共用計算サーバにおけるユーザ作成コンテナの実行を目的としたコンテナ作成ワークフローと実行環境の試作

宮下 夏苗 [1)], 間藤 真人 [1)], 本郷 研太 [1)], 井口 寧 [1)]

1) 北陸先端科学技術大学院大学 情報社会基盤研究センター

Container building workflow and testing environment for users aimed to run on shared computing servers

Kanae Miyashita[1)], Masato Mato[1)], Kenta Hongo[1)], Yasushi Inoguchi[1)]

1) Research Center for Advanced Computing Infrastructure, Japan Advanced Institute of Science and Technology.

概要

学生，研究者が研究・実験用リソースとして学内共用の計算サーバを利用する際，通常は一般ユーザ権限でサーバにログインし，利用したいプログラムやライブラリの不足分を自分のホームディレクトリにソースコンパイルして作りこむことになる．しかしコンパイルを伴う計算実行環境の準備は共用サーバの環境や利用者のノウハウにも大きく依存し，一般の利用者には必ずしも簡単な作業とはいえない．本学では利用者がコンテナを用いて計算実行環境を管理者権限のもとに作成し，これを共用計算サーバでのジョブ実行時に呼び出して利用することを目標としたコンテナ作成からジョブ実行までのワークフローを構成し，実験的に実行環境を試作した．本稿ではワークフローの構成と現在の課題について報告する．

1 はじめに

大学の研究者, 学生に共用計算サーバを提供するにあたり, 利用者側から見た利便性が多分に損なわれる要因に，個々の利用者が管理者権限を与えられない問題がある．

個々の利用者が利用したいライブラリやソフトウェアをシステムの管理者権限なくインストールするには, 権限のあるディレクトリにソースからコンパイルする方法が主となる．しかし yum や apt のようなパッケージ管理ツールを利用しない一般ユーザ権限でのインストール方法に関する情報は, 程度の差はあれど豊富とはいえない．また, コンパイル作業は基盤となる共用サーバ環境にも依存するため不足するライブラリまで自前で用意することにもなりかねず, コンパイルとこれに伴うトラブルシューティングに時間を割かれる状況は, 研究のためにシステムを利用したい利用者に無用な労力を要求する一因に他ならない．

このほか，サーバのアップデートやリプレースによってライブラリや OS など基本となる環境が変わり, 研究に用いた環境の保持と後日の再現が困難なことも利便性を減じる点として挙げられる．文部科学省のガイドライン [1] によれば,「一定期間研究データを保存し、必要な場合に開示することを義務付ける」べきであるとされ, 国立情報学研究所の資料では保存対象となるデータとしてアプリケーションソフトが一例に挙げられている．研究に用いられたソフトの保存を検討する場合に環境ごと保存し必要に応じて再現できることは，不正対策の観点からも有効であると思われる．

本学ではこのような課題の解決に向けてコンテナの導入を試みた．

利用者がコンテナに対して管理権限をもって必要なソフトウェアをインストールし, 作成したコンテナを共用計算サーバ上の計算実行時に呼び出して実行するまでのワークフローを作成し，このうち，共用計算サーバの計算実行時にコンテナを呼び出し，計算に利用する部分についてはすでに利用者に公開し，実運用に入っている．しかし現時点で利用できるコンテナは管理者が事前に準備したもののみであるため，利用者が自由にコンテナを作成し，利用できる環境を構築し，公開することが最終的な目標である．

