利用用途の選択でソフトウェアを追加するKubernetes構成ファイルの自動生成

大野 有樹[1,a)]　串田 高幸[1]

概要：Kubernetes の構成ファイルを書くためには、概念を理解しなければならず、学習コストが高い．そこで、Kubernetes を用いた環境を基礎知識がなくても構築できるソフトウェアを作成する．質問で利用用途を明確にし、ソフトウェア構成をユーザに提案する．提案したソフトウェアに対応した構成ファイルをあらかじめ用意して、そのファイルを元に環境に合うように構成ファイルを書き換えて出力する．出力した構成ファイルを用いて環境を構築し、ユーザにアクセス情報を渡すことで基礎知識がないユーザでも環境構築ができるソフトウェアを提案する．

1. はじめに

背景

Kubernetes は Google が 2014 年にオープンソース化したプロジェクトのことをさす．Google 社内で利用されていたコンテナクラスタマネージャ Borg が基になって作成された OSS である [1]．コンテナを管理するソフトウェアで、類似するソフトウェアは Docker Swarm, Mesosphere が挙げられる．Kubernetes は Google や Microsoft や Amazon、IBM や VMware といった大手の企業が環境を提供している．今、最も活発なコンテナ管理システムといえる．

コンテナ

コンテナは、アプリケーションの (コンパイルされた) コードと、実行時に必要な依存関係をパッケージ化するための技術のことをさす．実行する各コンテナは再現性があり、依存関係を含めることによる標準化は、どこで実行しても同じ動作が得られることを意味する．コンテナは基盤となるホストインフラからアプリケーションを切り離すことにより、さまざまなクラウドや OS 環境下でのデプロイが容易になる技術である．

コンテナが生まれた背景として、VM がかなり大容量の計算リソースであることが挙げられる．VM は、物理的なハードウェアの上で動作するハイパーバイザの上で動作する OS のコピーであり、アプリケーションはその上で動作する．このため、VM はスピードやパフォーマンスに課題がある．コンテナはより軽量な計算リソースを生成するという問題を解決することを目指して誕生した．

コンテナは、オペレーティングシステムの上にハイパーバイザーを持つことができる．つまり、物理マシンや仮想マシンにたくさんのコンテナを搭載することができる．これにより、非常に多くのスケーラビリティを得られることになる．また、コンテナは非常に軽量なため、開発者は PC 上にコンテナを構築して、本番環境を再現することができる．開発者はそのコンテナに対してアプリケーションを構築して実行することができ、これらのコンテナを移動させることが可能になる．すなわち、どのような環境でも本番環境の再現が可能となることになる [2].

マイクロサービスアーキテクチャ

近年、ソフトウェアアーキテクチャの新しい用語として”マイクロサービス”がある．マイクロサービスアーキテクチャは Netflix,Spotify,Uber といった様々なサービスで採用されている [3]．マイクロサービスを作成するためにコンテナとコンテナを管理する Kubernetes がよく用いられている．マイクロサービスのアーキテクチャスタイルは単一のアプリケーションをサービスの集合体として開発するアプローチのことをさす [4]．それぞれが独自のプロセスで動作し，軽量なメカニズム，多くの場合は get や post といった HTTP リソースを用いた API で通信を行う．

今までのアプリケーションの変更サイクルは，アプリケーションの小さな部分に変更を加えた場合，システム全体を再構築してデプロイする必要があった．時間が経つにつれて，モジュール構造を維持するのは困難で，そのモジュール内の 1 つのモジュールにのみ影響を与えるはずの変更が他のモジュール構造を崩すこともある．さらにスケーリングにおいて，より大きなリソースを必要とする部

[1] 東京工科大学コンピュータサイエンス学部
〒 192–0982 東京都八王子市片倉町 1404-1
[a)] C0118050

分ではなく，アプリケーション全体に対する調整しかできなかった．

このような問題から，マイクロサービスのアーキテクチャスタイルが生まれた．サービスが独立してデプロイ可能でスケーラブルであるだけではなく，各サービスのモジュールの境界がはっきりとしていて，異なるサービスを異なるプログラミング言語で記述することさえできる．また，マイクロサービスアーキテクチャはサービスごとに独立してデプロイされているため，異なるチームで管理することも可能となる**?**.

