ユーザの使用目的に合ったKubernetesの環境を構築する手法

大野 有樹[1,a)]　串田 高幸[1]

概要：Kubernetes を用いたアプリケーションの実行環境を構築するとき，Pod，Service，Deployment といった Kubernetes におけるオブジェクトを理解することと，構成ファイルの記述方法を知る必要がある．この 2 つを理解しなければ Kubernetes の環境構築ができないことは課題である．本研究では，Kubernetes の構成ファイルを記述する工程を自動化する手法を提案する．手法としては，ユーザへ質問を問いかけ，ユーザが使用したいアプリケーションの利用目的を明確にする．最後に，利用目的に最も近いアプリケーションの環境を提供する．本提案に対してアンケートによる評価を行った．Q1「Kubernetes を意識しないで環境を構築できたか？」の質問に対して 100%意識しないで環境を構築できたと回答した．Q2「目的と合った環境を構築できたか？」の質問に対して 50%がぴったり合っていると回答したものの，合っているが 25%，どちらともいえないも 25%だった．これは実験のために用意した環境が 3 種類しかないため，目的に合った環境を提供できたとは言えない結果になったが，Kubernetes を理解していなくても環境構築ができる手法であると分かった．

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である．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 を基に作成された OSS（Open Source Software） 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 が挙げられる [2]．Kubernetes は大手の企業（e.g. Google や Amazon，IBM）が環境を提供している [3]．今，最も活発なコンテナ管理システムといえる．

Kubernetes

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である．Kubernetes は分散システムを状況に応じて柔軟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 [4]．

Kubernetes のオブジェクトを一部あげると Pod，Service，Deployment がある．Pod はアプリケーションを実行する場所でコンテナを内包している．Service は Pod を外部公開するためのオブジェクトである．Deployment には Pod を制御する役割があり，Pod を生成するときに Pod 単体で起動するのではなく Deployment を用いて Pod を起動する．

図 1 は Kubernetes のデプロイ方法である．Kubernetes のデプロイ方法は yaml ファイル（図 1 では wordpress.yaml）に Kubernetes の API である kubectl を用いて Pod をデプロイする．

