AWS を用いたシステムの構築

学籍番号 1250373
溝口 洸熙
グループ 5C

Augst 7th, 2023

1　目的

本実験では，今までオンプレミスで行った作業を，クラウド上で実現する．クラウド化する目的は大きく 2 つある．

第一に，クラウド化することで，管理コストや運用コストを抑えられる．クラウドのサービスにもよるが，オンプレミスでは，サーバ本体を用意し，サーバ本体の管理もしなければならない．サーバ本体にかかるコストもあるが，サーバの管理には電気代や故障した場合の部品代など，運用にもコストが大きくかかる．クラウド化によって，これらのコストは不要になり，管理の手間もクラウドサービスに任せられる．さらに，用意したリソースと必要なリソースの差が問題になる図 1 に対して，クラウドでは図 2 のように，必要に応じてリソースを変更できる．これにより，大きくコストが抑えられる [1].

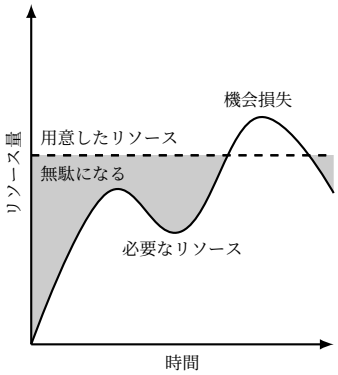

図 1　オンプレミス

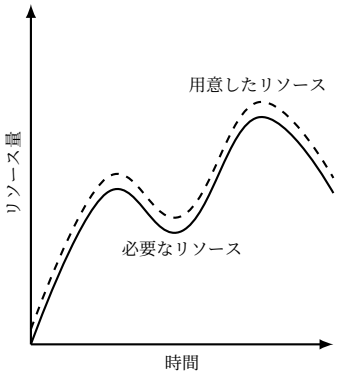

図 2　（例）AWS の利用

第二に，災害対策やセキュリティ対策に対しても，クラウド化による大きな利点がある．オンプレミスでは，当然セキュリティ対策や災害対策も含めて行わねばならない．これにも前述したコストや専門的な知識が必要であり，対策が不十分な場合は損害が発生することもある．クラウドサービスにもよるが，大概のクラウドサービスでは，セキュリティの専門知識がなくても高いセキュリティレベルを実現できる．さらに，国内外の遠隔地にデータを分散できるクラウドサービスは，災害対策にもつながる．

2　内容

クラウドサービスを用いて，以下のことを実現する．

1. Public IP アドレスの取得と，DNS への登録．
2. ファイアウォールの設定．
3. クラウドサービスへのリモートログイン．
4. WordPress を用いた Web ページ公開．
5. Web ページに Basic 認証を適用する．
6. Web ページの SSL 化による HTTPS 通信の実現．
7. Docker をクラウドにインストールし，`httpd` のイメージに対するコンテナを立ち上げる．コンテナを立ち上げると，8080 ポートでコンテナ内の Web サーバへアクセスできるようにする．
8. TEX ファイルをコンパイルするシステムを構築する（図 3）．
   a. FTP クライアントから，クラウドへ FTP 接続し，所定の場所へ TEX ファイルを置く．
   b. クラウドが TEX ファイルの更新を検知し，Docker のコンテナへ TEX ファイルを送る．
   c. クラウド上の Docker コンテナ内で，受信した TEX ファイルの更新を検知し，コンパイルして PDF を生成する．
   d. コンテナ内で生成した PDF ファイルを，コンテナ外から取得する．

