令和 5 年度高エネルギー加速器研究機構技術研究会、2024 年 3 月 7～8 日
https://www2.kek.jp/engineer/tsukuba/giken/2023/

Kubernetes(クーバネテス)基盤によるログ収集・分析環境の構築と監視運用

○中村貞次#
高エネルギー加速器研究機構 共通基盤研究施設 計算科学センター

概要

Kubernetes（クーバネテス）[1]とは、ギリシャ語で『舵取り』や『操縦士』を意味する。先頭の「K」と語尾の「s」間が 8 文字を数え「K8s」とも略され、コンテナ化したアプリケーションのデプロイ（実行可能な状態）、スケーリング（負荷に応じた適切なパフォーマンスと可用性を確保）および管理を行うオープンソースである。本報告では、ハイパーバイザー型仮想環境の違いと Kubernetes クラスタ基盤を構築し、ファイアウォールで検知される threat（脅威）や URL filter のログを収集し、分析環境の構築とそのログ監視運用を整備したので報告をする。

1. 経緯・導入

サービスの多様化と効率化が進みシステムを仮想化することは一般的になっている。ハイパーバイザー型仮想化は、物理的なハードウェア上に仮想化ソフトウェアを配置し、複数の仮想マシン（以下、VM という）をホストする仮想化環境である。各 VM は独立した仮想ハードウェア上で実行され、異なるオペレーティングシステム（以下、OS という）を実行している。一方、コンテナ型仮想化はコンテナと呼ばれる軽量な実行環境を使用してアプリケーションを分離する仮想化手法で、コンテナは、ホスト OS のカーネルを共有しアプリケーションの依存関係やランタイムを含んだ独立した実行環境である。コンテナ型仮想化は、カーネルを共有するためハイパーバイザー型よりオーバーヘッドが減り、高速起動や軽量で効率的なデプロイメントが行える利点がある。図 1 にハイパーバイザー型とコンテナ型の違いを示す。

Kubernetes（クーバネテス）は、複数のコンテナ化されたアプリケーションやマイクロサービスを効率的に管理・デプロイ・スケーリングするためのコンテナオーケストレーションプラットフォームである。また、Kubernetes クラスタは、複数の物理的または仮想的なマシンを冗長化し、スケーラビリティ、可用性、応答性を向上させる。元々Google 社が開発し、2015 年から CNCF（Cloud Native Computing Foundation）のコミュニティが継承し、オープンソースのアップストリームを開発・配布している。

