情報工学科演習用のコンテナ技術を用いた新規サービスの設計・実装

宮平 賢[2,a)]　河野 真治[2,b)]

概要：IT 技術を学ぶ時の学習環境の 1 つとして, OS 上の隔離された環境を構築する技術であるコンテナがある．これらはローカルに設置された計算機, あるいはクラウド上に作られる．作成されるコンテナは学生, あるいは教員側から適切に管理するシステムが必要となる．管理システムはマルチユーザで動作するのは当然として, 利用者や管理者に適した UI, sudo 権限で動作するコンテナへの対処などが含まれる．学生の演習には, Web サービスの実装や人工知能の学習などがある．そのため, 気軽に開発環境やテスト環境などを用意できる利用のしやすさが重要である．本稿ではコンテナ管理ソフトウェアである Docker, Singularity を用いた新規 Web サービスの設計・実装を行う．

1. はじめに

情報通信技術の普及に伴い学生が学ぶ学習環境が必要となる。その学習環境として VM や コンテナにより, 手軽に開発し試せる技術が普及している。だが, 手元の PC 上で VM や コンテナを立ち上げ, 開発を行うことはできるが, VM や コンテナの使用には高性能 PC や 有料のクラウドサービスが必要になる場合がある。この大きな負担を学生に負わせない仕組みが必要である。

琉球大学工学部工学科知能情報コースでは希望の学生に学科のブレードサーバから仮想環境を貸出すサービスを行なっている。貸出をする VM のデフォルトのスペックでは不足の場合, 要望に応じてスペックの変更を行なっている。だが, 貸出サービスでは GPU を利用した処理環境を提供することができない。GPU が搭載されている PC は研究室によっては用意されているが, 研究室に所属していない学生は利用することができない。そのため, 新たな仕組みが必要である。

学科のブレードサーバに搭載される GPU は VM の貸出サービスでは利用することができない。そこでコンテナ技術を利用する。コンテナ管理ソフトウェアである Docker では NVIDIA Container Toolkit である nvidia-docker を利用することで, 複数のコンテナで GPU を共有することができる。Docker は基本的に root 権限で動作する。また一般ユーザが docker コマンドを使用するには docker グループに追加する必要がある。そのため Docker をマルチユーザ環境で使用すると, 他ユーザのコンテナへアクセスができるなどセキュリティの問題がある。

そこで, 本論文では, Docker と マルチユーザ環境で利用しやすい Linux コンテナである Singularity を利用したコンテナ貸出サービスを提案する。このコンテナ貸出サービスでは, Web コンソールからコンテナの操作を行うことで他ユーザのコンテナへの操作をさせない。また, 本コースの類似サービスの課題でもあったデータの永続化を Singularity で, 外部リポジトリの利用を Docker の操作を HTTP API で提供することで解消する。

2. 本コースの類似サービス

本サービスに類似したサービスとして, Docker をラップし複数のユーザで利用することを目的とした ie-docker [1], Kubernetes を利用した教育用コンテナ貸出を目的とした, digdog [2] がある。

2.1 ie-docker

ie-docker とは Docker をラップし複数のユーザで利用することのできるコンテナ管理ツールである。利用する学生は ssh でブレードサーバへ接続し, ie-docker を使用してコンテナを操作することができる。ie-docker は UID 及び GID 情報を取得し他のユーザのコンテナを操作させない。またユーザが使える docker の機能を制限する。表 1 が ie-docker で利用できる機能である。

[1] 琉球大学大学院理工学研究科情報工学専攻
[2] 琉球大学工学部工学科知能情報コース
[a)] mk@cr.ie.u-ryukyu.ac.jp
[b)] kono@ie.u-ryukyu.ac.jp

表 1: ie-docker のコマンド

ps	起動中のコンテナの一覧を表示する
run	コンテナを作成する
start	コンテナを起動する
stop	コンテナを停止する
attach	起動しているコンテナに attach する
cp	コンテナにファイルを送信する
rm	コンテナを削除する

2.2 digdog

digdog とは Kubernetes を利用したコンテナ貸出サービスである。学生は Dockerfile を GitLab CI/CD を利用して GitLab Registry に Docker イメージを登録する。学科アカウントを使用して Web サービスへログインし, 登録した Docker イメージでコンテナを作成することができる。コンテナ作成時は digdog が Kubernetes に Deployment を設定する。Deployment は学生のアカウント名で作成された Namespace に設定される。Namespace は RBAC を用いたリソース操作のアクセス制御が設定されている。そのため学生は Kubernetes コマンドである kubectl コマンドで 手元の PC から Pod の操作を行うことができる。RBAC で許可されているリソース操作は表 2 である。

表 2: kubectl のコマンド

get	Pod の一覧を表示する
log	Pod の Log を表示する
exec	Pod にアクセスする

3. サービスの設計

学生が学習環境を利用する流れを図 1 に示し, 概要を以下で説明する。

3.1 利用技術

サービスではコンテナ貸出を行う。そこで, コンテナ管理ソフトウェアである Docker, コンテナオーケストレーションソフトである Kubernetes, マルチユーザ環境に適した Linux コンテナである Singularity を利用する。

