Linux カーネルがプロセスのリソースを隔離する機能である cgroups（control groups）の利用により提供されていたが，2013年にDockerが発表され，コンテナプラットフォームのデファクトスタンダートになったことから，大きく注目されるようになった．コンテナは，アプリケーションを実行するためのライブラリやコンフィグなどが含まれ，それらをファイルなどのテンプレートとして起動，管理，共有するソリューションであり，IT インフラにコード化（Infrastructure as Code：IaC），可搬性という新たな概念をもたらした．

本番環境においてDockerを運用する場合，Dockerでは複数のホストにコンテナを横断的に配置・管理する機能はなく，複数のホストでクラスタを組む運用に適さない．そのため，本番環境においてはDocker単体ではなく，コンテナアプリケーション管理ソフトウェアのKubernetesと合わせた運用が一般的である．Kubernetesでは複数のホストでクラスタを構築する場合，クラスタ全体の管理を行うMasterノード（以降，Master），コンテナのデプロイ先になるWorkerノード（以降，Worker）といった役割がある．両方とも複数のホストで構成できる．図4にKubernetesの主な構成要素を図示し，説明を表1にあげる．

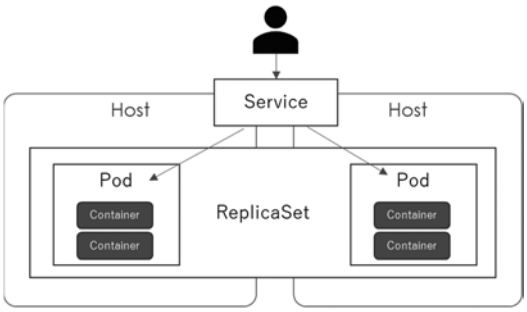


**図4　Kubernetesの構成要素**

**表1　Kubernetesの構成要素の説明**

| 構成要素 | 説　　明 |
|---|---|
| Pod | Kubernetesにおけるコンテナの管理体．一つ以上のコンテナをPodという形でデプロイする．Pod内に含まれるすべてのコンテナが同じホストにデプロイされる． |
| ReplicaSet | 複数のPodをセットのように管理できる機能．Pod数を設定でき，デプロイされているPod数と設定したPod数が足りない場合は新規Podを追加し，多い場合は既存のPodを停止する機能を持つ． |
| Service | Pod内のサービスにアクセスするためのロードバランサー． |

### 3.2.2 Container Network Interface（CNI）の解説

Kubernetesを構築する際，考慮すべき重要な要素に，Container Network Interface（以下，CNI）がある．CNIは，クラスタ環境で複数のホスト間のコンテナネットワークを管理するためのインターフェース標準仕様であり，Kubernetes構築の際は必須のものとなる．現在は

CNCF（Cloud Native Computing Foundation）というプロジェクトに移管されており，多くのネットワークベンダーがCNCFに参画している．CNIの中には，ネットワークベンダーが独自に提供するものもあり，有償のメーカーサポート対象のものもある．

CNIの中で主に利用されるものは，CoreOS社が提供するホスト間にオーバーレイネットワークを構築するFlannelや，Metaswitch Networks社が提供するホスト間の接続にBGPを用いるCalicoなどがある．オンプレミスにおいてKubernetesを構築する際は，各CNIの特徴を熟知した上で，要件に応じて使い分ける．例えば，Podへのアクセス制御を行う要件がある場合，実装方法としては，KubernetesのNetwork Policyという機能を使用する，もしくは物理ネットワーク機器側でアクセス制御を設定する方法が考えられる．前者では，Network PolicyはCNIによってサポートしていないものがあり，有名なCNIでもFlannelはNetwork Policyをサポートしていない．後者においては，物理ネットワーク機器の設定変更を要し，頻繁に構成変更が生じるコンテナ基盤とは相性が悪いというデメリットがある．