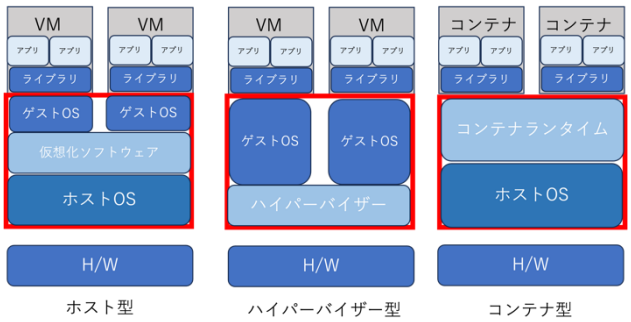

図 1. 従来型とコンテナ型の違い

2. Kubernetes の機能と利点

Kubernetes は、管理を行うコントロープレーンとアプリケーションの実行を行うワーカーノードを複数台用意してハードウェア障害、アップデート、負荷分散などに対応できるクラスタ構成を組み、無停止で運用を継続できる利点があげられる。Kubernetes には、機能要件と被機能要件があり、機能要件はアプリケーションなどの必要条件を指定し、被機能要件はスケーラビリティ、可用性、応答性能、ネットワーク（内部と外部サービス）、セキュリティなど、アプリケーション以外の全てを Kubernetes クラスタ基盤で管理する。被機能要件をどのように実現しているかについては、実行可能なアプリケーションのベースイメージを最小単位のコンテナ（容器）に収容し、これらの集合体を Pod と位置付け、Kubernetes クラスタ内の Pod ネットワークへ接続することが可能になる。ユーザは最終的に Pod のコンテナにアクセスしてアプリケーションを実行していることになる。他に Pod の負荷や状態を監視し、負荷が増加すると Pod の数を自動的にスケールアウトして負荷を分散する。また、Pod に障害が発生した場合は、自己修復（セルフヒーリング）機能によりサービスの継続を維持し、指定した Pod のレプリカ数を常時稼働する。例えば、Pod 数を 3 と指定した場合、Pod の稼働数が 2 となれば新たに一つの Pod を起動し、Pod 数 3 を維持する。なお、Pod は一時的なリソースであり永続的にデータを保存できないため、外部にストレージシステムを用意し、Kubernetes のストレージクラスの機能を利用して、永続的にデータの保存を行う。

2.1 Kubernetes のコンポーネントとリソース

Kubernetes では、アプリケーションを管理するため以下の基本的なコンポーネントを構成し、各種リソースの管理を行う。図 2 に Kubernetes 概念図を示す。

コンポーネント:

- apiserver: Kubernetes API サーバーでクラスタ内のリソースの作成、更新、削除などの操作を受付
- scheduler: Pod をどのノードで実行するかを決定
- controller-manager: クラスタ内の様々なリソース（以下説明）の起動と状態を監視
- etcd: Kubernetes クラスタの永続的なデータ保存、key/value ストア

teiji.nakamura@kek.jp

- kubectl: apiserver へリソースの要求を指示
- kubelet: Pod の作成や管理（Pod のリソースやネットワーク）
- kube-proxy: 外部から Pod へのアクセスをルーティング

リソース:

- Configmap: アプリケーションの構成情報（環境変数、コンフィギュレーションファイル、コマンドライン引数などの設定を保存）
- Secret: 機密情報（パスワード、トークン、SSL 証明書など）Base64 エンコードされた値として保存
- Pod: 1 つ以上のコンテナをまとめた最小の実行単位
- Deployment: アプリケーションのレプリカセットを管理し、スケーリングやローリングアップデートを実施
- StatefulSet: 永続的な識別子とネットワーク識別子を持つ Pod を作成。データベースやキャッシュなどの状態を保存
- DaemonSet: クラスタ内の全ノードに Pod をデプロイ。ログ収集、ノード監視、ストレージデーモンなどのデーモンプロセスを実行
- Job/CronJob: 一回限り、または定期的なバッチジョブを実行
- Service: 一連の Pod にアクセスするための永続的なネットワークエンドポイントを提供
- Ingress: クラスタ外部からクラスタ内のサービスへの HTTP および HTTPS ルーティング（リバースプロキシ）
- Storage Class （SC）/Persistent Volume （PV）/ Persistent Volume Claim （PVC）: 永続的なストレージの管理と利用
- Role/ClusterRole/RoleBinding/ClusterRoleBinding: リソースへのアクセス制御用