サービスは Docker や Kubernetes のみで提供することもできる。だが, コンテナ内のデータの永続化が問題となる。そのため Singularity を利用する。Singularity では デフォルトで $HOME, /tmp, /proc, /sys, /dev がコンテナにマウントされる。そのため, コンテナのデータの永続化や大量のデータを扱う場合に適している。

3.2 コンテナの作成

学生は学科アカウントで Web コンソールへログインする。Web コンソールでは 学生のコンテナ一覧や Docker イメージ一覧を確認することができる。コンテナ作成を選択するとコンテナを作成するために必要な情報を入力する。入力する内容は表 3 である。コンテナ名には学生のアカウント名が補完されるため, 他の学生と被ることはない。Docker イメージは Docker Hub に登録されているイメージや, 作成したイメージを入力することができる。環境変数とゲストポートはスペース区切りで複数入力することができる。ホストポートは, エフェメラルポート の範囲から設定される。学生は設定されたホストポートを使用してコンテナのサービスへアクセスする。また, 学生はコンテナに対して Web コンソールから, または手元の PC から操作することができる。必要なくなったコンテナは Web コンソールのコンテナ一覧から削除することができる。

表 3: コンテナ作成時の入力内容

ContainerName	コンテナ名
Image	Docker イメージ
Environments	コンテナ作成時の環境変数
GuestPort	コンテナが使用するポート番号

3.3 イメージの作成

Docker イメージの作成は学科で利用している GitLab の CI/CD 機能を使用する。学生は学科 GitLab から CI/CD トークンを取得し, Web コンソールで取得したトークンをセットする。この時 Docker 側に GitLab Runner [3] の立ち上げを依頼する。トークンの設定後, Web コンソールから CI/CD 用の Yaml ファイルをダウンロードし Dockerfile と一緒に学科 GitLab のリポジトリに Push する。Docker イメージの Build が成功すると Web コンソールのイメージ一覧で確認ができる。作成した Docker イメージは編集からイメージの使い方を記述でき, 他の学生に共有するか設定を行える。必要なくなったイメージは Web コンソールのイメージ一覧から削除することができる。

3.4 Singularity の利用

Singularity は Docker イメージを Singularity 用に Build することで, Docker イメージを使用することができる。だが, イメージの Build には sudo 権限が必要となる。Docker イメージの Build を申請性にすると, 管理者の仕事が増え, 学生も利用しづらい。また, Singularity はユーザ権限で動作するため, 学生が ssh でブレードサーバへ接続し利用する方が適している。そこで, Web コンソールから Singularity 用のイメージをダウンロードできる仕様とする。

学生は利用したいイメージをダウンロードし, ブレードサーバへ送信して Singularity を使用する。Singularity を利用する流れを図 2 に示す。

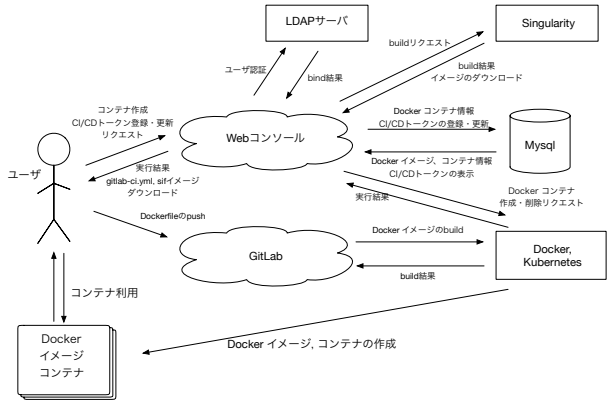

図 1: システム構成

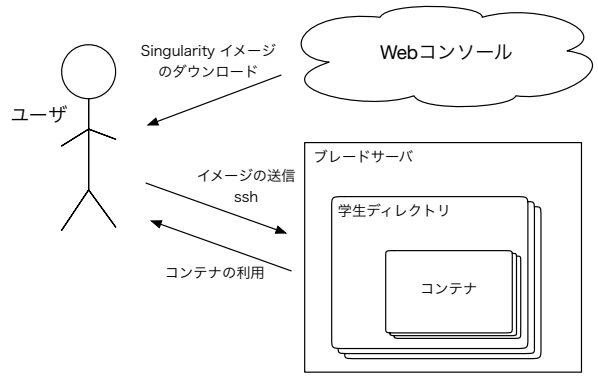

図 2: Singularity の利用

4. サービスの実装

本コースでは学科システムを教員の指導の下, 学生主体でシステム管理チームと呼ばれる組織によって構築・運用・管理が行われている。学科システムはブレードサーバを 4 台, SAN 用ストレージと汎用ストレージをそれぞれ 2 台ずつ導入している。本コースの基幹サービスはこのブレードサーバの仮想環境上で VM として動作している。新たにサービスを実装するとなると, システム管理チームが運用・管理を行いやすい実装にする必要がある。