課題

Kubernetes 未経験ユーザがコンテナと Kubernetes を用いてアプリケーションの実行環境を構築をするとき，3 つの課題がある．

(1) 構成ファイルの記述方法を覚える必要がある

[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

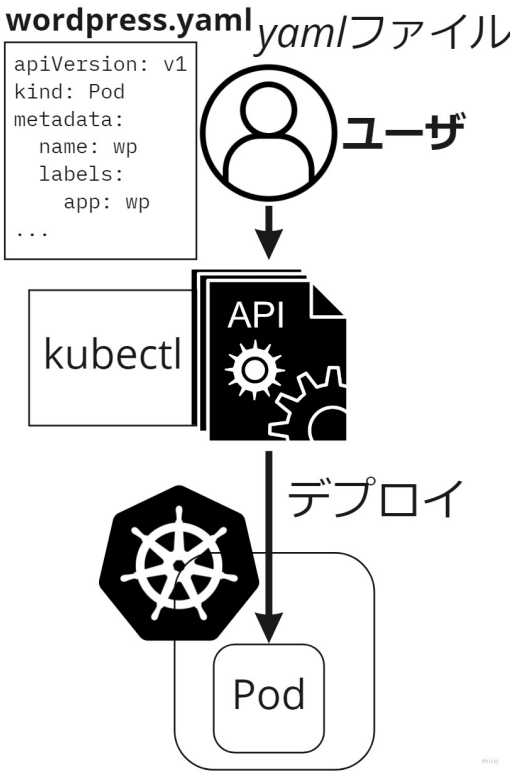

図 1 Pod の生成方法

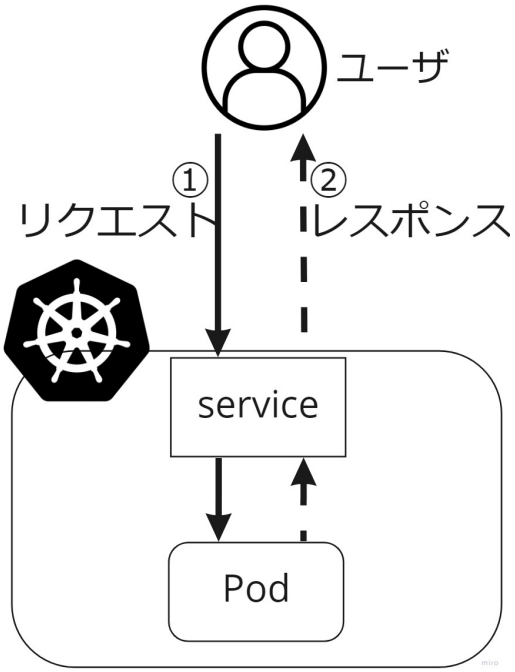

図 2 ユーザと Pod のやり取りの方法

(2) Kubernetes のオブジェクトを理解する必要がある
(3) 使用するコンテナを選ぶ必要がある

Kubernetes の設定項目は構成ファイルで記述する．Google は Kubernetes の設定のベストプラクティス[*1]より構成ファイルに yaml ファイルを用いることを推奨している．yaml（YAML Ain't a Markup Language）は構造化データの表現方法，すなわちデータ形式言語のひとつで設定ファイルに用いられる拡張子である．yaml はの記法に合わせた記述をする必要がある．例えば，yaml ファイルはインデントをスペース 2 つにしなければ読み取りに失敗するといったことである．

また，Kubernetes の構成ファイルを記述するためには Kubernetes の Pod，Service，Deployment といったオブジェクトを理解する必要がある．構造を理解していない場合，構成ファイルを記述してもユーザの想定通りに起動できない課題がある．例えば，Kubernetes で Web ページを公開しようとしている人がいたとする．図 2 は Kubernetes の Pod は Service を通してユーザと通信をすることを表している．従って，Pod をたてることに成功したとしても，Service が無ければ Pod 内の Web ページへの URL が分からずに Web ページへアクセスできないことが挙げられる．

Kubernetes は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．Kubernetes はユーザにとってアプリケーションの選択肢が多く，目的のためにどのアプリケーションを使えばよいかユーザが判断する必要がある．例えば，商品宣伝用の Web ページを公開することを目的にしているとすると，Web ページ公開のためにどのアプリケーションを使用すればよいか，選択肢が多いゆえに Kubernetes の未経験者は使うコンテナを定めることができない．

この様な「構成ファイルの記法」「Kubernetes のオブジェクト理解」「コンテナの選択」の課題があり，ユーザが Kubernetes 環境を構築するための学習と操作が必要になる．

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 がある．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自動化することである．YAIM はテンプレートファイルを作成し自動インストールを行う [5]．Puppet というインフラストラクチャと複雑なワークフローを管理および自動化を行う構成管理ツールがある．この Puppet に先述の YAIM の構成ファイル生成システムを組み合わせて，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 [6]．Kubernetes は構成

[*1] https://kubernetes.io/docs/concepts/configuration/overview/

ファイルを記述しデプロイすることで環境を構築するため，YAIM と類似した環境構築の手法といえる．YAIM と Puppet を組み合わせた手法を見ると，構成ファイルを生成してファイルに記述された内容に合わせてインストールを実施する提案は自動化において有用性があることが分かる．しかし，この提案は複数の同じ環境を作り出すための手法であり，様々な種類の環境を作り出す手法として Puppet だけでは十分ではない．

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．この提案は熟練したユーザ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である．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．学習時間が削減した理由は習熟したユーザがマニュアルを作成したことにある．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不足しているため，未経験者ユーザにとっては難しい点に焦点を当てている．習熟したユーザによる未経験のユーザに対するサポートは環境構築を簡易化するうえで重要である．この提案はマニュアル作成をするための提案であり環境構築の自動化の提案ではないため，環境構築へ熟練ユーザが未経験ユーザをサポートする提案が必要である．

既存のアプリケーションを，拡張性があり，可搬性があり，かつ継続的にデプロイ可能なマイクロサービス・システムに変換するプロセスを紹介した論文がある．開発者がアプリケーションを 4 つに分類しパッケージ化した後，コンテナとして Kubernetes でデプロイする．最後に必要な呼び出しをすべてリストにまとめて最終的にひとつの呼び出しで実行することでサービス間の呼び出しを最小限に抑えるアルゴリズムのガイドラインを作成するという手順を踏む [8]．この手法の問題点として，開発者がアプリケーションを分類する必要がある．アプリケーションを分類するためにはアプリケーションの構造を理解したうえで分解する必要がある．Kubernetes 環境に落とし込むまでのプロセスを自動化ができないところに課題がある．

Kubernetes におけるインフラストラクチャの自動化を提案した論文がある．Terraform，Helm Charts，HyperFlow を使用してインフラストラクチャのプロビジョニングプロセスを簡素化している [9]．しかし，この提案ではツールを使うためのツールを使うため，Kubernetes 以外のツールを使えるようになるための学習が必要になる．Kubernetes を簡易化するために他のツールの学習する必要があることは，たとえ構造の簡易化ができたとしても Kubernetes を使えるようになるための簡易化ができていないところに課題がある．