2 ワークフローの概要

2.1 全体

ワークフロー全体の構成を図 1 に示す．

ユーザの利用手順は以下となる．

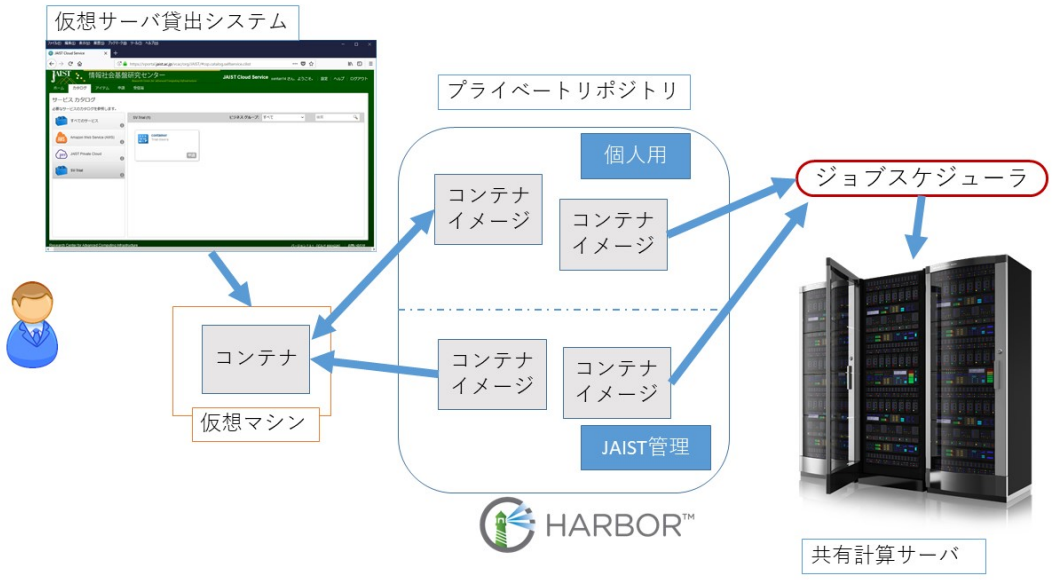

図 1　全体構成

1. Web ポータルより個人用コンテナホスト作成
2. 仮想マシン内でのコンテナ操作（イメージ呼び出し, 起動, カスタマイズ, アップロード等）
3. 学内プライベートレジストリに作成イメージのアップロード
4. 共用計算サーバでのジョブ実行時におけるイメージ呼び出し

構成においては UNIX 環境に慣れていない利用者がコンテナの概念や利用法についての知識を要求されることなく, 可能な限り直感的に利用できる環境を目標とした.

2.2　コンテナホスト作成

利用者が利用したいソフトウェアをインストールしたコンテナを作成するために, コンテナイメージの操作，レジストリからのイメージの読み出しとアップロードができるコンテナホストが必要になる.

ここで，利用者向けのコンテナホストの準備にあたり, Docker Security[2] に記載のあるように，利用者にコンテナを操作させるためのデーモンに「現時点で root の特権が必要である」とされるセキュリティリスクがある．また, コンテナホストを複数人で使いまわすことで，方法によってはレジストリへのログイン認証トークンが環境上に残り, 他人の権限でレジストリにアクセスされるような問題も想定される.

これらの問題を回避するため, 試作の環境ではユーザごとに軽量な Linux ベースの仮想マシンを作成し, 各利用者が個々に分離された個人用の仮想マシン上でコンテナを操作するものとした.

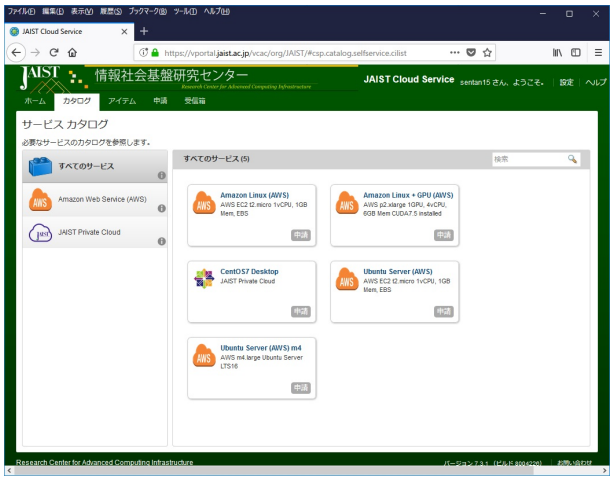

図 2　仮想マシン作成ポータル (VRA)

コンテナを作成，レジストリにアップロードするための仮想マシンを利用者に提供するインタフェースとして, 本学で導入している VMWare vRealize Automation(以下 VRA)(図 2) を利用した．利用者は VRA の Web ポータルから事前に管理者がカスタマイズした仮想マシンのテンプレートを選択することで, 自由に仮想マシンのクローンを作成し, ネットワーク設定などの必要な初期設定を自動的に完了した仮想マシンを入手できる.

