Vol.2017-IOT-39 No.10
Vol.2017-SPT-25 No.10
2017/9/29

Dockerを用いたコンピュータ演習室向けLinux端末システムの設計

佐藤 悠[1,a)]　萩原 威志[1]

概要：新潟大学工学部情報工学科では，Linux 環境を用いた演習を行うための計算機演習室システムを運用している．システムは主に複数台の演習用サーバと，シンクライアントである X 端末群，ファイルサーバによって構成されているが，このような常時稼働の共用システム下での利用を想定していないアプリケーションも多く，トラブルが頻発してシステム運用上の問題となっていた．そこで，利用者が多いために問題解決もより進んでいると考えられるスタンドアロンのデスクトップ PC に近い形態で実行するため，コンテナ型仮想化ソフトウェアの Docker を利用し，1 ユーザに対し 1 コンテナを実行する環境を提供することで，各ユーザの利用に合わせて起動・終了するシステムを構築した．また併せて，この際に作成した Docker イメージを学生が所有する PC 上で実行可能な形で配布することで，学生が利用できる Linux 演習環境として，新しい在宅学習システムを提案した．

Linux terminal system design for computer room by Docker

Sato Yu[1,a)]　Hagiwara Takeshi[1]

1. はじめに

新潟大学工学部情報工学科では，Linux 環境を用いた演習を行うための計算機演習室システムを運用している．システムは主に複数台の演習用サーバ (Debian Linux)，ユーザのホームディレクトリを格納するファイルサーバ，そしてシンクライアントである X 端末群によって構成されており，ユーザは X 端末を介していずれかの演習用サーバへ接続することで，システムを利用している．また，デスクトップ環境としては XFCE を利用しているが，Debian + XFCE + 日本語環境 + X 端末 となると利用者数のボリュームゾーンを外れるようで，トラブルレスとは言い難い状態が続いていた．しかも，その都度，原因究明・根本対策を行うにはスタッフ不足であり，場当たり的な対応を繰り返すことが多かった．そこで，少しでも利用者数が多く，問題解決も進んでいるであろうスタンドアロンのデスクトップ PC に近づけるべく，常時稼働の共用システムをやめ，端末利用中のみ動かす仮想 PC のシステムへの移行を考慮することにした．

演習環境をスタンドアロンで提供するにあたっては，X 端末ごとに仮想マシンを割り当てる方法がまず考えられるが，仮想マシンを多数作成することはハードウェアリソース的に無駄が多い．そこで，コンテナ型仮想化ソフトウェアの Docker によって作成したコンテナ内でデスクトップ環境を含むスタンドアロンの演習環境の提供を試みた．サーバ上で各ユーザごとに割り当てた Docker コンテナの中で演習環境を実行する仕組みを作ることで，1 ユーザに対し 1 つのサーバを割り当てることが可能で，さらにユーザの利用に合わせて起動・終了するシステムを仮想的に実現した．実行するコンテナはデスクトップ演習環境を起動し，これをユーザの利用している X 端末に転送するほか，演習室内ファイルサーバへのアクセス機能を持つよう起動時に設定を行う．

また，仮想マシンを演習用サーバとして利用する場合は，サーバのシステム設定や，セキュリティアップデート，アプリケーション追加等の変更によって問題が生じた際に，

[1] 新潟大学
Niigata University
[a)] y-sato@cs.ie.niigata-u.ac.jp

© 2017 Information Processing Society of Japan

正常時の仮想マシンスナップショットへのロールバック作業を必要としていたが，本システムでは演習環境を Docker イメージ上に構築するため，問題発生時は正常に作成された過去のイメージを起動するだけで対応可能となる．

さらにこのイメージを学生が所有する PC 上で実行することができれば，学生が在宅時などにも利用できる演習環境として提供することが可能となる．既に演習室システムでは学外からの接続サービスを提供しているが，リモートデスクトップであるため，利用回線品質によって大きく影響を受ける．本システムで提供する在宅学習システムであれば，導入後はオフラインで利用することも可能でありながら，演習室で実行するものと変わらない環境を利用できるという利点があるため，これを在宅学習環境の新しい選択肢として提案した．

2. システム概要

ユーザの演習環境を Docker コンテナで提供するに際し検討する必要がある主な問題として，ユーザ情報を含まないイメージから作成されたコンテナにユーザをどう結びつけるかという点と，揮発性の保存領域であるコンテナ内での作業データをどのように保存するかという点が挙げられる．

