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は CPU 1 コア, メモリ 1GB, ストレージ 10GB である。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。本コースは 2017 年度よりコース制へと移行し, 人工知能やシステム開発などの先端技術を身につける講義や実験が設けられた。これまで講義の演習や実験は学生の PC や 貸出 VM で実行していたが, 課題によってはスペックが足りなく処理に時間がかかることがあった。例として, 人工知能の課題のプログラムの処理には CPU より GPU を用いることで処理時間を早くすることができる。だが, 現在の VM 貸出サービスでは GPU 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, 本コースの学生が学習するための高性能な環境を利用できる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を操作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コンテナプラットフォームである 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外部リポジトリの利用は, Docker の機能を HTTP API で提供することで解消する。

2. 技術概要

本研究で必要な技術概要を述べる。

2.1 Docker

Docker 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

[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

© 2020 Information Processing Society of Japan

メージの作成や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 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 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。本研究では以下のオブジェクトを主に利用する。

- Pod
  - Kubernetes で作成, 管理できる最小単位。Pod 内に 1 つ以上のコンテナを起動できる。
- ReplicaSet
  - 安定した Pod の維持を行い, クラスタで必要な Pod 数を管理する。Pod のセルフヒーリングを行う。
- Deployment
  - ReplicaSet のロールアウトを図るなど, 管理を行う。
- Service
  - Pod にアクセスするための IP アドレスやポートを割り振る。
- Ingress
  - 外部からのアクセスを管理する。負荷分散, SSL 終端, 名前ベースの仮想ホスティングの機能を提供する。
- Role
  - 仮想クラスタとしてグループ化して取り扱える。
- RoleBinding
  - ユーザやグループに Role を関連付ける。

2.3 Singularity

Singularity とは, HCP 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 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 はオンプレミス環境で利用できるため, 本コースでは GitLab を使用している。また, 本研究では GitLab の統合機能の GitLab CI/CD, GitLab CI/CD と組み合わせて使用する GitLab Runner 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。本コースでは, Operating System という講義で Mercurial と Jenkins を利用してコードのテストを行う課題などがある。

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。

3. 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, libvirt の CLI である virsh をラップしマルチユーザ VM 環境を提供する ie-virsh [1] 。また, 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 , 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。

3.1 ie-virsh

ie-virsh とは, 本コースの Operating System という講義に向けに libvirt の CLI である virsh をラップし複数のユーザで利用することができる VM 管理ツールである。ie-virsh は 本コースの講義に向け作成されたが, 学生の演習でも利用ができる。課題では VM の環境を学生が設定し, 情報工学科の持つブレードサーバ上にアップロードし, プログラムの実装や測定を行う。[1] 学生は手元の PC で作成した VM をブレードサーバ上にデプロイすることで, 演習環境を構築することができる。ie-virsh は ユーザの UID 及び GID 情報を取得することで, 他のユーザの VM を操作させない。表 1 は ユーザが利用できる ie-virsh の機能である。

表 1: ie-virsh のコマンド

define	XML の template を下に domain を作成
undefine	define で作成した domain を削除
list	define で作成した domain の一覧表示
start	指定した domain 名の VM を起動
destroy	指定した domain 名の VM を停止
dumpxml	domain の XML を参照

3.2 ie-docker

ie-docker とは Docker をラップし複数のユーザで利用

© 2020 Information Processing Society of Japan

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 情報を取得することで, 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 2 が ie-docker で利用できる機能である。

表 2: 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

3.3 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 を利用してビルドし 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 Role と RoleBinding を用いた, Role-based access control (RBAC) 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 3 である。

表 3: 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

4. サービスの設計

サービスでは本コースの学生や教員がにコンテナ貸出を行う。このコンテナ貸出の構成を図 1 に示し, 概要を以下で説明する。

4.1 コンテナの作成

学生または教員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 ユーザのコンテナ一覧や 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 4 である。コンテナ名にはユーザのアカウント名が補完されるため, 他のユーザと被ることはない。Docker イメージには Docker Hub に登録されているイメージや, ユーザが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。ユーザ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, ユーザ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 4: 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

4.2 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。ユーザは学科 GitLab から CI 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 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。Docker イメージのビルドが成功すると 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の記述や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。

