情報工学科演習用のコンテナ技術を用いたサービスの設計・実装

Design and Implementation of service using container technology for Information science exercise

175733E 氏名 宮平 賢 指導教員：河野 真治

令和 2 年 9 月 14 日

Abstract

With the spread of information technology, there is a need for a learning environment in which information students can do assignments and exercises. There are VMs and containers as a way to provide a learning environment for students. The University of the Ryukyus has a VM rental service. Some assignments and exercises require GPUs rather than CPUs. However, the VM rental service in this course does not support GPU sharing. Therefore, we provide a learning environment that includes GPUs by using Docker, Kubernetes, and Singularity, which can utilize container virtualization technologies. In this paper, we design and implement a service that provides a learning environment.

1 コンテナ技術を用いた学習環境の提供

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。

本コースでは希望する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 VM スペックは CPU 1 コア，メモリ 1GB，ストレージ 10GB である。このスペックで不足する場合，要望に応じてスペックの変更を行なっている。だが，貸出 VM でも課題によっては処理に時間がかかることがある。例として，人工知能の講義において課される課題においては CPU より GPU が必要となる場合がある。しかし，現在の VM 貸出サービスでは GPU の共有に対応していない。VM 上で GPU を共有するには PCI パススルーを利用することで可能だが，PCI パススルーでは複数の VM に共有することができない。そこで，GPU を含めた学習環境をコンテナ技術を用いて提供するサービスの設計・実装を行う。

2 技術概要

本研究で使用したコンテナ仮想化技術，また本コースで利用しているサービスについての概要を説明する。

2.1 Docker

Docker[1] とは OS レベルの仮想化技術を利用して，ソフトウェアをコンテナと呼ばれるパッケージで提供する。またコンテナの実行だけでなく，コンテナの実行に用いるイメージの作成，イメージを共有する仕組みを持つコンテナ管理ソフトウェアである。コンテナの実行には Docker 社が提供している Docker Hub[2] に登録されているイメージ，Dockerfile を用いて作成したイメージを利用することができる。Dockerfile を用いることで，必要なソフトウェアや各種設定を含んだイメージを作成できる。

2.2 Kubernetes

Kubernetes[3] とは，アプリケーションのデプロイ，スケーリング，及び管理を用意にするためのコンテナを動的管理するコンテナオーケストレーションである。Kubernetes ではオブジェクトによりクラスターの状態を表現する。オブジェクトはコンテナだけでなく，ネットワークやストレージ，接続ポリシーの望ましい状態を記述できる。

2.3 Singularity

Singularity[4] とは，HPC クラスタ上で複雑なアプリケーションを実行するために開発されたコンテナプラットフォームである。Singularity は マルチユーザに対応しており，コンテナ内での権限は実行ユーザの権限を引き継ぐため，ユーザに特別な権限の設定が必要ない。またデフォルトで，$HOME，/tmp，/proc，/sys，/dev がコンテナにマウントされ，サーバ上の GPU を簡単に利用できる。Singularity のコンテナイメージは Docker Hub に登録されているイメージ，または Dockerfile から作成したイメージを変換することで利用することができる。

2.4 GitLab

GitLab[5] とは バージョン管理システムである Git のリポジトリマネージャである。GitLab は GitHub と違い，オンプレミス環境に構築することができるため，本コースでは GitLab を使用している。本研究では GitLab の統合機能の GitLab CI/CD[6]，また GitLab CI/CD と組み合わせて使用する GitLab Runner[7] を利用する。

GitLab CI/CD は 継続的インテグレーション (CI)・継続的デリバリー (CD) を GitLab から利用することができる。CI では GitLab のコードを定期的または自動的にビルド・テストを行う。CD は CI を拡張した機能であり，ビルドやテストだけでなくリリースの準備も行う。

GitLab Runner とは，ビルドのためのアプリケーションであり，GitLab CI と連携することで別の場所でビルドを動かすことができる。

2.5 digdog

digdog[8] とは Kubernetes を利用し Web コンソールからコンテナを作成することができるコンテナ貸出サービスである。学生は学科アカウントを使用して Web サービスへログインし，登録されている Docker イメージでコンテナを作成することができる。

3 サービスの設計

サービスは本コースの学生や教員が利用する。そのため，ユーザが他のユーザのコンテナの削除などの操作を行えないように制限をするなどの，マルチユーザ環境へ対応する必要がある。また，管理者にコンテナで利用するイメージを用意してもらうのではなく，利用したい学習環境をユーザが構築できる仕組みが必要である。GPU を含む学習環境を提供するために，複数のコンテナへ GPU を共有できる仕組み，またコンテナへのファイルの共有ができる仕組みが必要となる。