本システムと類似した形で大学内のシステムを Docker ベースのシステムで提供した例として，東京工科大学の Jeneau[2] が挙げられる．Jeneau はブラウザ経由で利用するシステムで，ユーザ認証は大学ポータルサイトでのシングルサインオンを利用しており，ログイン時にブラウザにセットされた Secure Cookie を元にして，実行したコンテナとユーザを結びつけている．またコンテナ内作業データの保存については，コンテナ停止後にユーザ作業分の差分レイヤを作成し，Docker Registry に転送することで実現している．本システムでは，演習室内のユーザ認証サーバ上でユーザ管理を行っているため，このサーバでの認証を介して得たユーザ情報を利用し，コンテナと結びつける仕組みを実装した．ファイルの保存については，Jeneau がコンテナの一部をパブリッククラウドへオフロードする都合上，転送量を抑えるためにコンテナ停止後に作成した差分レイヤを一括して転送する必要があったのに対し，本システムでは演習室内のファイルサーバをコンテナ外の保存領域として活用できる．

以上の点を踏まえながら，システム設計について解説する．

2.1 システム設計

本システムの構成を図 1 に示す．本システムはシステム内の管理プロセスを扱う'swarm-master' と，実際に演習用のユーザコンテナを実行する'swarm-node' の 2 種類のサーバから構成される．本システムは Docker Swarm によってクラスタリングしており，swarm-master は 1 つ，swarm-node は複数存在する．

- swarm-master
  Swarm クラスタのマネージャノード．
  - Docker Registry
    swarm-master 内で作成された演習用イメージを swarm-node の Docker Engine と共有する．
  - Swarm Manager
    Docker Swarm クラスタ参加ノードを管理し，起動するコンテナの割り当て等を行う．
  - Display Manager
    X 端末からの接続を受け付け，認証サーバを介してユーザログインを行う．
  - Container Runner
    ユーザログイン後に実行され，実行する演習イメージ選択インタフェースの提供とコンテナの作成を行う．
- swarm-node
  Swarm クラスタのワーカーノードであり，本システムをホストする物理サーバの台数に合わせ，swarm-node は 3 台実行している．
  - User Containers
    ユーザが実際に演習等を行う際に利用する演習用コンテナで，swarm-node を介してファイルサーバ内のホームディレクトリがマウントされる．管理上識別しやすいよう，コンテナ名は実行者のユーザ名に設定している．

また，swarm-master と swarm-node のそれぞれの仕様を表 1 に示す．

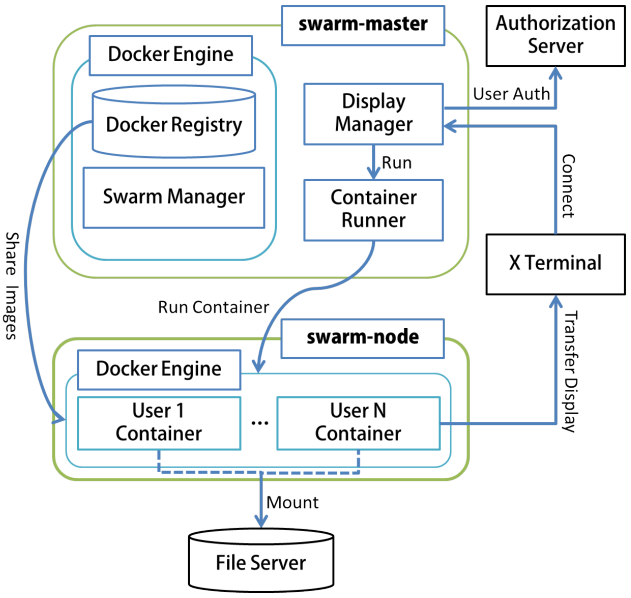

図 1 システム構成図

表 1　仮想サーバ仕様

swarm-master	
CPU	2.70Ghz * 4 コア
メモリ	4GB
OS	Debian GNU/Linux 9.1 x86_64
swarm-node	
CPU	2.70Ghz * 16 コア
メモリ	32GB
OS	Debian GNU/Linux 9.1 x86_64