3. 提案

本研究では，Kubernetes の構成ファイルを記述する工程を省略化する手法を提案する．手法としては，ユーザへ質問を問いかけ，ユーザが使用したいアプリケーションの利用目的を明確にする．最後に，利用目的に最も近いアプリケーションの環境を提供する．

まず初めに，本提案はなぜ Kubernetes でこの手法を行うのか．理由は 2 つあり，ひとつは可搬性，もうひとつは可用性である．コンテナは可搬性があり，同じコンテナエンジンがあればどの環境でも一様に実行できることが利点である．また Kubernetes は可用性があり，Pod が何かしらの原因で終了したとしても Pod の再起動を行うため，ユーザが意識せずに復旧を行うことができる．

次に，質問をする理由はツールを使う際に必要な知識を減らすためと，ユーザが求めている環境と本提案のソフトウェアが提供する環境の違いをできる限り無くすためである．ツールを使う際に必要な知識を減らす理由は，アプリケーションを簡易化するためのソフトウェアが簡易化するためのソフトウェアを使うために知識が必要な場合があり，簡易化できているとは言えないからである．ユーザと提案の環境の差を無くす理由は，本提案で自動的にアプリケーションの環境を提供できたとしてもユーザが求める環境と違う環境では意味がないからである．

質問文は専門用語をできる限り排除したものにする．それは，先述の通りアプリケーションの簡易化に学習を必要とするソフトウェアは簡易化したとは言えないからである．質問で問う内容はアプリケーションで使うことのできる機能ではなく，アプリケーションを使用する目的を問うものに限定する．なぜなら，アプリケーションの実行環境が欲しいユーザはアプリケーションを使用する目的を持っているが必ずしもこのアプリケーションの機能が欲しいと具体的なイメージを抱いているわけではないからである．さらに，ユーザは質問を通して目的を明確にしながら回答することができるが，機能の場合，機能を調べ，目的が合うかどうか判断し，回答するといった手間がかかり，質問回答に滞りが起こる．だから質問文はアプリケーションの利用目的を問うものに限定する．

本提案によって，Kubernetes 未経験ユーザが環境構築することを可能にする．本提案の利点として，例え構成ファイルに関する知識がなくとも構築できるため，構成ファイルを意識することなくアプリケーションの実行環境を生成できるようになる．

本提案は 9 つの段階から構成される．アーキテクチャを図 3 に示す．

(1) ユーザが Web ブラウザを通してソフトウェアへアクセス

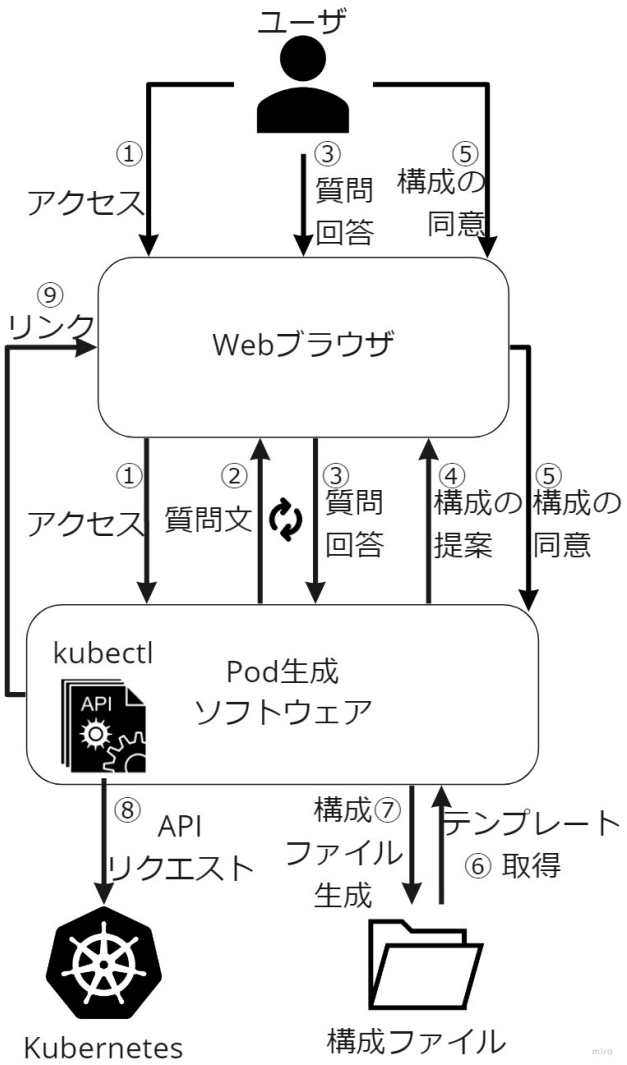

図 3 アーキテクチャ図

(2) 質問をユーザへ送る
(3) 質問の答えをユーザから受け取る
(4) ユーザの目的に類似した構成をお勧めする
(5) 提示した構成にユーザが同意する
(6) テンプレートの構成ファイルを取得する
(7) テンプレートを基に構成ファイルを生成
(8) API リクエストを用いて Kubernetes による環境構築を行う
(9) 作成した環境にアクセスするためのリンクをユーザに渡す