このように，Kubernetesを運用するエンジニアは，OSSのCNIを実装する場合，実現したいネットワーク要件に合う仕様のCNIを選定するために，各CNIの仕様およびネットワークに関する知識を習得しなければならない．また，実際の運用においてネットワークの監視，障害時のトラブルシューティングを行うとともに，CNIをはじめとしたソフトウェアライフサイクルが短いKubernetesのコンポーネントに対し，アップデート情報を常に収集しつつアップデートを実施しなければならない．

### 3. 2. 3　BCF＋Container 連携の解説

前項のCNCFに参画しているBig Switch Networks社は，独自のCNIプラグインを提供している．自動デプロイによるコンテナネットワークの簡単な構築，コントローラーによるコンテナネットワークの一元管理，Podへのアクセス制御の容易な実装を提供する．

本項では，BCF＋Container連携の動作の解説を行う．以下にBCF＋Container連携で提供される主な機能を挙げる．

1）　ホスト（Master，Worker）の自動検知・自動LAG構築
　　ホストがBCFスイッチに接続したタイミングで，自動でLAGの設定追加．
2）　PodのIPアドレス管理/ネットワークの自動構築
　　Kubernetes側の操作に応じて，BCFコントローラーの自動設定変更．
　　コンテナデプロイ時にIPアドレスの自動払い出し．
3）　コンテナのイベント可視化機能
　　BCFの標準機能のAnalyticsでコンテナのイベントを可視化．

BCF＋Container連携はコンセプトとして，ネットワーク機器に触らなくても，Kubernetes運用者でインフラの管理ができること，ネットワークを知らない人でもCNIやネットワークを意識せずにKubernetes環境を構築できることを目指している．Kubernetes側の操作に応じたネットワークの自動設定変更や，BCFにおいてコンテナを含めたネットワークの一元管理を提供している．BCF＋Container連携の主な構成要素と役割を図5に示し，表2で説明する．

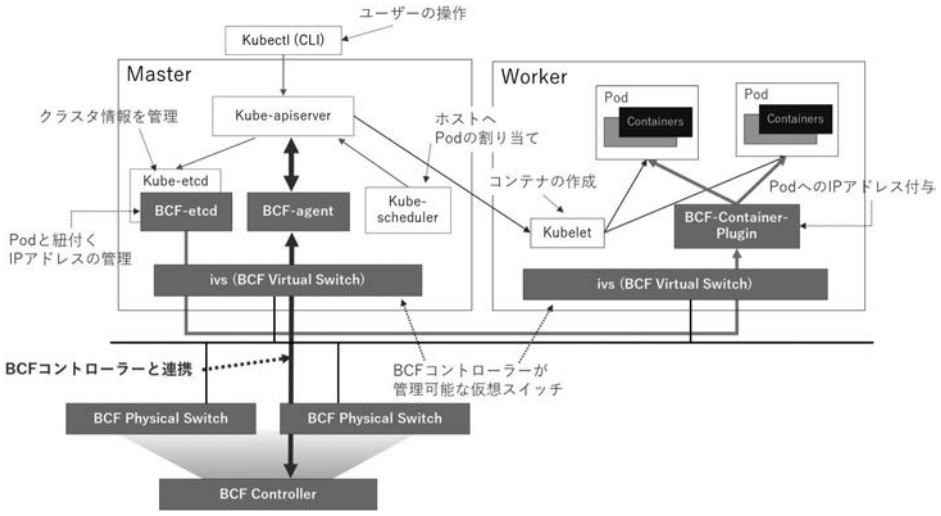


**図5　BCF＋Container 連携の構成要素**

**表2　BCF＋Container 連携の構成要素の説明**

| 構成要素 | 説　　明 |
|---|---|
| kubectl | Kubernetes の CLI クライアント. |
| kube-apiserver | Kubernetes の API サーバー．kubectl からの REST API リクエストを実際に処理し，kube-etcd 内の対応するオブジェクトを更新する. |
| kube-scheduler | 作成された Pod を各 Worker へ割り当てるスケジューラー. |
| kubelet | Worker で Pod の起動・管理を行う. |
| BCF-Container-Plugin | Pod に対して IP アドレスを提供する. |
| ivs | Linux カーネルに組み込まれている OpenvSwitch8 ベースの仮想スイッチで，BCF の OS と BCF が提供する CNI のプラグインを組み込み，BCF コントローラーが管理可能なスイッチとして動作する. |
| BCF-agent | kube-apiserver からの REST API のリクエストを BCF コントローラーの API が処理できる形に変換し，BCF コントローラーへ伝える. |
| BCF-etcd | クラスタ情報に加えて，BCF 特有の設定情報を管理する. |