e. 生成された PDF を FTP クライアントから取得する．

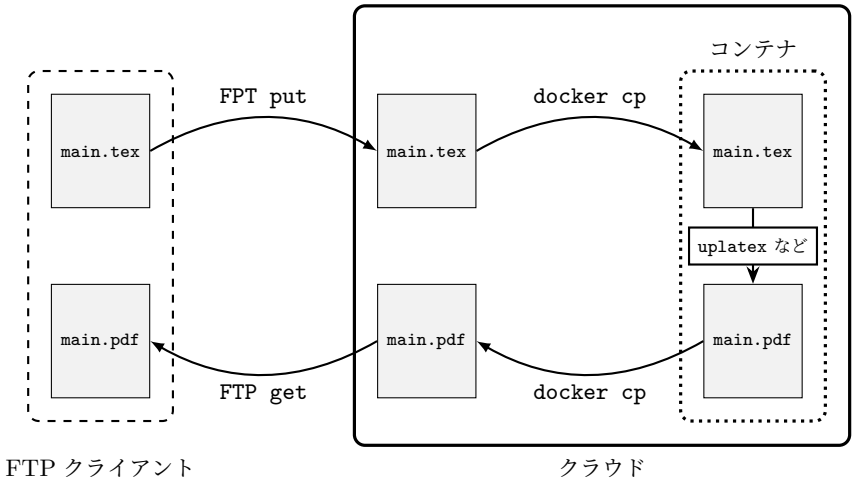

図 3　処理の流れ

3　要素技術

3.1　仮想マシン

仮想化は，辞書で以下のように説明されている．

> “コンピューターの物理的資源を論理的に分割して，それぞれ独立並列した状態で利用できるようにすること。1 台のサーバーで，複数の基本ソフトを独立並列に動作させるサーバー仮想化など。” [2]

仮想マシン（Virtual Machine）は，辞書で以下のように説明されている．

> “あるコンピューターシステムの動作を，別システムで再現するソフトウエア。また，そのような動作環境。ある OS の動作を別の OS 上で再現する場合など。バーチャルマシン。VM。” [2]

つまり，1 台の計算機で複数の OS やアプリケーションなどを並列に動作させる，仮想化を実現するためのソフトウェアや動作環境のことを「仮想マシン」と呼んでいる．今回は「ホスト型」と「ハイパーバイザ型」の仮想化について説明する．

ホスト型

ホスト型の仮想化は，「ホスト型仮想化ソフトウェア」を用いて仮想化する手法である（図 4）．ホスト型の仮想化メリットは，ホスト型仮想化ソフトウェアが扱いやすい点にある．代表的なものとして VirtualBox がある．

ホスト型仮想化のメリットとして挙げられるのが，既存 OS を利用できる点である．それに対して，ホスト OS を動作させるためのリソースが必要になり，物理サーバはもちろん，後述するハイパーバイザ型に比

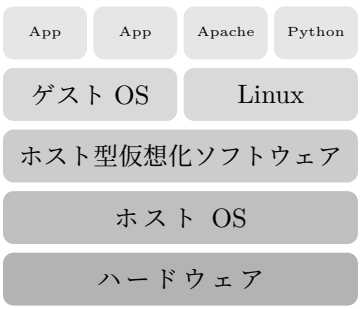

図 4　ホスト型

べても性能が劣る [3].

ハイパーバイザ型

ホスト OS を必要としない,「ハイパーバイザ」と呼ばれる仮想化ソフトウェアを用いた仮想化（図 5）．ホスト OS を必要としないので，ホスト型と比べてリソースの使用効率が良い．このハイパーバイザ型には,「準仮想化」と「完全仮想化」の 2 種類が存在する．

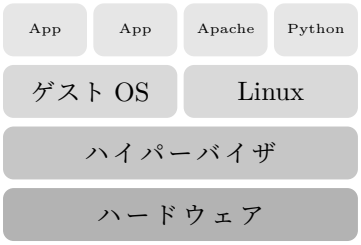

図 5　ハイパーバイザ型

完全仮想化　ハードウェアも含めて仮想化する方式．ユーザモードで動作するゲスト OS から特権命令を実行したとき，必要に応じてハイパーバイザへ処理を移行する．“ユーザモードで動作するゲストオペレーティングシステムは，自分の動作している環境（仮想化であるかないか）をまったく意識する必要がないため，完全仮想化と呼ばれている．”[4, p.159] この方式は，特権命令をハイパーバイザへ移行して処理するので，オーバーヘッドが大きく，ハイパーバイザも特権命令の実行を常時監視する必要がある．

準仮想化　仮想マシンからハードウェアを直接操作できるように，ゲスト OS を改変した方式．完全仮想化に比べてオーバーヘットが激減するが，OS の書き換えが必要であるため，すべての OS には対応できない．

仮想化の利点と欠点

仮想化は，リソースの集約に役立つ技術である．1 台のサーバ上に複数の仮想環境を作ることで，初期コスト，管理コストの削減ができ，リソースを集約することで管理が簡便になる．

ただし，物理サーバに比べて性能が劣り，リソースを集約していることで，ハードウェアの故障による障害範囲が広くなる．

3.2　スケールインとスケールアウト

クラウドでは，仮想サーバのスペック（CPU やメモリサイズなど）や台数の変更によるサーバリソースの調節が簡便にできる．スケールインとスケールアウトは，仮想サーバのリソース調節における分類である．