3.1 マルチユーザへの対応

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 グループに追加されたユーザは，他ユーザのコンテナを操作できるなどセキュリティ上の問題がある。

そこで，Web コンソールを用いて管理を行う。 Web コンソールには学科のアカウントを用いてログインし，コンテナの作成や操作を可能とする。コンテナ作成は Docker コンテナと Kubernetes コンテナの 2 つから選択することができる。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 1 である。作成時にコンテナ名をユーザのアカウント名で補完されるため，他のユーザと被ることはない。

表 1: 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
GPU	GPU を使用するか

3.2 イメージの作成

Docker イメージの作成は学科で使用している GitLab の CI/CD の CI 機能を利用する。ユーザがイメージを作成する流れを図 1 に示す。ユーザは学科 GitLab から CI トークンを取得し，Web コンソールに取得したトークンをセットする。この時 Docker 側に GitLab Runner 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 用の YAML ファイルをダウンロードし Dockerfile と一緒に学科 GitLab のリポジトリにプッシュする。GitLab にプッシュした Dockerfile が GitLab Runner 上でビルドされる。ビルドの成否は GitLab から確認することができ，作成されたイメージは Web コンソールから確認することができる。GitLab の CI/CD 機能を利用することで，学生に権限を与えることなくイメージの作成を行うことが可能となる。

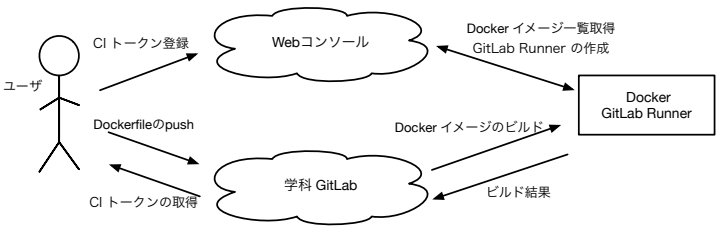

図 1: イメージの作成

3.3 GPU の利用

GPU を使った学習を行うには NVIDIA Container Toolkit である nvidia-docker[9] を利用する。nvidia-docker を導入することで，GPU を利用するための環境が整っている nvidia/cuda イメージを利用することが可能となる。GPU を使った学習環境を利用するには nvidia/cuda でイメージを作成する。作成したイメージをコンテナ作成時の表 1 の Docker イメージに入力する。また GPU を利用するかのチェックを入れることで，コンテナへ GPU を共有することが可能となる。

3.4 ファイルの共有

コンテナに大量のデータを送信する必要がある場合や，データを永続化させたい場合に Singularity を利用する。Singularity はユーザ権限で動作することから，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。Singularity は Docker イメージを変換し使用できる。だが，イメージの変換には sudo 権限が必要となる。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。

ユーザ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。

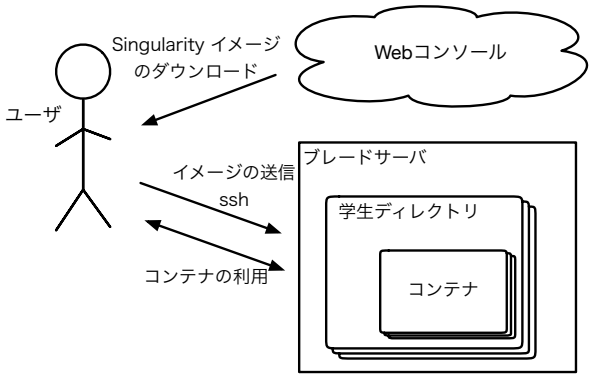

図 2: Singularity の利用

4 サービスの実装

サービスのシステム構成を図 3 に示す。Web コンソールで Docker や Kubernetes の操作をまとめるのではなく，機能ごとに以下の 3 つにサービスを分ける。Web コンソールから HTTP API で各機能へリクエストを送信し操作を行う。

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

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 の操作を行う機能では，ユーザの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。ユーザがコンテナやイメージの操作を行う時は，紐づけられたアカウント ID の確認を行うことで，他のユーザのコンテナやイメージの操作を制限する。

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，本研究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 [10] 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

Web コンソールから JSON 形式でリクエストを受信する。このリクエストを元に上記の操作を行う。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。

4.3 Kubernetes の操作

Docker と同様に Kubernetes のすべての操作を必要としないため，Kubernetes と対話するためのライブラリである client-go [11] 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 Kubernetes 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
- 認証トークンの取得