BCF-agent は BCF コントローラーと API 連携し，kube-apiserver からの REST API のリクエストを BCF コントローラーの API が処理できる形に変換して BCF コントローラーへ伝え，kubernetes 側の操作に応じてネットワークの設定を変更できるようにしている．また，CNI プラグインを組み込んだソフトウェアスイッチ（ivs）を，BCF のコントローラーで管理できる．これにより BCF コントローラーで CNI を含むコンテナネットワークの管理を実現している.

### 3. 2. 4　BCF＋Container 連携の顧客導入事例

ユニアデックスでは，A 社のコンテナ開発基盤において，BCF＋Container 連携の導入を行っ

た．BCF 導入の経緯として，A 社は元々コンテナベースの開発基盤を運用していた中で，コンテナネットワークの管理をより簡素にしたいという目的があった．そのため，CNI を自動で導入し，コントローラーでコンテナネットワークの管理を一元的に行うこと，また Pod へのアクセス制御，Pod ネットワークのセグメント分割という要件を満たし，CNI に対するメーカーサポートが付属する BCF＋Container 連携を選定するに至った．

本項では，導入前の実機検証において判明した，BCF＋Container 連携環境の運用面における留意点と，実際に導入する際の対応について記載する．留意点は大きく分けて二つある．

一つ目の留意点は，BCF のアップデート時の通信断時間である．BCF のような SDN 製品は非常に早いサイクルで機能追加が行われており，ソフトウェアのライフサイクルが従来のネットワーク製品に比べて短く，定期的にアップデートを要する．BCF の OS のアップデートは，コントローラーでアップデートコマンドを実施した場合，管理しているスイッチも自動的に OS のアップデートを実施する，ゼロタッチアップデートを特徴としている．しかし，今回のような Kubernetes のサーバー側の ivs に組み込まれている BCF OS に関しては，ゼロタッチでアップデートを実行できず，ivs のアップデート[*4] を手動で行うため，ivs を組み込んでいる環境で BCF の OS アップデートを行う場合はサービス断が生じる．

これの回避方法としては，別の BCF コントローラーを用意して，Container 連携環境を構築し，公開している Pod を退避させる，また，アップデート用にあらかじめメンテナンスタイムを確保し，長い通信断時間の承諾を得るなどが挙げられる[*5]．

もう一つの留意点は，一部機能の制限と冗長性の問題である．BCF＋Container 連携において，Community 版 Kubernetes を用いた連携を行う場合，サポートしているのが Kubeadm[*6] を用いた構築のみである．Kubeadm は簡単に構築できる反面，フルスクラッチで Kubernetes を構築する場合に比べ，Service の一部の機能が使用不可で，細かな制御もできない．さらに，Master1 台に対して Worker はその Master に対してのトークンしか持てないため，Master の冗長構成をとることができない．

回避策としては，BCF＋Container 連携で Master の冗長をサポートしているコンテナプラットフォームを連携に使用することである．BCF においては RedHat が提供する Kubernetes ベースのコンテナプラットフォームの OpenShift との連携を使用することで回避できる[*7]．

A 社には 2018 年に BCF＋Container 連携を導入し，その 1 年後の設定変更の際に，稼働状況についてのヒアリングを行った．その結果，システム全体では特に問題なく安定して稼働しており，ほとんど BCF に触れなくても運用できているとの評価を得た．