スケールアップ　サーバのスペックが不足しているときに，仮想サーバのスペックを上げること．
スケールダウン　サーバスペックが過剰であるときに，仮想サーバのスペックを下げること．
スケールアウト　同じようなスペックのサーバを複数台並べて，最適なリソース量を提供すること．
スケールイン　リクエストに対して，サーバ台数が過剰であるとき，サーバの台数を減らして最適なリソース量を提供すること．[5]

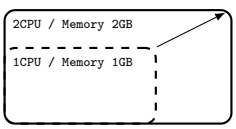

図 6　スケールアップ

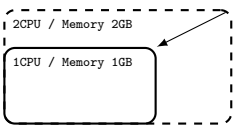

図 7　スケールダウン

図 8　スケールアウト

図 9　スケールイン

3.3 IaaS, PaaS, SaaS

クラウドを利用するとき，クラウド事業者が管理する部分と，クラウド利用者が管理する部分を明確にする必要がある．IaaS，PasS，SaaS とは，クラウド事業者が管理する部分に対して分類したものである．これを「サービスレイヤ」と呼ぶこともある．

SaaS: Software as a Service　アプリケーションソフトウェアをサービスとして提供する．電子メール（Gmail）やオンライン会議システム（Zoom）などがある．

PaaS: Platform as a Service　ライブラリの利用をサービスとして提供する．ユーザはライブラリを用いてアプリケーションを構成する．提供されているライブラリの管理はクラウド事業者が行うので，ユーザはライブラリを意識する必要がない．代表的なものにストレージサービスの Amazon S3 がある．

IaaS: Infrastructure as a Service　インフラストラクチャ（仮想ハードウェアやネットワーク）をサービスとして提供する．ユーザは提供された仮想ハードウェア上に OS をインストールし，そのうえでアプリケーションなどを動かす．代表的なものに Amazon EC2 がある．

[4, p.164]

4 作業記録

内容に基づいた作業記録を記す．今回は，クラウドサービスとして AWS を用いる．

4.1 EC2 インスタンスの作成と諸設定

EC2 インスタンスを，表 1 のもとで作成する．Elastic IP アドレスの割り当てで，パブリック IP アドレスを固定化する．また，表 2 に沿って，ファイアウォールを設定する．外部への，任意のアクセスを許可し，外部からのアクセスは，指定した一部のアクセスのみを許可する．

インスタンスを作成すると，鍵ファイル `c5.pem` のダウンロードが始まる．このファイルは厳重に管理する．また，Elastic IP アドレスを，DNS の A レコードに記載し，Elastic IP とドメイン名 `c5.exp.info.kochi-tech.ac.jp` を対応づける．

4.2 SSL を用いたリモートログインの実現

SSH 接続する Client は，高知工科大学ワークステーションのコンピュータを利用する．

1. ダウンロードされた鍵ファイル `c5.pem` ファイルを`~/.ssh` に移動させる．
2. `~/.ssh` がない場合は，`mkdir` コマンドを用いてディレクトリを作成する．
3. `~/.ssh/config` ファイルがない場合は，`touch` コマンドを用いてファイルを作成して，以下の内容を書き込み，保存する．この設定により，`c5.exp.info.kochi-tech.ac.jp` のホストへアクセスするときに，ユーザ名，鍵ファイルの場所，プロキシの設定を省略できる．インスタンス作成時に，デフォルトで作成されるユーザ名は `ec2-user` である．高知工科大学のプロキシは，`http://proxy.noc.kochi-tech.ac.jp:3128` である．

表 1　インスタンスの作成とネットワークの設定

設定内容	設定値
名前とタグ	c5
OS	Amazon Linux 2023 AMI
アーキテクチャ	64 ビット (x86)
インスタンスタイプ	t2.micro
キーペア	RSA（.pem）
ストレージ	1 x 8 GiB gp3 ルートボリューム（暗号化なし）
Public IP の割り当て	有効化
セキュリティグループ	セキュリティグループを作成する
Elastic IP アドレスの割り当て	有効化（3.94.71.105）
ネットワークボーダーグループ	us-east-1
パブリック IP アドレスルプール	Amazon の IPv4 アドレスプール