Web コンソールや Docker の操作を 1 つにまとめると, Docker コンテナの作成が 1 台のブレードサーバのみになってしまう。そこで, コンテナ貸出システムは, 機能ごとに以下の 3 つにサービスに分ける。Docker や Kubernetes の操作を HTTP API で提供することで, 図 3 のようにリクエスト先の変更で複数のブレードサーバにコンテナを分散することができる。だが, 現時点では未実装である。

実装には Docker や Kubernetes の実装言語であり, 操作するためのライブラリが揃っている Go 言語を使用する。

- Web コンソール
- Docker の操作
- Kubernetes の操作

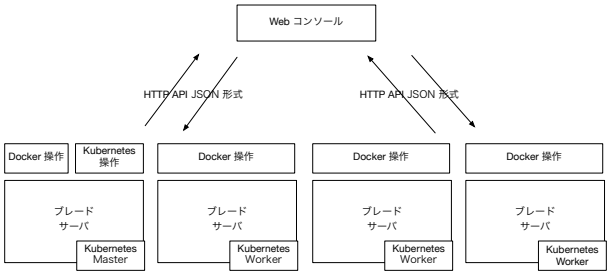

図 3: 機能の分散

4.1 Web コンソール

Web コンソールは本コースの学生や教員が利用するため, 学科アカウントでログインできる必要がある。学科の LDAP サーバを利用して学科アカウントで LDAP 認証を実装する。

Docker の操作や Kubernetes の操作を行う HTTP API はセッション管理を行わないため, Web コンソールで管理する必要がある。そのため, ユーザのコンテナやイメージの情報をデータベースに格納して管理する。ユーザが作成する Docker イメージの情報を取得しユーザのアカウント ID と紐付けを行う。また, 作成した Docker イメージは共有することができ, 共有されたイメージはユーザのイメージ一覧とは別の一覧で確認することができる。ユーザはコンテナ作成時にイメージを入力することができる。この時, 他のユーザの作成したイメージの場合, そのイメージが共有されたイメージなのか確認を行うことで, 非共有に設定されたイメージではコンテナの作成はできない。コンテナの操作を行う時, コンテナに紐づけられたアカウント ID との確認が行われることで, 他のユーザのコンテナを操作することはできない。同様にイメージの削除を行う時にもアカウント ID の確認が行われる。

4.2 Docker の操作

Docker は Docker Engine API を提供している。Docker デーモンは指定した IP アドレスと ポート を リッスンする。 IP アドレスと ポートの指定を行うことで外部から Docker の操作が可能になる。だが, Dockr デーモンが稼働しているホスト上の root アクセスを得られるため, 推奨されてない。また, 本論文で実装するサービスでは Docker のすべての操作を必要としない。そこで, Docker の操作を行うための SDK [4] を使用し, 必要な機能のみを実装する。

サービスを提供する上で Docker の必要となる操作は以下である。

- コンテナの作成
- コンテナの削除
- コンテナでのコマンド実行
- コンテナへファイル送信
- イメージ一覧の取得
- イメージの削除

コンテナは, 表 3 で入力した情報を下に作成を行う。コンテナ名は Web コンソールからリクエストを送るタイミングで補完される。また, コンテナが属するネットワーク名も補完される。リクエストは JSON 形式で受け, JSON 形式でレスポンスを返す。リクエストからコンテナを作成後, 作成したコンテナ ID や ネットワーク ID , コンテナのステータスを返却する。返却したコンテナ ID や ネットワーク ID を下にコンテナ削除やコマンドの実行, ファイルの送信を処理する。だが, ファイルの送信では JSON 形式ではなく multipart/form-data 形式でリクエストを受ける。

Docker イメージは GitLab CI/CD を利用して作成するが, Build が成功したかを判断することはできない。そのため, Web コンソール側から 5 分に一度イメージの更新リクエストが送られ, Docker イメージの一覧をリストにまとめ返却を行う。

ユーザが作成するコンテナとは別に GitLab CI/CD で Docker イメージを Build するための GitLab Runner を立てる必要がある。立ち上げはユーザが Web コンソールで CI/CD トークンの設定時に行われる。GitLab Runner をユーザごとに立ち上げることで, 複数のユーザが同時に Build を行うことができる。

4.3 Kubernetes の操作

実装には Kubernetes の操作を行うためのライブラリである client-go [5] を使用する。

5. サービスの評価

6. 類似サービスとの評価

7. 今後の課題

8. まとめ

参考文献

[1] 平良太貴 : IT 技術学習のための教育用計算機システムの研究 (2015).
[2] 秋田海人，高瀬大空，上地悠斗，長田智和，谷口祐治 : 情報系学科における教育研究情報システムの運用管理並びに新規システムの構築に関する取り組み，インターネットと運用技術シンポジウム (2019).
[3] GitLab Runner Docs: *https://docs.gitlab.com/runner/*.
[4] Docker Engine API: *https://docs.docker.com/engine/api/*.
[5] Go clients for talking to a kubernetes cluster: *https://github.com/kubernetes/client-go*.
[6] Singularity: *https://sylabs.io/singularity/*.