Docker と同様に Web コンソールから JSON 形式でリクエストを受信し，リクエストを元に上記の操作を行う。Kubernetes ではコンテナへのコマンドの実行やファイルの送信は実装せず，認証のトークンを取得する機能を実装する。Kubernetes では，Kubernetes API の利用権限 Namespace ごとに定義する Role，ユーザやグループに Role を関連付ける RoleBinding がある。この Role と RoleBinding を用いた Role-based access control (RBAC) を利用することで手元の PC からコンテナに対して操作を行うことが可能となる。そのため，RBAC への認証トークンを取得する。RBAC で許可するリソースの操作は表 2 である。

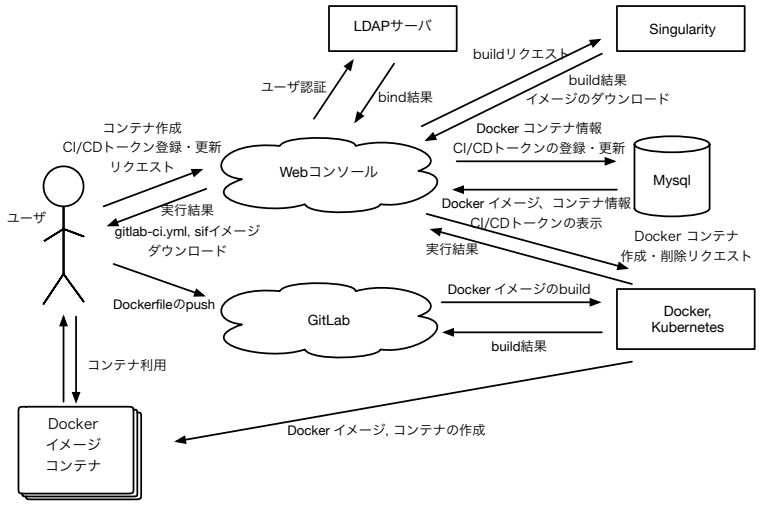

図 3: システム構成

表 2: kubectl のコマンド

get	Pod, Deployment, Service, Ingress 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
cp	Pod にファイルを送信する

5 今後の課題

本研究で実装したサービスでは学生が学習環境として利用するには，まだ必要な実装が不足している。

本サービスでは，大量のデータを用いる時に Singularity を使用できる環境を用意している。だが，Web コンソールから作成した Docker や Kubernetes のコンテナではデータの永続化に対応していないため，コンテナの削除で削除されてしまう。そこで，学科のサーバでは学生ごとのディレクトリにマウントするなどの対策を行う必要がある。

本サービスでは，学生が自由に Docker イメージを作成できる。また，Docker イメージを Singularity 用のイメージに変換する。そのため，イメージの容量でブレードサーバのストレージを圧迫してしまう可能性があることから，あまり利用されていないイメージは定期的に削除する必要がある。

次にネットワークの設定である。作成したコンテナへのアクセスには，コンテナが動作しているサーバの IP アドレス，設定されたポート番号を使用する。そのため，外部からアクセスなどに対応することができない。そこで，コンテナごとに IP アドレスを設定するなどの対策が必要である。

また，本サービスではユーザごとにリソースの制限を行っていないため，過剰なリソースの占有を防ぐための対策をする必要がある。GPU などの負荷がかかるプログラムの実行で使用されるリソースにはジョブ管理ソフトウェアなどで対策をとる。

参考文献

[1] Docker, https://www.docker.com/. 2020/9/11.

[2] Docker Hub, https://hub.docker.com/, 2020/9/11.

[3] Kubernetes, https://kubernetes.io/, 2020/9/11.

[4] Singularity. https://sylabs.io/singularity/, 2020/9/11.

[5] GitLab, https://about.gitlab.com/, 2020/9/11.

[6] GitLab CI/CD, https://docs.gitlab.com/ce/ci/, 2020/9/11.

[7] GitLab Runner Docs, https://docs.gitlab.com/runner/, 2020/9/11.

[8] 秋田 海人 and 高瀬 大空 and 上地 悠斗 and 長田 智和 and 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).

[9] NVIDIA Container Toolkit, https://github.com/NVIDIA/nvidia-docker, 2020/9/11.

[10] Docker Engine API, https://docs.docker.com/engine/api/, 2020/9/11.

[11] Go clients for talking to a kubernetes cluster, https://github.com/kubernetes/client-go, 2020/9/11.