今回, このポータル上に個人用コンテナホスト向けにセットアップした Linux テンプレートを作成し, 各利用者がコンテナを操作するための仮想マシンを作成

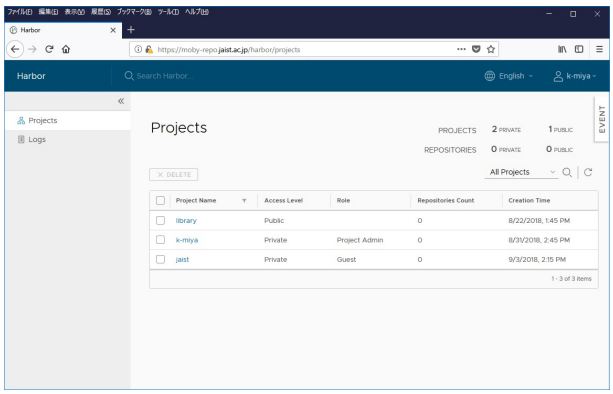

図 3　学内プライベートレジストリ (Harbor)

できるようにした.

2.3　コンテナ操作

コンテナを扱うためのコマンドはあまり複雑というわけではないが, 利用者の主目的は共用計算サーバでの計算実行であるため, あえてコンテナの概念とコマンドの習得を必須とさせる理由はない. そこで対話型の簡易なスクリプトにより, 初心者がより簡易にコンテナの操作ができるように試みた.

前項に示した個人用コンテナホストにあらかじめスクリプトを入れておき, 実行すると利用したいイメージ，タグ名を対話的に問い合わせる，イメージを起動して attach するなどの処理を行う．もちろん，Docker コンテナに慣れた利用者であれば，コンテナホスト上から一般的な方法で docker コマンドの実行, dockerfile を使用したコンテナの作成もできる.

2.4　学内プライベートレジストリ

オープンソースの Harbor を用いて学内用のプライベートレジストリを作成した.(図 3) レジストリのユーザ認証には学内の LDAP を用いる.

コンテナイメージはレジストリ内に,「プロジェクト」という単位に分けて格納する．各プロジェクトごとにプロジェクト所有者, 開発者, ゲストの三通りの権限を付与できる.

試作した環境では管理者が提供する基本的なイメージを格納する jaist プロジェクトおよび, その他各利用者が作ったイメージを格納する個々のユーザ名のプロジェクトを作成することにした.

一般の利用者に対し, jaist プロジェクトにはゲスト権限のみ付与し, イメージの読み込みと利用のみ可能とする．ユーザ名を冠したプロジェクトには各利用者ごとの所有者権限を与え, イメージの書き込み, 読み出しのほか, プロジェクトへのメンバー追加も可能にした．このことで, 研究室などでコンテナイメージを共有するなどの操作も可能となる．また, 所有者が意図しない他人はイメージを利用することはもちろん, 視認することもできない.

プロジェクトへの権限付与など, 管理者権限が必要な操作には Harbor の API を利用し利用者がシステムの利用を開始する際に, バックエンドでプロジェクト作成と権限追加を行う.

ここで, 利用者が自由にコンテナを作成するうえで, 外部から持ち込んだプログラムやライブラリを組み込む際に悪意のあるプログラムや脆弱性を含んだライブラリが取り込まれてしまう可能性も想定される.

Harbor は基本機能としてコンテナの脆弱性検知ツールである Clair に対応しており，利用者が作成したイメージを格納するにあたって脆弱性のリスクを軽減する効果が期待される.

2.5　共用計算サーバでのコンテナ実行

共用計算サーバではジョブスケジューラとしてオープンソースの OpenPBS を用いている．利用者が実行したい計算内容と必要なリソースをスクリプトにまとめてジョブとして登録しておくことで, リソースが空き次第順番にジョブを実行する.

このジョブの実行前処理として, リソースとしてコンテナが指定された場合にレジストリから指定されたコンテナイメージを読み込み, 起動したコンテナ内でジョブを実行する処理を加えている.