以下に図 3 の説明を述べる．1 つ目の「ユーザが Web ブラウザを通してソフトウェアへアクセス」はユーザが Web ブラウザから URL を打ち込み，ソフトウェア即ち Web アプリケーションへアクセスすることを示している．2 つ目の「質問をユーザへ送る」はソフトウェアから質問を Web ブラウザを通してユーザへ送信している．3 つ目の「質問の答えをユーザから受け取る」はユーザが 2 つ目の「質問をユーザへ送る」で送信された質問に対して，ユーザの答えを受け取ることを示している．2 つ目と 3 つ目は繰り返し行い，ユーザが求める環境を明確にすることを目的としている．4 つ目の「ユーザの目的に類似した構成をお勧めする」は 2 つ目 3 つ目によって明確にしたユーザの目的に最も近い構成をユーザへ提案する．5 つ目の「提示した構成にユーザが同意する」は 4 つ目のお勧めの提示に対して，ユーザがお勧めを同意するかを問いている．同意した場合構成の生成へ移るが，同意しなかった場合 3 つ目に戻り質問を最初からやり直してもらう．6 つ目の「テンプレートの構成ファイルを取得する」はストレージにあらかじめ用意しておいた構成ファイルを読み取り，ソフトウェア内の変数に代入する．7 つ目の「テンプレートを基に構成ファイルを生成」は 6 つ目の変数を用いて新たに構成ファイルを作成する．8 つ目の「API リクエストを用いて Kubernetes による環境構築を行う」は 7 つ目で作成した構成ファイルを基に API リクエストを用いて Kubernetes による環境構築を行う．9 つ目の「作成した環境にアクセスするためのリンクをユーザに渡す」は 8 つ目で Kubernetes によって構築された環境へアクセスするための URL を Web ブラウザを通してユーザへ渡す．これら 9 つの段階を踏むことで Kubernetes による環境構築が完了する．ユーザはこのソフトウェアに対して Web ブラウザでアクセスし，質問に回答し，お勧めされた構成に同意するという操作のみになっていて，Kubernetes に関する知識がなくても環境構築ができるようになっている．

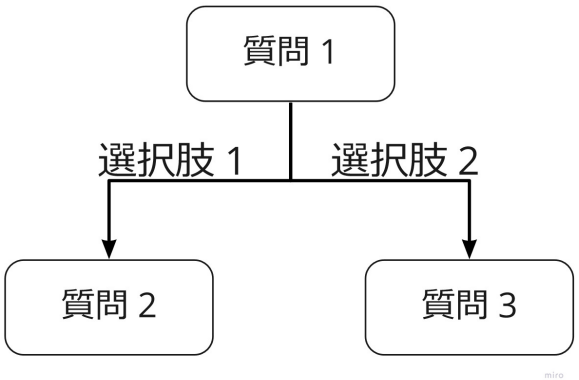

図 4 質問システムアーキテクチャ図

本提案のシステム「Pod 生成ソフトウェア」の内の質問のやり取りを行う機構を「質問システム」とする．「質問システム」の アーキテクチャ図を図 4 に示す．表示する質問はユーザの回答によって異なり，ユーザがどの選択肢を選ぶかによって分岐式で次の質問が変わる．質問を木構造の様に設定することで，質問の選択肢を増やしたり，下の階層に質問を追加したりと質問を拡張することができる．第 1 階層はユーザの利用用途を問う質問を用意し，階層が進むごとに質問の内容を詳細にしていく．質問の例として，「どの目的で使用しますか？」の問いに対して選択肢