Kubernetes

本番環境では、アプリケーションを実行しダウンタイムが発生しないように、コンテナを管理する必要がある．例えば、コンテナがダウンした場合、他のコンテナを起動する必要となる．このような動作がシステムに組込まれているため、Kubernetes によるコンテナ管理が簡単になるだろう．

Kubernetes は、コンテナ化されたワークロードやサービスを管理するための、ポータブルで拡張性のあるオープンソースのプラットフォームのことをさす．Kubernetes は分散システムを弾力的に実行するフレームワークを提供している．

Kubernetes の特徴として以下が挙げられる．

- サポートするアプリケーションの種類を制限しない
- アプリケーションレベルの機能を組み込んで提供しない

Kubernetes は非常に多様なワークロードをサポートすることを目的としている．そのため、アプリケーションがコンテナで実行できるのであれば、Kubernetes 上で問題なく実行できるようになっている．また、アプリケーションレベルの機能を組み込んでいない理由も同様で、多様なワークロードに対応するためあえて組み込まない選択肢を取っている．これらにより、弾力性を持ったシステムを Kubernetes は維持し続けている．

課題

Kubernetes を用いた環境構築をするためには Kubernetes の概念と設定項目を理解しなければならない．例えば、Pod と言われても何を指し示すのか分からなければ、Pod に関する設定を行うことができないだろう．例え、Kubernetes の概念を理解できていたとしても、実際に扱うとき設定項目をどの様に指定するか悩むことになる．また、Kubernetes の設定は yaml ファイルという設定ファイルを記述しなければならない．yaml ファイルは構造化データの表現方法のひとつで、設定ファイルに用いられる拡張子である．Kubernetes の設定は yaml ファイルで記述することを推奨している．すなわち、Kubernetes の環境構築は概念を理解したうえで、設定を指定し、yaml ファイル独自の記法で設定ファイルを記述することで初めて構築することができる．Kubernetes は使いこなせるようになるために踏まなければならない段階が多く、とっつきにくい印象を受ける．さらに、サーバーに関する基礎知識や設定項目に関する理解が足りていない場合、yaml ファイルを記述しても設定が適切でないために起動できない、もしくは思い描いた動作をしないという課題がある．Kubernetes は拡張性が高く、サポートするアプリケーションの種類を制限しない．アプリケーションがコンテナで実行できるのであれば、Kubernetes 上で問題なく実行できる．しかし、拡張性が高いということは選択肢が多いことになる．従って、選択肢が多すぎて自分が作成したい環境を構築するために、何をすれば良いのか判断に困るという課題がある．

各章の概要

2 章では関連研究について説明する．3 章では本論文におけるソフトウェアの提案を示す．4 章では実装と開発環境について説明する．5 章は評価・分析,6 章に議論，7 章にまとめで締める．

2. 関連研究

この章では本研究と関連した先行研究について取り上げ、その関連性について述べる．ミドルウェア用のグリッドサービス設定ソフトウェアツールとして YAIM が挙げられる．このツールの目的は、グリッドサービスのインストールと設定を可能な限り簡単にすることである．このツールは生成された構成ファイルを環境に合わせて書き換える必要がある [5].

YAIM を用いた自動で構成を作り出すシステムとしては、構成管理ツール Puppet を用いた VM における Linux インストールと設定を自動化した提案がある．この提案では Puppet に構成ファイルを渡すことでシステムの自動インストールを可能にしたである [6]．仕組みは違えど、Kubernetes はすでに構成ファイルによる環境構築の展開ができる．双方に言えることとして構成ファイルを記述するということは、構成ファイルの書き方を知る必要があり、とっつきにくさを生みだしているといえる．

ソフトウェアの環境構築を簡易化しようという試みは環境構築自体を簡易化する以外にも方法がある．それは構成マニュアルを自動生成する方法だ．この先行研究は、OSS のインストールと初期構成は、適切なマニュアルが一般的に不足しているため、初心者ユーザーにとっては難しいというところに焦点を当てている．この提案は熟練したユーザーがインストールプロセス中に記録したログ情報を編集することにより、OSS の Web ベースのインストールマニュアルを自動的に生成する方法を提案したものだ．結果、学習効果に違いを出さずに費やされる時間の削減に成功した [7]．この提案はより短時間で学習して環境を構築

する上では有用といえる．しかし、私が目指していることは学習コストを限りなく無くすことである．なので、学習するといった行為自体がない提案が望ましい．

3. 提案

本研究では、ふたつの課題、Kubernetes を用いた環境構築をするためには Kubernetes の概念と設定項目を理解しなければならない．また、やりたいことが定まっていたとしても拡張性が高いためにどのソフトウェアを使用するか悩む事態を解決する提案として、構成ファイル即ち yaml ファイルを記述する工程を自動化する手法を提案する．