2.1.1　ユーザ認証

ユーザ認証部分を設計するにあたり，ユーザ認証をコンテナ外で行うか，またはコンテナ内で行うという二通りの方法が考えられる．前者の場合，認証機構を含まないためにイメージをシンプルにできる一方で，認証後のユーザと，内部にユーザ情報を持っていないコンテナを結びつけて利用させる仕組みを作る必要がある．後者の場合，コンテナが直接認証サーバに接続するため，前述のようなユーザとコンテナを結びつける仕組みが不要となる．一方で，認証サーバに接続するためのクライアントパッケージ等をイメージに含める必要があり，イメージの複雑化やサイズの肥大化につながる．

ここで，本システムで作成する演習用イメージはそのまま在宅学習支援システムに流用するものであり，演習室システムと接続しない学生の PC 上でコンテナとして実行した際には演習室システム向けの認証機構は不要であることから，イメージはシンプルに留めるためにユーザ認証はコンテナ外で行うものとして，X 端末起動後に接続される swarm-master のディスプレイマネージャを介してログイン後，コンテナを実行するという形を採った．

ユーザを実行するコンテナを結びつける仕組みは，図 1 の Container Runnner とコンテナ内のスタートアップスクリプトによって提供される．Container Runner は実行するイメージを選択するインタフェースを提供するスクリプトと，Docker コンテナを実行するための'docker run' コマンドのラッパープログラムによって構成されており，ユーザ認証後にディスプレイマネージャによって実行される．ラッパープログラムは，ユーザ情報 (ユーザ名，グループ名，ホームディレクトリパス等) や，利用している X 端末のホスト名を実行するコンテナ内の環境変数としてセットする．演習用コンテナが起動時に実行するスタートアップスクリプトでは，これらの環境変数を用いてコンテナ内にユーザを作成する処理を行う．こうしてコンテナ外のユーザ情報をコンテナ内に擬似的に持ち込むことで，ユーザ情報を持たないイメージから実行ユーザ専用のコンテナを作り出す仕組みを実現している．

2.1.2　ファイル保存

Container Runnner によって docker run コマンドを実行する際，外部保存領域のマウントオプションによって演習室ファイルサーバ内にある実行ユーザのホームディレクトリをコンテナ内にマウントする．こうして揮発領域であるコンテナの外に永続的なデータ保存領域を確保している．

2.1.3　コンテナの起動

マネージャノードである swarm-master 内で実行された docker run コマンドによって，ワーカーノードである swarm-node 上でコンテナが起動する．コンテナが起動するノードは各 swarm-node の実行中コンテナ数に応じて決定され，最も実行中コンテナが少ないノードが選ばれる．

コンテナは起動時にスタートアップスクリプトを実行する．ここで上述のようにコンテナ実行ユーザと同じ情報を持つユーザの作成が行われるほか，演習室内のプリンタ等を利用するためのセットアップ処理が行われる．最後にデスクトップ環境を起動し，ユーザの利用する X 端末へ転送する．

2.1.4　コンテナの廃棄

コンテナは通常，ユーザのログアウトや X 端末の電源断によるセッションの終了に合わせて停止し，その後コンテナをホストしていたサーバ上から削除される．しかし，コンテナ内のデスクトップ環境やアプリケーションがセッションの終了を検知できなかった場合，コンテナはユーザの利用終了後もシステム内に留まり続けてしまう．この問題に対処するため，swarm-master は実行中の各コンテナを起動したそれぞれの X 端末に対して定期的に ping による応答確認を行い，応答がなければその端末の電源が既に断たれており，対応するコンテナはユーザの利用終了後に残ってしまったものであると判断し，停止と削除を行う．

2.2　演習用イメージの管理

本システムで演習環境を提供するコンテナは，そのひな型となるイメージから作成されるが，このイメージについて，デスクトップ環境や基本的な機能を提供するベースイメージと，ベースイメージを継承した上で，個別の演習に必要な機能を追加して作成するイメージに分けることで，演習担当者の裁量によって演習環境を構築できる仕組みを提供している．なお全てのイメージは swarm-master 上で管理され，Dockerfile からビルドされる．以下，それぞれのイメージについて説明する．