は「Web サイトの公開」「ゲームサーバの公開」「テキストデータの管理」となる．

4. 実装と実験環境

4.1 実装

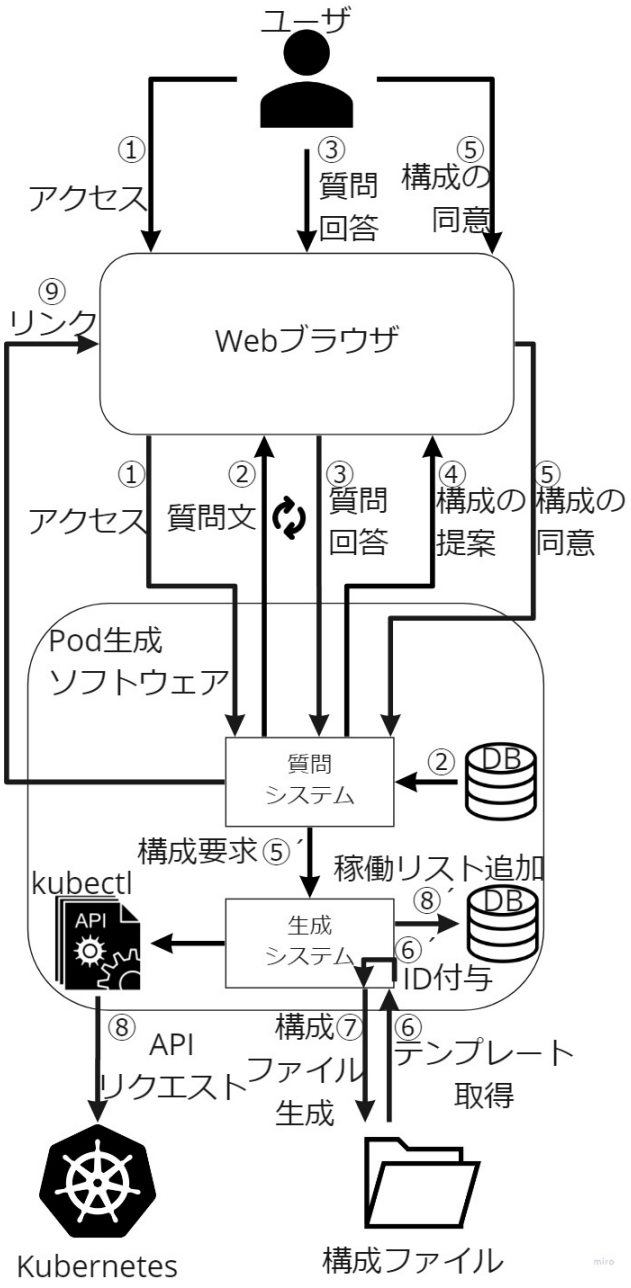

図 5 実装図

実装は以下の 9 つで構成する．実装を図 5 に示す．

(1) ユーザが Web ブラウザを通してソフトウェアへアクセス
(2) DB に保存している質問をユーザへ送る
(3) 質問の答えをユーザから受け取る
(4) ユーザの目的に類似した構成をお勧め
(5) 提示した構成にユーザが同意した後に質問システムから生成システムへ選んだ構成情報を送信
(6) テンプレートの構成ファイルを取得して ID を付与
(7) テンプレートを基に構成ファイルを生成
(8) API リクエストを用いて構成ファイルを基に Kubernetes による環境構築を行い，稼働リストに構成名と ID とリンクを追加する
(9) 作成した環境にアクセスするためのリンクをユーザに渡す

(1)，(3)，(4)，(7)，(9) は 3 章提案で示したものと説明が同じであるため省略する．(1)，(3)，(4)，(5)，(9) は Pod 生成ソフトウェアの中でも質問システムのやり取りで，(6)，(7)，(8) は生成システムのやり取りとなる．以下に (2)，(5)，(6)，(8) の説明を述べる．(2) の「DB に保存している質問をユーザへ送る」は DB に予め保存してある質問を Web ブラウザを通してユーザへ送信している．(5) の「提示した構成にユーザが同意した後に質問システムから生成システムへ選んだ構成情報を送信」は 4 つ目のお勧めの提示に対して，ユーザがお勧めを同意するかを問いている．同意しなかった場合 3 つ目に戻り質問を最初からやり直してもらう．同意した場合質問システムから生成システムへ構成情報を送信する．構成情報は DB に保存されていて，2 つ目と同じ方法で情報を取得している．(6) の「テンプレートの構成ファイルを取得して ID を付与」は予め保存してある構成ファイルのテンプレートを読み取り，ソフトウェア内の変数に代入する．構成ファイルのテンプレートには ID を振るための変数を用意しているので，ランダム生成した ID を変数に代入することで ID を付与する．(8) の「API リクエストを用いて構成ファイルを基に Kubernetes による環境構築を行い，稼働リストに構成名と ID とリンクを追加する」は 6 つ目で ID を付与した構成ファイルを指定して API リクエストを用いて Kubernetes に構成を構築してもらう．構築が終われば，稼働リストを表示するための DB へ構成名と ID とリンクを追加する．ユーザはいつでもこの稼働リストの DB を参照することで起動中の構成を把握することができる．以上 9 つの段階を経て環境構築が完了する．