本提案によって、Kubernetes に関する予備知識無い人が環境構築することを可能にする．本提案の利点として、例え設定することが苦手だったとしても設定を意識することなく生成できるだけでなく、設定ファイルに関する知識なく設定でき、習熟していなくても作成できるようになる．本提案はサーバー基盤のソフトウェアを作成するエンジニアの中で Kubernetes に関する知識もしくはサーバーに関する知識に疎い人を対象としたものである．

本手法は 3 つの機能から構成される．

(1) 質問と答えをユーザとやり取りする
(2) 構成ファイルを生成し、Kubernetes による環境構築を行う
(3) 作成した環境にアクセスするための情報をユーザに渡す

アーキテクチャを図 1 に示す．

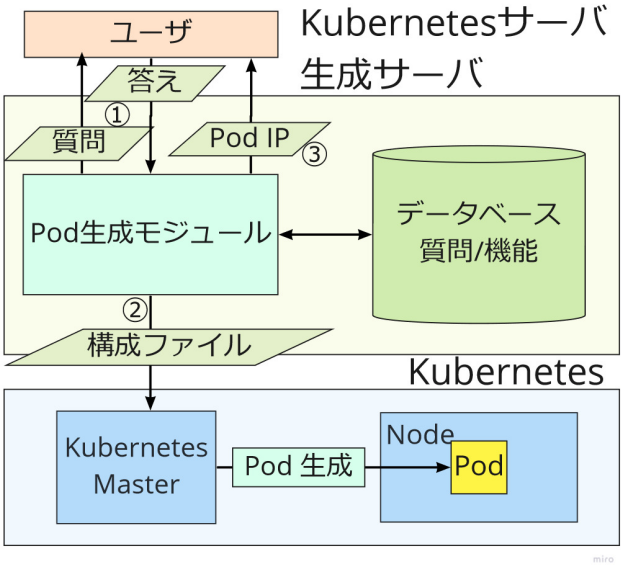

図 1 アーキテクチャ図

「Pod 生成モジュール」が本提案のソフトウェアである．

ひとつ目の機能である「質問と答えをユーザとやり取りする」は「ユーザ」が「Pod 生成モジュール」を通して質問に回答させる機能である．「質問」はデータベースに保存されていて、「Pod 生成モジュール」によって表示する質問を制御する．質問の結果から必要となるソフトウェアリストを作成し、構成ファイルの基とする．

ふたつ目の機能である「構成ファイルを生成し、Kubernetes による環境構築を行う」は使用するソフトウェアに合わせて構成ファイルを作成し、Kubernetes に Pod を生成するように命令する．ひとつ目の機能から送信されたソフトウェアリストから必要となる構成ファイルのデータをデータベース「機能」から持ってきて作成する．作成された構成ファイルを対象とした Kubernetes の Pod 生成コマンドを実行し、「Kubernetes Master」に「Pod」を生成するように指示する．

みっつ目の機能である「作成した環境にアクセスするための情報をユーザに渡す」はふたつ目の構成ファイル作成時に定めた情報から Pod にアクセスするための「IP」アドレスを「ユーザ」に渡す．このアドレスを用いてユーザは Pod にアクセスする．