- ベースイメージ
  演習室ベースイメージには，デスクトップ環境，ブラウザ，文書作成ソフトウェアといった基本的な機能を提供するパッケージがインストールされている．また，プリンタ等の機器を利用するための演習室内特有の設定情報や，コンテナ起動時に実行されるスタートアップスクリプト等を含んでいる．
- 個別演習用イメージ
  個別演習用イメージはベースイメージを継承する形で作成される子イメージで，Dockerfile の'FROM' タ

© 2017 Information Processing Society of Japan

グにベースイメージを指定することにより，ベースイメージにレイヤを追記する形で新しいイメージを作成することができる．ベースイメージに施された各種設定が継承されるため，イメージ作成者は演習室特有の設定等を意識することなく演習用のイメージを作成することができる．

個別演習用イメージは，ベースイメージの内容だけでは実施できない演習を行う際に，演習担当者が必要なパッケージ，設定等を追加する形で作成する．こうすることで，ベースイメージの管理は演習室管理者，個別演習用イメージの管理は演習担当者と分けることで演習室管理者の負担を削減するほか，演習担当者は基本的な演習室用設定がなされたベースイメージに追記する形で，自由に演習環境を構築することが可能になる．また，ベースイメージが際限なく肥大化することを避け，在宅学習支援システムによって学生にイメージを配布する際のサイズを抑える狙いもある．

イメージのビルド後は，swarm-master の Docker Registry を介して各 swarm-node へイメージを配布するスクリプトが実行される．また，ベースイメージ及び個別演習用イメージは，パッケージのセキュリティアップデート等を適用するため定期的に Dockerfile から再ビルドされ，これも同じく各 swarm-node へ配布される．

また，新しくビルドしたイメージに問題があった場合に備えて，ベースイメージ，個別演習用イメージともに 1 世代前のものはシステム内に保存している．Container Runner ではユーザのログイン後に起動するイメージを選択するダイアログが表示されるが，ここで表示されるイメージはタグによってフィルタリングしているため，表示対象外のタグを付与することでユーザが選択できない状態でシステム内に残しておくことができる．問題発生時には古いイメージを選択可能なタグに戻すだけで演習環境の復旧を行うことができるため，仮想マシン上で演習環境を提供する場合のようなスナップショットの保存やロールアップ作業が不要となっている．

2.3 システム運用結果

2017 年 4 月より，計算機演習室において本システムの運用を開始した．運用開始に先立ち，演習室管理者にはシステム概要や障害発生時の対処法等を記載したマニュアルを，演習担当者には演習実施時の注意事項等を記載したマニュアルと，個別演習用イメージ作成時のマニュアルを必要に応じて配布している．

運用開始後，前システムで生じていた一部のアプリケーションが不具合動作を起こす問題については，大きく報告件数が減った．また，同様の問題が生じたとして報告したユーザはいずれも 2 年次以降の学生であり，本システムへの移行前から演習室システム内にアカウントを持ち，前システムから利用していたユーザに当たる．これらのユーザに対しては当該ソフトウェアの設定ファイル等を初期化して対応する必要があったものの，一度対応してからは再発していない．

また，運用中に実施された演習は 4 つであり，うち学部 1 年生向けの演習では，演習室内の 110 台ある X 端末ほぼ全てを利用する人数が受講した．ユーザのログインに伴って作成されるコンテナは 3 台の swarm-node に均等に分散され，特定のノードに負荷が集中するといった事態は発生しなかった．

3. 在宅学習支援システム

学生が演習室閉室時間帯や自宅等の学外からも利用できるよう，計算機演習室システムでは学外接続サービスを提供しているが，リモートデスクトップ接続であるために，利用上の操作性等は接続に使用する回線品質の影響を大きく受けるほか，従量課金制や転送量上限のある回線を契約している場合には利用しづらい側面もある．そこで，第 2.2 項で演習室用に作成された Docker イメージを，学生が所有する PC 上で実行させることで，演習室閉室時や他演習利用時といった利用不可能な場合にも演習室環境と同等の Linux 学習環境を提供し，在宅学習等に活用してもらうための仕組みの構築を試みた．

学生が所有する PC 上へ Linux 学習環境を提供する試みとしては，Ubuntu の LiveCD を用意し，学習に必要なソフトウェア，コンテンツをインストールし，学生へ配布するといった先行研究が行われているが，この際の課題として LiveCD を必要数分準備するための負担や，一度 LiveCD を用意した後はソフトウェアの追加等の対応が容易ではないという点が挙げられている [3]．一方，本システムにおいては，Linux 学習環境を含むイメージは第 2.2 項で作成したイメージを流用したものであり，学生に提供するにあたって別途作業を行う必要はなく，内容の変更もそのまま反映されるため，このような問題は起こらない．