4.3 Singularity の利用

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 Docker イメージを変換し使用できるが, イメージの変換には sudo 権限が必要となる。Docker イメージの変換を申請性にすると, 管理者の仕事が増え, またユーザも利用しづらい。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。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。

© 2020 Information Processing Society of Japan

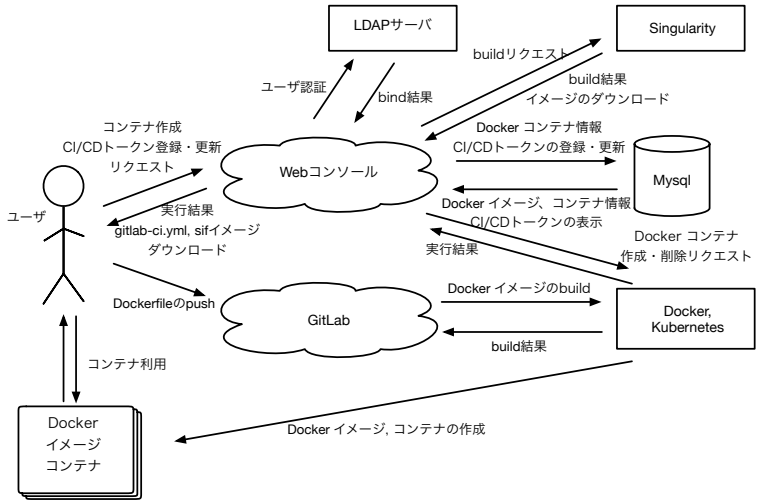

図 1: システム構成

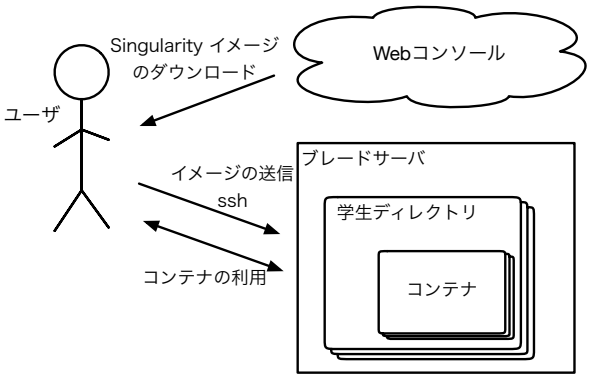

図 2: Singularity の利用

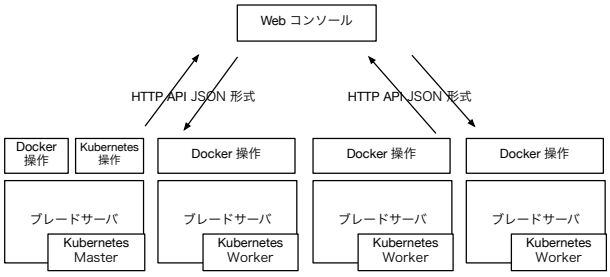

図 3: 機能の分散

5. 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

5.1 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。

© 2020 Information Processing Society of Japan

5.2 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 [3] 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 4 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を受け, 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。

5.3 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 [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 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

Kubernetes でのコンテナ作成は Pod を作成することである。 Kubernetes でのコンテナ作成は Namespace, Deployment, Service, Ingress の流れでオブジェクトを作成する。コンテナの作成は Docker と同様に表 4 の情報を下に作成する。作成するそれぞれのオブジェクト名は Web コンソールで コンテナ名とアカウント ID で補完される。また, Namespace はアカウント ID となる。コンテナの削除にはそれぞれのオブジェクト名と Namespace を用いる。

Kubernetes で作成したコンテナは Web コンソールから操作できないため, digdog でも利用されている RBAC を用いる。RBAC で使用する 認証トークンはユーザごとに作成された Namespace に設定されるトークンを返すことで, 他のユーザが認証することはできない。またアクセス制御は Namespace ごとに設定されることで, 他のユーザのコンテナを操作することはできない。RBAC で許可するリソースの操作は表 5 である。