4.2 質問システム

質問システムのシーケンス図を図 6 に示す．この図はユーザが表示された質問に対して選択肢を選んだところから始まる．すなわち図 5 で表すなら (3) から始まる．ブラウザは質問モジュールに対して次の質問 ID を送信する．次の質問 ID を受け取った質問モジュールは DB に対して ID 検索をかける．DB は次の質問 ID と一致する tire（階層），内容（質問），選択肢，次の質問 ID を送信する．このとき tire が 0 のとき内容（お勧め），ソフトウェア名を検索結果として返信する．DB から送られてきた内容を質問モジュールはブラウザに送信してユーザが見れるようにする．

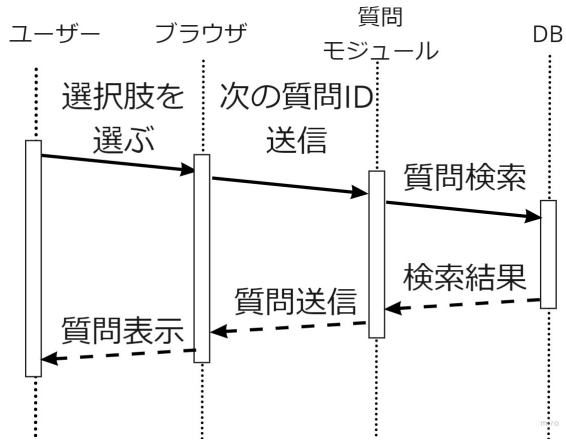

図 6 質問システムシーケンス図

質問のデータベースの具体例を以下に挙げる．表は表示される文と文の階層を保存した表 1 と対応する contentsID と次の質問の ID，質問の選択肢を保存した表 2，対応する contentsID と使用するソフトウェアを保存した表 3 の 3 つの表を用いている．

表 1 質問

ID	tire	contents
1	1	どの目的で使用しますか?
2	0	WordPress をお勧めします。

表 2 選択肢

ID	contentsID(FK)	nextID	choices
1	1	2	Web サイトの公開
2	1	3	ゲームサーバの公開
3	1	4	テキストデータの管理

表 3 答え

ID	contentsID(FK)	software
1	2	mysql:5.7.33
2	2	wordpress:latest

始めに表 1 の「ID」が”1”の「contens」を表示する．同様に表 2「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，「生成モジュール」に構成リストを渡す．

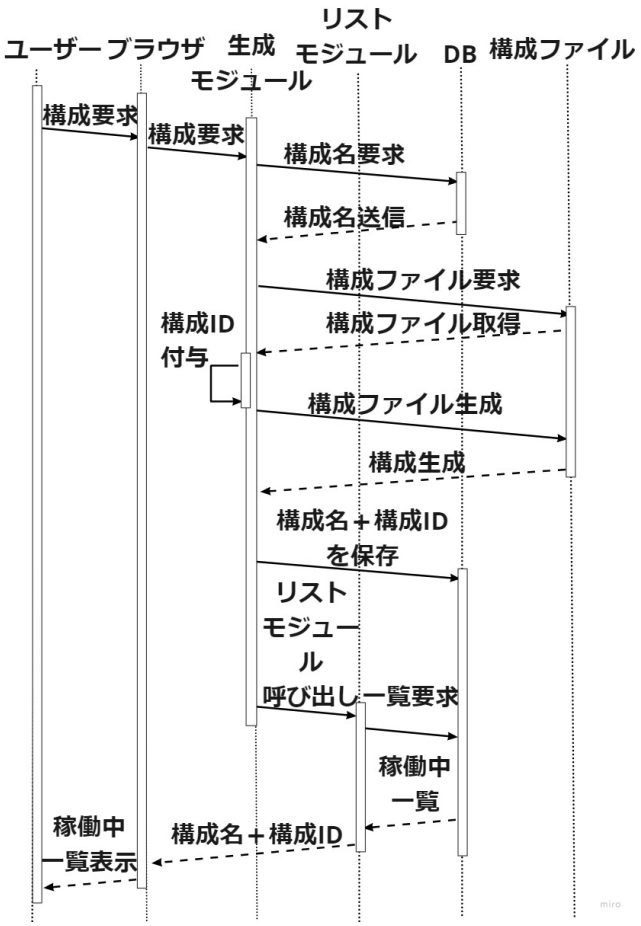

図 7 生成システムシーケンス図

4.3 生成システム

生成システムのシーケンス図を図 7 に示す．この図はユーザがお勧めの構成を同意して Kubernetes による構成を生成するところから始める．すなわち図 5 で表すなら (5) から始まる．