3　課題

3.1　GUI 化

レジストリへのコンテナイメージの入出力やコンテナ起動停止などはコンテナホスト上から簡易スクリプトを通して実行するものとしているが，これらも Web ポータルから GUI で操作できれば，初心者にもより利用しやすい環境になると思われる.

手法として，個人用コンテナホストに対しオープンソースの Admiral(図 4) を追加する方法を検討した. しかしコンテナの起動停止などは Web の UI から実行可能になるが，レジストリと連携しアップロードやイメージ管理を Admiral から実行するにはレジストリの管理者権限が必要になるため，個人用レジストリを立ててバックエンドで共用のレジストリに対してイメージレプリケーションを行うなどの工夫が必要となり，構成が複雑化する問題点がある.

3.2　イメージの肥大化

利用者が作成したコンテナイメージは複数のライブラリやソフトウェアを追加するため，総じて容量が

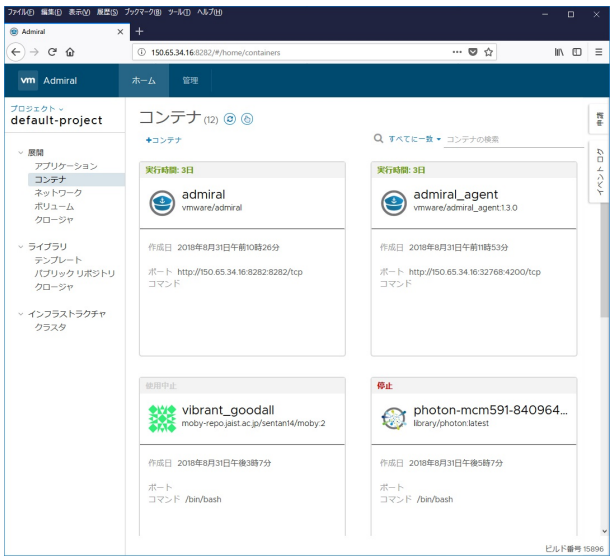

図 4　Admiral

大きくなる．これを毎回ダウンロードして実行するため，計算の実行開始までの所要時間が増加する問題がある．

3.3　計算機上に残るイメージの処理

共用計算サーバでイメージを読み出して実行した後，サーバ内にイメージが残る．このイメージの削除タイミングの検討が課題として挙げられる．

計算サーバ内に前回のイメージが残っていればイメージダウンロードの時間を削減できる．とくに利用者が作成したイメージは容量が大きくなることから，この時間短縮は計算の実行の上で効果的と考えられる．

しかし，利用者が同じイメージ名，同じタグ名でイメージをレジストリにアップロードしてしまった場合などには計算サーバでの実行時にアップロードされた新イメージを読み込むことなく，サーバ内に残っている同名の古いイメージがそのまま読み込まれてしまう問題がある．また利用者のイメージ更新により，利用されなくなったイメージがサーバ内に残ってしまうことにもなる．このことから適切なタイミングでのイメージ削除が必要である．

3.4　リソース管理

大学という機関の性質上，学生の入れ替わりが定期的に発生する．このため学生の修了後などに不要となったリソースの整理と回収が必須となる．

前項の VRA では仮想マシンの貸出時に貸出期間の設定を必須の申請項目とすることができ，貸出期間終了後にはマシンの停止，さらに一定期間後のマシンの削除までを自動化することができるため，必要な期間のみ仮想マシンを利用させ，不要となれば削除するまでの自動化を実現している．

しかしレジストリに残されたコンテナイメージについては，研究環境の保存の観点を含めてデータ整理ルール，所有者へのイメージ還元方法を検討する必要がある．

4　まとめ

構成したワークフローのうち，計算機でのジョブ実行時にコンテナを呼び出す処理については，本学においてすでに実環境で利用者に開放されている．この環境に接続し，利用者がコンテナを用いて必要なプログラム実行環境を管理者権限のもとに作成し，共用計算サーバでのジョブ実行時に利用するためのテスト環境を試作した．細部のポリシーを定めたうえで試行を続け，早期にワークフロー全体の利用者向けの公開を目指している．

参考文献

[1] 文部科学省「研究活動における不正行為への対応等におけるガイドライン」2014.

[2] Docker Security, https://docs.docker.com/engine/security/security/