表 5: 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

6. 他のサービスとの比較

今回作成した Web サービスは主に学生の学習環境をコンテナ技術を利用して提供する。そこで, これまで本コースで使用されてきたサービスと, 近年普及しているクラウドサービスと比較する。

6.1 ie-virsh

ie-virsh は手元の PC で作成した VM を学科のブレードサーバにデプロイできるサービスである。VM は OS の仮想化環境を提供するため, ユーザが好みの環境を構築できるなど自由度が高い。

本研究で実装したサービスでは, Docker イメージで構築されたアプリケーションに限定される。また, ユーザが欲しい環境は Docker イメージを作成しなければいけないため, Docker について学習する必要がある。だが, VM と違い気軽に環境の構築やテストを行える。また Docker イメージを共有することで, 自身と同じ環境を他のユーザに利用してもらえるなどの良さがある。

6.2 ie-docker

ie-docker は Docker をラップしたツールであり, ユーザは学科のブレードサーバへ ssh で接続を行い CUI から利用することができる。表 2 の機能でコンテナを操作するこ

© 2020 Information Processing Society of Japan

とができる。だが, ie-docker ではユーザがコンテナで使用するイメージを管理者が用意する必要がある。

本研究で実装したサービスでは, コンテナで使用するイメージは Docker Hub に登録されているイメージ, または作成したイメージを利用することができる。また, ユーザが Docker イメージを作成できることから管理者の負担が少なくなると考える。

6.3 digdog

digdog は Kubernetes を利用したコンテナ貸出サービスである。コンテナ作成時に選択できるイメージはユーザが作成する必要があり, Docker Hub に登録されているイメージを選択することができなかった。

本研究で実装したサービスでは, コンテナで使用するイメージは Docker Hub に登録されているイメージ, または作成したイメージを利用することができる。また Kubernetes でのコンテナ貸出だけでなく, Docker でのコンテナ貸出を行うことができる。そのため, Kubernetes 全体が停止したとしてもブレードサーバの Docker のみでサービスを提供することができる。

6.4 クラウドサービス

近年様々なクラウドサービスが普及し手元の PC から高性能なクラウド環境を利用することができる。だが, クラウドサービスは無料から有料までであり。無料では時間制限や容量制限など様々な制限がある。また有料だと気軽に利用しづらいなどの問題もある。そのため, 本サービスはオンプレミス環境で実装を行った。オンプレミス環境を利用することで制限がなく, サービスで使用するデータを自身で管理できるなどの良さがある。だが, クラウドサービスと違い利用ドキュメントが無いため, 初めてでも利用しやすいよう改善や機能の追加をしていく必要がある。

7. 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が対応していないため, コンテナの削除時に共に削除されてしまう。学科のブレードサーバでは学生用ディレクトリがある。Docker ではそのディレクトリをコンテナ立ち上げ時にマウントすることで, コンテナ内のデータの永続化に対応できる。また, Kubernetes では Peresistent Volumes という永続ボリュームの仕組みがある。この Peresistent Volumes をユーザごとに管理することで, コンテナのデータの永続化に対応できる。このような対策でコンテナでデータの永続化に対応できるが, コンテナごとにデータの保存場所が違うなどの問題があるため, データを管理する仕組みが必要だと考える。

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, 定期的にイメージを削除する必要がある。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。

本サービスは Docker Hub に登録されている Dokcer イメージを利用できるが, Docker Hub は誰でもイメージを登録することができる。そのため, Docker Hub に登録されているイメージにマルウェアが仕込まれている可能性がある。イメージの取得時にスキャンを行うなど, セキュリティ対策を考える必要がある。

8. まとめ

本稿では, 本コースで利用する新規サービスの設計と実装を行った。

参考文献

[1] 平良太貴，河野真治：OS 授業向けマルチユーザ VM 環境の構築，研究報告システムソフトウェアとオペレーティング・システム (OS) (2014).
[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] Docker Engine API: *https://docs.docker.com/engine/api/*.
[4] Go clients for talking to a kubernetes cluster: *https://github.com/kubernetes/client-go*.
[5] Singularity: *https://sylabs.io/singularity/*.
[6] GitLab Runner Docs: *https://docs.gitlab.com/runner/*.

© 2020 Information Processing Society of Japan