4. 実装と実験環境

4.1 実装

実装は以下のの 5 つで構成する．

(1) 質問を行う
(2) 答えを渡す
(3) 答えから構成ファイルを作成する
(4) Pod を生成する
(5) 出来上がった環境へのアクセス方法を返す

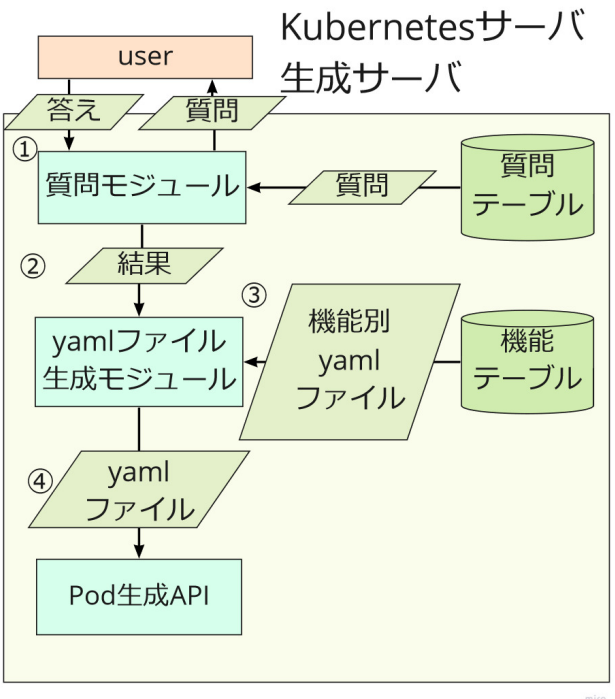

図 2 質問ソフトウェア実装図

まずひとつ目の「質問を行う」はデータベースに保存された質問をユーザーに問い、答えを溜めておく機能を持つ．質問内容と選択肢は web ブラウザに表示される．

その後、質問結果と使用するソフトウェアの情報をデータベースに保存し、「構成ファイルを作成する」にも同様の情報を送る．

次に使用するソフトウェアを起動するための構成ファイルをデータベースから取ってきて構成ファイルを生成する．ここで CMS やデータベースの設定を行う．構成ファイルには一定の書き方があるため、連携するための記述を変数として設定をし、使用するソフトウェアに合わせて代入することで様々な構成ファイルの作成をすることができるようにする．

作成した構成ファイルを用いて Kubernetes の master、すなわち Pod を生成する API に Pod を作成してもらう．

最後に構成ファイルで設定したアクセスするための情報をユーザー（web ブラウザ）に返すことで環境構築が完了する．

4.2 質問モジュール

質問のデータベース構造は以下のようになっている．

表 1 質問

ID	tire	contents
1	1	どの用途で使用しますか?
2	0	WordPress をお勧めします。
3	2	公開するゲームを教えてください。
4	2	MySQL(SQL) か MongoDB(NoSQL) どちらにしますか?
5	0	minecraft server のお勧めです
6	0	phpmyadmin をお勧めします
7	0	mongo-express をお勧めします

表 2 選択肢

ID	contentsID(FK)	nextID	choices
1	1	2	サイトの公開
2	1	3	プログラムの公開（ゲームサーバ）
3	1	4	データの管理・保管（テキストデータ）
4	3	5	Minecraft サーバ
5	4	6	MySQL サーバ
6	5	7	MongoDB サーバ

表示される文と属性を保存した表 1 と質問の選択肢と質問の ID を保存した表 2、使用するソフトウェアを保存した表 3 のみっつの表を用いている．

始めに「ID」が”1”の「contens」を表示する．同様に「contentsID(FK)」が”1”の「choices」を表示する．ユーザが表示された「choices」からひとつを選択すると、「質問モジュール」が選ばれた「choices」の「nextID」を参照し、それに対応した「contens」を表示する．もし「tire」が”0”でないならば、「nextID」に対応した「choices」を表示し、「contens」と「choices」の表示を繰り返す．このときもし、「tire」が”0”であるならば、「nextID」に対応した「software」を表示し繰り返しは終了する．出力された「software」はリスト化され、「yaml ファイル生成モジュール」「結果」として渡される．

表 3 答え

ID	contentsID(FK)	software
1	2	mysql:5.7.33
2	2	wordpress:latest
3	5	openjdk:11.0.10-nanoserver
4	6	phpmyadmin:fpm-alpine
5	7	mongo-express:0.54

4.3 実験環境

実験環境を図 3 に示す．

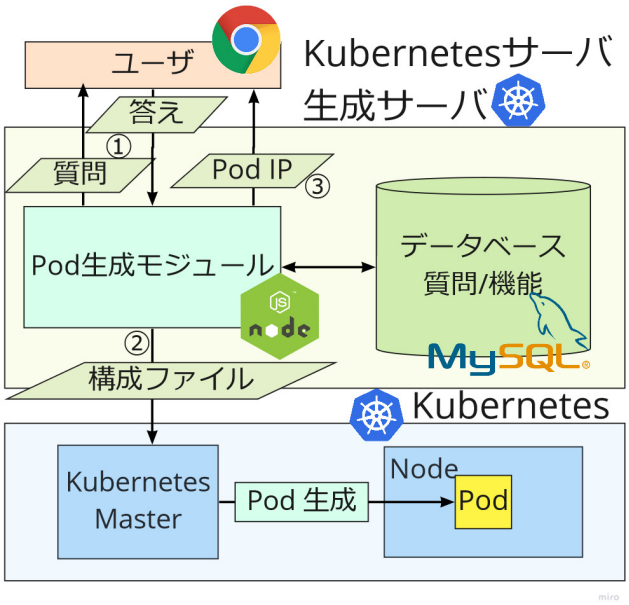

図 3 実験環境図

使用方法として, ユーザーがブラウザ (e.g. google chrome) を用いてアクセスする．開発したソフトウェアは node.js を用いて作成した．DB は MySQL を用いて行い, アクセスは node.js のライブラリ express を使用した．作成される環境は Kubernetes から作られたものとなる．