しかし，作成されたイメージは演習室システム内で実行されることを前提としており，コンテナ内ユーザ情報の設定や外部保存領域はシステム内の認証サーバ，コンテナ起動プログラム，ファイルサーバ等に依存している．そのため，スタンドアロンである学生の PC 上で同じイメージを実行するためには，これらに代わる機能を学生の PC 内に用意する必要がある．

3.1 システム設計

本システムの構成図を図 2 に示す．本システムでは，学生が演習室システムの swarm-master から演習用 Docker イメージを自身の所有する PC 上へコピーし，これを学生の PC 上で実行することで，演習室システム内で利用しているものと同じ演習用コンテナを生成する．

© 2017 Information Processing Society of Japan

本システムで想定している学生の PC のホスト OS は Windows または MacOS であり，Linux コンテナを実行するための Docker Engine はこれらの OS 上で直接実行することはできない．そこで，Docker 社の提供する Docker Toolbox を導入することで，Oracle Virtualbox 内に Linux 仮想マシンを動作させ，その中で Docker Engine を実行する．

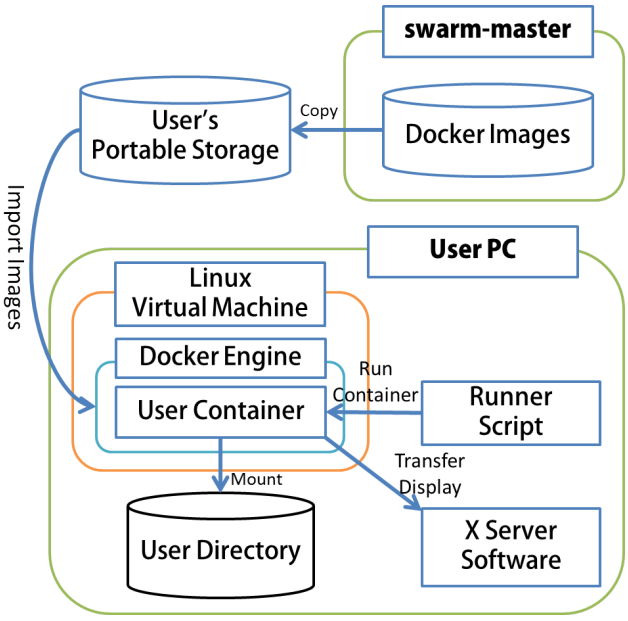

図 2　システム構成図

本システムの利用開始時は，イメージと併せて配布する Runner Script を実行する．これは Linux 仮想マシンの実行，Docker Engine 上のイメージの管理，コンテナの実行を行うためのスクリプトで，ユーザに対してコマンドラインでの対話的インタフェースを提供する．このスクリプトが演習室システム内における'Container Runner' に相当し，コンテナ実行コマンドのラッパーとして，実行されるコンテナに以下の設定を施す．

- コンテナ内の環境変数の設定
  演習室システム内と同様に，コンテナの起動時にはスタートアップスクリプトが実行され，コンテナにセットされた環境変数を参照してユーザを作成する．演習室内では実行ユーザと同様の情報を持ったユーザを作成していたが，学生の PC 上で実行する本システムにおいては，Docker Engine の動作する Linux 仮想マシン上の'Docker' ユーザと同じ UID，GID とすることで，仮想マシン内の保存領域をコンテナ内にマウントした際のアクセス権限を確保している．
- 外部保存領域のマウント
  本システムでは実行する PC 内にコンテナ内の作業データの保存領域を確保する必要がある．作業に必要なファイル，または成果物をシステム内外とやり取りする際にはこの保存領域を経由することになるので，なるべくユーザがアクセスしやすい場所に設定することが望ましい．そこで，PC のホスト OS が直接アクセスできるユーザディレクトリ内に保存領域用のディレクトリを作成，コンテナ内からマウントし，外部保存領域とした．
  ただし，一部の設定ファイル等はコンテナ内と同じファイルシステム (ext4) に保存されていないと正常に読み込まれなかったため，これらは Docker Engine の動作する Linux 仮想マシン内の保存領域をマウントして保存している．