## 4. お わ り に

ユニアデックスの IT インフラ自動化の取り組みとして，ネットワーク自動化についての要素解説と，導入事例として BCF＋Container 連携を紹介した．最近では，BCF の製品自体のアップデートで機能が改善した他，Cisco ACI，VMware NSX などにもコンテナ連携に関する機能のラインナップが充実してきており，他製品による代替もできるようになってきた．今後も，IT インフラ自動化ソリューションに関しては，既存製品のアップデートや他製品のリリースによって，改善や代替がされていくと期待される．

アフターコロナにおける社会では，デジタルトランスフォーメーション（DX）も加速すると見込まれる．自ずとハイブリッドクラウド，コンテナの利用も増え，ネットワーク自動化の

必要性もますます高まっていくと考えられる．それにより，IT インフラエンジニアにも，自動化に関連したプログラミング知識など，求められるスキルも変化していくだろう．

本稿が，今後も進化する SDN 管理における複雑なインテグレーションやアーキテクチャ設計に貢献することを期待する．

---

＊1　North-South 通信とは，ネットワークにおいて基幹部から末端の端末へ流れる通信のことを指す．ネットワークの物理構成図において，上下方向のやり取りが為されるので地図の南北に見立てて North-South 通信と呼ばれる．
＊2　East-West 通信とは，ネットワークにおいて末端同士の通信のことを指す．ネットワークの物理構成図において，左右方向のやり取りが為されるので地図の東西に見立てて East-West 通信と呼ばれる．
＊3　実際は，より複雑な処理を行うため，Python 等の言語を使用し独自にプログラミングを行う．製品によっては，サンプルコードやネットワーク機器に受け渡す情報とデータ形式を公開しているものもあり，それを環境に合わせカスタマイズすることで実装する．
＊4　アップデートのほとんどは，AnsiblePlaybook において半自動で行うことができる．
＊5　A 社への導入時は，アップデートは新たにシステムを構築し直す方針で導入となったため，アップデートの課題に関しては大きな障壁とはならなかった．
＊6　Kubeadm とは Kubernetes Community が提供している，Kubernetes クラスタ簡易構築ツールである．
＊7　A 社では，開発基盤のため元々 Community 版 Kubernetes を使用する方針で検討していたこと，また，Master が 1 台障害にあったとしても，現状デプロイされている Pod には影響を及ぼさないため，当初の方針通り Kubeadm を使用して構築を行うこととなり，冗長性のリスクに関しては受容する形となった．

**参考文献**　［1］ Converged Cloud Fabric，Arista Networks, Inc, 2020, https://www.arista.com/en/products/converged-cloud-fabric
［2］ 青山尚輝，市川 豊，境川章一郎，佐藤聖規，須江信洋，前佛雅人，橋本直哉，平岡大祐，福田 潔，矢野哲朗，山田修司，コンテナ・ベース・オーケストレーション Docker / Kubernetes で作るクラウド時代のシステム基盤，翔泳社，2018 年 3 月，P270 ～ 278
［3］ 田村郷司，次世代の IT インフラ - ゼロトラスト・アーキテクチャ ネットワーク視点で捉えるゼロトラストの必要性，PwC コンサルティング合同会社，2020, https://www.pwc.com/jp/ja/knowledge/column/awareness-cyber-security/zero-trust-architecture03.html
［4］ P. Lapukhov, A. Premji, J. Mitchell, Ed., Use of BGP for Routing in Large-Scale Data Centers, Internet Engineering Task Force (IETF), 2016 年 8 月 https://tools.ietf.org/html/rfc7938

※上記参考文献に含まれる URL のリンク先は，2020 年 10 月 14 日時点での存在を確認．

**執筆者紹介**　鈴　木　駿　介（Shunsuke Suzuki）
2016 年ユニアデックス（株）入社．SDN ビジネスに関わり，主にデータセンターネットワークの技術サポート業務に携わる．現在まで，金融系クラウド基盤の導入，製造系コンテナ基盤の設計・構築を担当．