最初に作成する構成名を DB から取得する．この構成名は環境構築が完了した際にユーザがどの構成を起動しているかを分かるようにするためにつける名称である．次に構成ファイルを要求して，構成 ID を付与する．これは図 5 の (6) にあたる．ID を付与した理由は同じ種類の構成を構築したとき，ユーザが区別できるようにするためである．次に ID を付与した構成ファイルを生成する．これは図 5 の (7) にあたる．そして，作成された構成ファイルを基に Kubernetes による環境構築を行い，構成名と ID を DB に保存する．これは図 5 の (8) にあたる．最後にリストモジュールを呼び出し，構成名と ID，構成へアクセスするためのリンクをブラウザへ表示を行う．これは図 5 の (9) にあたる．

4.4 実験環境

実験環境を図 8 に示す．検証環境は Ubuntu 18.04 をインストールした VM 上に Node.js で作成した Pod 生成ソ

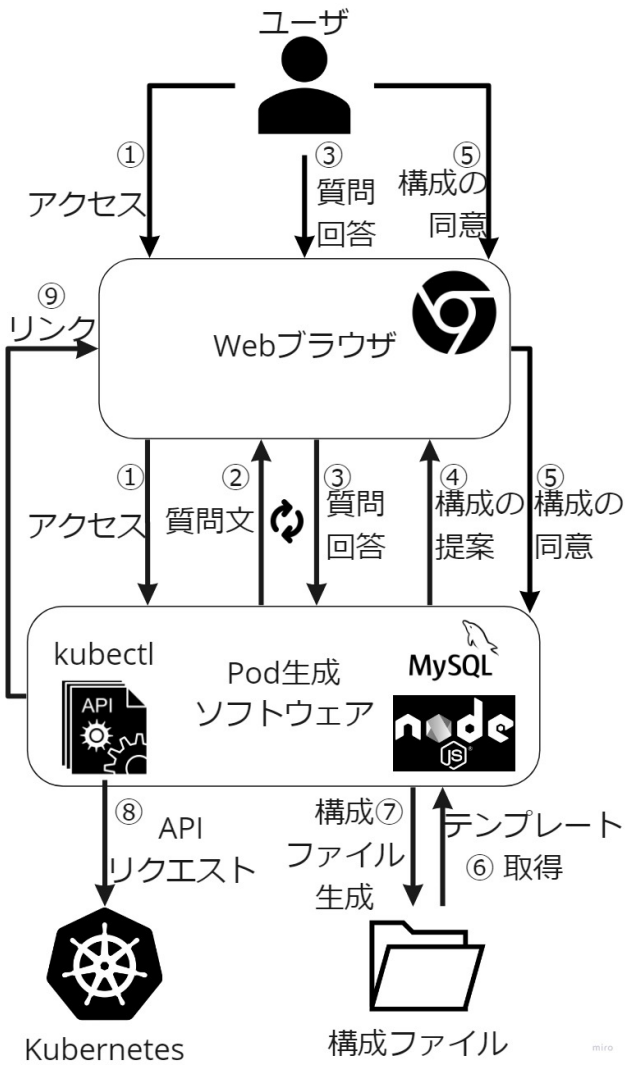

図 8 実験環境図

フトウェアと MySQL のコンテナを Docker で作成した．また Ubuntu 18.04 VM 上に K3s による Kubernetes クラスタを用意した．Docker は Docker Hub からイメージをダウンロードすることが容易なため採用した．ユーザーはブラウザ (e.g. Google Chrome) を用いてアクセスする．

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．

- Kubernetes を意識することなく環境を構築できるか
- お勧めされた環境は目的と合致したものか

評価では本提案がどれだけ課題を解決できたかをアンケートによって集計し計測する．

5.1 実施方法

評価は研究室のメンバーを対象で 8 人に行った．評価は以下の流れで行った．

(1) 本提案のソフトウェアを実際に触ってもらう
(2) 本提案に対するアンケートを受けてもらう
(3) アンケートを集計して分析を行う

5.2 分析

Q1「Kubernetes を意識しないで環境を構築できたか？」この質問は課題で挙げた「構成ファイルを書く必要がある」「Kubernetes を用いた環境構築をするのであれば構造を理解する必要がある」この 2 つの課題を解決できたかを測る指標として聞いた．結果は，アンケートに協力してくれた方全員が意識しないで環境を構築できたと回答した．本提案によって 2 つの課題は解決できたといえる．