5. 評価と分析

本提案によって、Kubernetes を用いた環境構築を基礎知識がなくともできるソフトウェアを実現した．評価は以下の 2 点において行う．

(1) このソフトウェアによって設定できる項目数

(2) このソフトウェアの使いやすさ

5.1 実施

評価は研究室のメンバーを対象に行った．

(1) 作成しようとしている環境のアンケートをとる
(2) 実際に触ってもらう
(3) 使いやすさを評価してもらう
(4) 自動で生成された設定項目数の数を計測

6. 議論

本提案によって、Kubernetes に関する知識が無くとも環境構築ができるソフトウェアを作成した．しかし、この提案ではシステムの維持に関する記述がないので以下に載せる．

ひとつ目は質問や環境の拡張である．現在、質問の数と対応できる構成の範囲は足りていない．質問と環境を拡張できる機能が必要となる．なので、本提案ではプログラムに埋め込んだり、テキストファイルを使用せず、データベースを使用している．

ふたつ目はログデータの保存である．ログデータを保存することでユーザがどの様な環境を必要としているのかをデータをとることで明確にすることができる．

みっつ目はユーザ管理である．ユーザを管理することで、質問や環境を追加する人が管理者だけではなくなり、ユーザも追加更新できるようになる．ユーザが質問や環境を追加できるようになることで、拡張性が大幅に広がる．それだけではなく、ログデータにユーザデータを入れることで誰がどうのような環境を欲したのかといった情報が取得でき、サービスの向上を図ることができる．ログデータにユーザデータが加わることでログデータの分析がより詳細になるだろう．

7. おわりに

本稿では、Kubernetes に対する知識がなくとも、質問に答えることで使用するソフトウェアを明確にし、Kubernetes による Web サーバーの環境構築ができるソフトウェアを提案した．Kubernetes の問題として、使用するソフトウェアを選ばなければならない．また、構成ファイルを作成する際に、更新頻度が高いためインターネットに公開されている構成ファイルをコピー＆ペーストしたとしても、サポートが切れているため動かないときがあるという課題がある．このソフトウェアによって、使用するソフトウェアの選定や更新頻度に合わせた構成ファイル更新ができるため、記述するための事前知識が必要なくなった．ただし、サービスとして公開するにはまだまだ問題点が多く、改善の余地があるシステムとなった．

参考文献

[1] Verma, A., Pedrosa, L., Korupolu, M., Oppenheimer, D., Tune, E. and Wilkes, J.: Large-scale cluster management at Google with Borg, *Proceedings of the Tenth European Conference on Computer Systems*, pp. 1–17 (2015).

[2] Anderson, C.: Docker [Software engineering], *IEEE Software*, Vol. 32, No. 3, pp. 102–c3 (online), DOI: 10.1109/MS.2015.62 (2015).

[3] Jamshidi, P., Pahl, C., Mendonça, N. C., Lewis, J. and Tilkov, S.: Microservices: The Journey So Far and Challenges Ahead, *IEEE Software*, Vol. 35, No. 3, pp. 24–35 (online), DOI: 10.1109/MS.2018.2141039 (2018).

[4] Balalaie, A., Heydarnoori, A. and Jamshidi, P.: Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture, *IEEE Software*, Vol. 33, No. 3, pp. 42–52 (online), DOI: 10.1109/MS.2016.64 (2016).

[5] Jayalal, M., Rajeswari, S. and Murty, S. S.: Application of YAIM tool in Grid computing, *Seminar on Applications of Computer & Embedded Technology, October 28-29, 2009, VECC, Calcutta* (2009).

[6] Magherusan-Stanciu, C., Sebestyen-Pal, A., Cebuc, E., Sebestyen-Pal, G. and Dadarlat, V.: Grid System Installation, Management and Monitoring Application, *2011 10th International Symposium on Parallel and Distributed Computing*, pp. 25–32 (online), DOI: 10.1109/ISPDC.2011.14 (2011).

[7] Murakami, Y., Kagawa, E. and Funabiki, N.: Automatic Generation of Configuration Manuals for Open-Source Software, *2011 International Conference on Complex, Intelligent, and Software Intensive Systems*, pp. 653–658 (online), DOI: 10.1109/CISIS.2011.109 (2011).