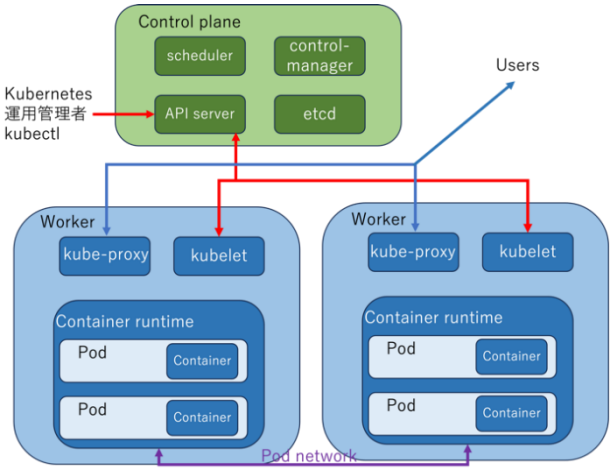

図 2. Kubernetes 概念図

2.2 Kubernetes の構築手順

- クラスタの準備: Kubernetes を構築する手順としては、環境や要件によって異なるが、インフラの準備としてコントロールプレーンとワーカーノード用の Linux 物理マシンや仮想マシンを用意する。
- コンテナランタイムのインストール: Kubernetes クラスタを構築する前に、Containerd[2]や cri-o[3]などのコンテナランタイムをインストールする。これは、クラスタ内のコンテナ化されたアプリケーションを実行するための基本的な要件である。
- Kubernetes ツールのインストール: kubectl（Kubernetes コマンドラインツール）や kubeadm（Kubernetes クラスタのセットアップツール）、Kubernetes を操作するためのツールと合わせて、kubelet（Kubernetes の中核的なコンポーネント）をインストールする。
- クラスタの初期化: kubeadm を使用して、コントロールプレーンを初期化し、Kubernetes クラスタの設定を行う。これにより、API サーバー、コントローラー、スケジューラーなどの Kubernetes コアコンポーネントがセットアップされる。
- ワーカーノード: kubeadm を使用してワーカーノードをクラスタに紐付け、ワーカーノードはコントロールプレーンと通信し、ポッドの実行などのタスクを受け入れることができる。
- ネットワークの設定: Kubernetes クラスタ内のポッド間の通信を確立するために、ネットワークプラグインをインストールし、設定する必要がある。Calico[4]などのネットワークプラグインが一般的に使用される。

3. Kubernetes クラスタの構築と要件

Kubernetes クラスタ基盤を構築するにあたりオンプレミス環境でミッドレンジのエンタープライズ機器を使用し、Kubernetes のコントロールプレーン 3 台とワーカーノード 4 台でクラスタリングを組み、ベース OS は全て Rocky Linux 9.2 を使用した。
データ保存として外部ストレージシステムを用意し、Kubernetes のストレージクラスの機能を利用して、永続的にデータの保存を行う。必要なパッケージソフトウェア（Kubernetes、CRI: Containerd、CNI: Calico、Helm[5]など）をインストールした。

4. ログ収集・分析環境の構築

ファイアウォールでは、threat（脅威）や URL filter のログは WebUI や syslog サーバで CLI（Command Line Interface）により確認することが可能であるが、リアルタイム検索、豊富なクエリ機能を有する Elasticsearch[7]、Kibana（データの可視化とダッシュボード作成ツール）、Logstash[8]（データ収集、変換、送信ツール）の 3 つを組み合わせたログ収集・分析環境を整備した。Kubernetes 上には Helm を利用して、Elasticsearch、Logstash、Kibana[9]の 3 つのオープンソースソフトウェアを組み合わせた ELK スタックを配備する。各々のコンポーネントが特定の役割を担い、ログ管理、監視、およびデータの可視化を行う。

Elasticsearch は、分散型の検索エンジンでリアルタ

イムに大量のデータの索引付けと検索を可能とする。Logstash は、データの収集、変換、および処理を行うデータパイプラインエンジンで、Kibana は、Elasticsearch に保存されたデータを可視化して分析するためのツールとして使用する。

4.1 ログ収集および可視化するための事前準備

ELK スタックに必要なパッケージのデプロイ:
Kubernetes 上の Logstash で threat 及び URL filter ログの取得設定:

- 外部データ受信用の LoadBalancerIP
- Pod の外部と内部の受信ポート、プロトコル
- syslog データの tags（threat、URL filter）付け、及び抽出
- フィールドの作成、型指定（日付、数値など）
- Elasticsearch 出力のインデックス指定、テンプレート作成