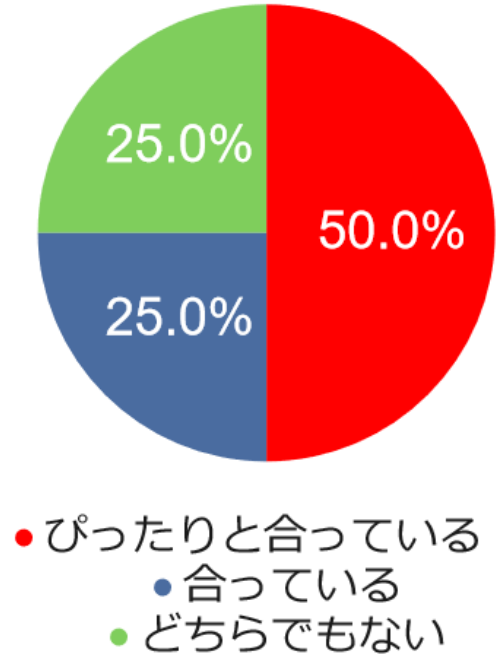

図 9 Q2「目的と合った環境を構築できたか？」

Q2「目的と合った環境を構築できたか？」この質問は課題で挙げた「Kubernetes は使用できるアプリケーションの種類を限定しないため選択肢が多く，使いたい環境に合ったソフトウェアを定められない」ことを解決できたかどうかの指標として聞いた．結果は半数がぴったり合っていると回答したものの，合っているが 25%，どちらともいえないも 25%だった．これは実験のために実装した環境は 3 種類しかなく，バリエーションが少ないために目的に合った環境が構築できていないのではないかと推測できる．

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．本提案の評価はアンケートで行った．

総務省労働力調査より 2020 年の日本の情報通信業の雇用者数は 228 万人である．母集団が 100 万人以上になると必要サンプル数が横ばいとなり，約 384 人アンケートをとれば必要サンプル数を満たすことになる [10]．しかし，アンケートをとる必要人数 384 人は現状で届き得ない数字である．そのため，384 人以上取る方法が必要となる．

本提案の対象者はエンジニアとしている．従って，一般の人へ無作為にアンケートを取る方法は不適切である．エンジニアを対象にアンケートを取る方法は技術展といった技術的な展示イベントで公開することである．この方法のメリットとして自分自身の伝手では集めることのできない対象者へアンケートを取ることができる．自分自身の伝手と技術展での公開によって 384 人以上取ることは可能である．

7. おわりに

Kubernetes を用いたアプリケーションの実行環境を構築するとき，Pod，Service，Deployment といった Kubernetes におけるオブジェクトを理解することと，構成ファイルの記述方法を知る必要がある．この 2 つを理解しなければ Kubernetes の環境構築ができないことは課題である．本研究では，Kubernetes の構成ファイルを記述する工程を自動化する手法を提案する．手法としては，ユーザへ質問を問いかけ，ユーザが使用したいアプリケーションの利用目的を明確にする．最後に，利用目的に最も近いアプリケーションの環境を提供する．本提案に対してアンケートによる評価を行った．Q1「Kubernetes を意識しないで環境を構築できたか？」の質問に対して 100%意識しないで環境を構築できたと回答した．Q2「目的と合った環境を構築できたか？」の質問に対して 50%がぴったり合っていると回答したものの，合っているが 25%，どちらともいえないも 25%だった．これは実験のために用意した環境が 3 種類しかないため，目的に合った環境を提供できたとは言えない結果になったが，Kubernetes を理解していなくても環境構築ができる手法であると分かった．

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] Marathe, N., Gandhi, A. and Shah, J. M.: Docker Swarm and Kubernetes in Cloud Computing Environment, *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, pp. 179–184 (online), DOI: 10.1109/ICOEI.2019.8862654 (2019).

[3] Abdelbaky, M., Diaz-Montes, J., Parashar, M., Unuvar, M. and Steinder, M.: Docker Containers across Multiple Clouds and Data Centers, *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, pp. 368–371 (online), DOI: 10.1109/UCC.2015.58 (2015).

[4] Shah, J. and Dubaria, D.: Building Modern Clouds: Using Docker, Kubernetes Google Cloud Platform, *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0184–0189 (online), DOI: 10.1109/CCWC.2019.8666479 (2019).

[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).

[8] Wang, Y. and Ma, D.: Developing a Process in Architecting Microservice Infrastructure with Docker, Kubernetes, and Istio (2019).

[9] Orzechowski, M., Balis, B., Pawlik, K., Pawlik, M. and Malawski, M.: Transparent Deployment of Scientific Workflows across Clouds - Kubernetes Approach, *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, pp. 9–10 (online), DOI: 10.1109/UCC-Companion.2018.00020 (2018).

[10] Krejcie, R. V. and Morgan, D. W.: Determining sample size for research activities, *Educational and psychological measurement*, Vol. 30, No. 3, pp. 607–610 (1970).