Runner Script は以上の設定を起動するコンテナに対して行い，さらに X サーバソフトウェアを起動する．起動した X サーバのディスプレイ情報はコンテナ内の DISPLAY 環境変数にセットされているため，スタートアップスクリプトが起動したコンテナ内のデスクトップ環境をホスト OS 上の X サーバから操作することができる．

4. おわりに

本論文では，新潟大学工学部情報工学科内の計算機演習室システムについて，これまでの常時稼働の共用サーバ上を運用する上で発生していたトラブルに対処するため，これらの共用サーバ環境を廃して，サーバ上で作成されるコンテナ内で演習環境を実行し，ユーザの利用に合わせて起動・停止する仕組みを作ることで，ユーザにとってスタンドアロンな演習環境を提供するシステムの構築を行った．

また Docker ベースの演習システムを構築するにあたり，Docker イメージの持つ環境再現性，親イメージの継承，可搬性といった性質を活用することで，演習用イメージを学生の PC 上で実行可能にする仕組みを作り，オフライン環境でも学生が演習室と同じ Linux デスクトップ演習環境を利用できるシステムとして提供した．

しかしながら，ベースイメージで対応できない演習に関しては担当者が個別演習用イメージを作成して対処するという仕組みについては，運用開始時にイメージ管理の明確なポリシーを示せていなかったこともあり，現時点では効率的に活用できているとは言えず，ベースイメージに対するパッケージの追加要請が上がってくることがしばしばあった．どこかの時点でベースイメージに含めるパッケージをきちんと選別し，それ以降は全て個別演習用イメージで対応するという方針を周知する必要があると考える．

また現時点では演習環境としてデスクトップ環境のみを提供しているが，現在コンテナ内ターミナルのみをインタフェースとして利用し，デスクトップ環境を扱わない演習環境を提供する機能の追加作業を行っている．現ベースイメージはデスクトップ環境をはじめ，GUI で利用するアプリケーションも多く含まれているが，CUI 環境ではこれらのパッケージは不要となるため，ベースイメージをそ

© 2017 Information Processing Society of Japan

のまま利用することは非効率的である．しかし CUI 専用のイメージを別途作成，管理することも負担となる．そこで，ベースイメージは CUI で利用する機能のみ提供するものに変更し，デスクトップ環境等はこのベースイメージを継承した子イメージで追加することにより，両イメージの管理の一元化を図ることを検討している．現在のベースイメージは GZIP 圧縮状態で 2.6GB，Docker Engine へのインポート後の展開時 5GB 程度の容量を必要とするために，在宅学習支援システムを利用する学生の PC 上のディスク空き容量や，イメージを運ぶ際の USB メモリ等の外部記憶装置の容量に対して相応の要求をしているが，同様の機能をこちらにも追加することで，サイズの小さい CUI ベースのイメージが利用可能になり，学生の選択肢を増やすことが期待できる．

その他の今後の展望としては，演習室システム内に学生が自身のイメージを持つことができる仕組みを検討している．本システムではベースイメージと，これを継承して作成される個別演習用イメージを分けて作成することで，演習室システム内で実行する際の基本的な機能と，各演習に必要な機能を分けて提供する仕組みを持っている．現時点では個別演習用イメージを作成できるのは教職員に限られているが，これを学生にも公開することで，学生が自身で管理できるイメージをシステム内に持ち，演習時に利用したいアプリケーション等を自由に導入できるなど，演習の効率化や自発的な Linux 環境の学習に役立つことを期待できる．そのためには，学生によって作成されたイメージのセキュリティ管理方法や，演習担当者の作成したイメージと学生の作成したイメージをどのようにすり合わせるかといった問題について検討する必要があると考えている．

参考文献

[1] Docker - Build, Ship, and Run Any App, Anywhere `https://www.docker.com/`
[2] 田中 遼，田胡 和也 - コンテナ型仮想化機構を用いた大学向けハイブリッドクラウドの構築， 第 77 回全国大会講演論文集
[3] 橋 文徳，師玉 康成 - Ubuntu による学習環境の構築と評価，コンピュータ＆エデュケーション VOL.25 2008 p74-77

© 2017 Information Processing Society of Japan