5. ログ監視運用

ログ収集は、Kubernetes 構築後の 2023 年 9 月から開始しファイアウォールの threat、URL filter のログを日々の運用で可視化している。threat ログでは、機構外から KEK DMZ 宛への脆弱性攻撃や SSH Brute Force attack が多く検知されているが、これらはファイアウォールによって遮断されている。ただし、Spyware カテゴリでは Critical 判定された Command and Control Traffic、Web shell Access、およびマイニングなどが検知され、これらは探索などに使用されている可能性がある。Spyware についても今後、遮断の対象として検討を行う。また、threat のログだけでは、一連の通信が分からず、traffic の通信も合わせて確認する必要があるため、今後については通信ログ（traffic ログ）も収集し、threat＋traffic で調査・分析出来る環境を構築する予定である。ELK スタックによるデータ収集を行ってから機構外部から KEK DMZ 宛の threat 遮断件数は約 7 万 8 千件であった。（図 3 に示す）

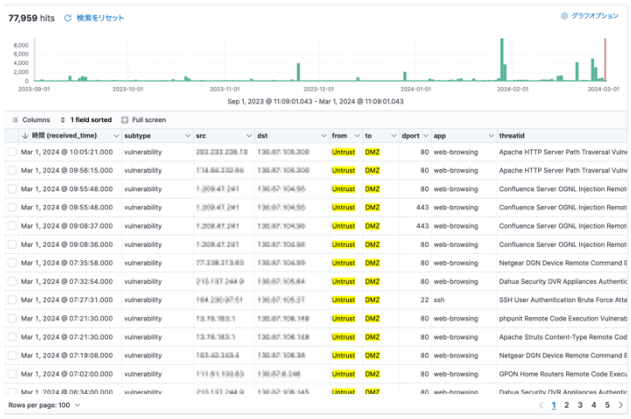

図 3. Untrust to DMZ threat reset

URL filter については、機構内から機構外へ web アクセスにおいてブロックされた件数は、約 14 万 1 千件であった。（図 4 に示す）98.8%が 80/tcp 宛と 1.2%が 443/tcp 宛となっていた。

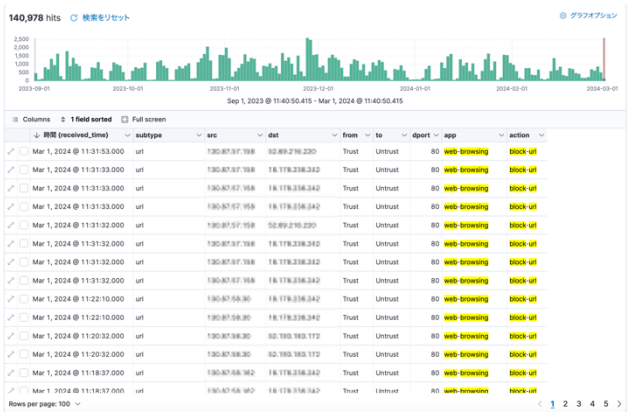

図 4. Trust to Untrust URL filter

6. まとめ

コンテナ型仮想環境として Kubernetes クラスタ基盤、およびログ収集・分析環境用の ELK スタックを構築した。従来のハイバーバイザー型と比較して、ハードウェアリソースの有効活用、クラスタ構築によるマシン管理の効率化、アプリケーション（サービス）構築プロセスの簡素化による効率化と迅速な環境構築が可能であることを確認した。

実際の運用では、ファイアウォールで検知される threat および URL filter のリアルタイム検索とクエリ要求によるログ分析、監視運用の利便性向上が行えた。その他サービスとして運用サーバのリソース・死活監視、UPS 監視システムなどを Kubernetes クラスタに統合することが出来た。

今後については、ファイアウォールの traffic ログを収集し、threat と合わせた分析環境を構築する。また、大局的な監視サマリー画面の作成や通知機能の実装（Grafana 等の活用）を行い、異常が検知された際には関係者に適切なアラートが送信され、迅速に対処するよう努める。

参考文献

[1] https://kubernetes.io/ja/
[2] https://containerd.io/
[3] https://cri-o.io/
[4] https://github.com/projectcalico/
[5] https://helm.sh/
[6] https://github.com/elastic/
[7] https://github.com/elastic/elasticsearch
[8] https://github.com/elastic/logstash
[9] https://github.com/elastic/kibana