表 2　ファイアウォールの設定

ルール	プロトコル	ポート範囲	送信先
アウトバウンドルール	すべて	すべて	0.0.0.0/0

ルール	タイプ/プロトコル	ポート範囲	ソース
インバウンドルール	SSH/TCP	21	0.0.0.0/0
	カスタム TCP/TCP	22	0.0.0.0/0
	HTTP/TCP	80	0.0.0.0/0
	HTTPS/TCP	443	0.0.0.0/0
	カスタム TCP/TCP	8080	0.0.0.0/0
	カスタム TCP/TCP	60000 - 60010	0.0.0.0/0

src. 1　~/.ssh/config

```
Host c5.exp.info.kochi-tech.ac.jp
 User ec2-user
 IdentityFile ~/.ssh/c5.pem
 ProxyCommand nc -X connect -x proxy.noc.kochi-tech.ac.jp:3128 %h %p
```

4. 以下のコマンドを実行し，インスタンスへ SSH 接続できるか確認する．

```
$ sudo chmod 600 ~/ssh/c5.pem
$ ssh c5.exp.info.kochi-tech.ac.jp
```

以下の出力を確認できたら，SSH によるリモートログインに成功している．

```
A newer release of "Amazon Linux" is available.
  Version 2023.1.20230725:
```

```
Run "/usr/bin/dnf check-release-update" for full release and version update
  info
    ,     #_
    ~\_  ####_        Amazon Linux 2023
   ~~  \_#####\
   ~~     \###|
   ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
    ~~       V~' '->
     ~~~         /
       ~~._.   _/
          _/ _/
        _/m/'
Last login: Thu Aug  3 14:25:50 2023 from xxx.xxx.xxx.xxx
[ec2-user@ip-172-31-93-16 ~]$
```

4.3 WordPress のインストールと Web ページの公開

WordPress を用いた Web ページを公開する．

1. WordPress に必要なパッケージをインストールする．

```
$ sudo dnf install wget php-mysqlnd httpd php-fpm php-mysqli mariadb105-
  server php-json php php-devel -y
```

2. データベース，Web サーバを起動する．

```
$ sudo systemctl start mariadb httpd
```

3. wget コマンドを利用して，WordPress のパッケージをダウンロード，解凍し，所定の場所にコピーする．そして，/var/www/wordpress のディレクトリ管理権限を apache に変更する．

```
$ sudo cd /tmp
$ sudo wget http://ja.wordpress.org/latest.tar.gz
$ sudo tar -xvf latest.tar.gz
$ sudo cp -r wordpress /var/www/
$ sudo cp -r wordpress/* /var/www/html/
$ sudo chown -R apache.apache /var/www
```

4. httpd の設定ファイルを更新する．
   - DocumentRoot を/var/www/html に変更．
   - /var/www/html のディレクトリに，.htasccess によるディレクティブ上書きを許可する．さらに，ファイル名を指定しないアクセスに対しては index.php を返す設定をする．

src. 2 /etc/httpd/conf/httpd.conf

```
– 略 –
DocumentRoot "/var/www/html"
– 略 –
```

```
<Directory "/var/www/html">
            AllowOverride All
– 略 –
<IfModule dir_module>
            DirectoryIndex index.php
</IfModule>
– 略 –
```

5. データベースにアクセスして，WordPress 用のユーザを作成する．作成したユーザに対して，完全な権限を付与する．

```
$ sudo mysql -u root -p
> CREATE USER 'wordpress-user'@'localhost' IDENTIFIED BY '******';
> CREATE DATABASE `wordpress-db`;
> GRANT ALL PRIVILEGES ON `wordpress-db`.* TO "wordpress-user"@"localhost";
> FLUSH PRIVILEGES;
```

6. https://api.wordpress.org/secret-key/1.1/salt/ にアクセスして，ランダムに生成されるキーセット値を取得する．
7. /var/www/html/wp-config.php ファイルでデータベース情報と，キーセットを登録する．

src. 3 /var/www/html/wp-config.php

```
define( 'DB_NAME', 'wordpress-db' );
define( 'DB_USER', 'wordpress-user' );
define( 'DB_PASSWORD', '******' );
define( 'DB_HOST', 'localhost' );
define( 'DB_CHARSET', 'utf8' );
define( 'DB_COLLATE', '' );
define('AUTH_KEY',         '**secret**');
define('SECURE_AUTH_KEY',  '**secret**');
define('LOGGED_IN_KEY',    '**secret**');
define('NONCE_KEY',        '**secret**');
define('AUTH_SALT',        '**secret**');
define('SECURE_AUTH_SALT', '**secret**');
define('LOGGED_IN_SALT',   '**secret**');
define('NONCE_SALT',       '**secret**');
```

8. /var/www 以下のディレクトリに対して，パーミッションを変更し，SGID を付与する．

```
$ sudo chmod 2775 /var/www
$ find /var/www -type d -exec sudo chmod 2775 {} \;
$ find /var/www -type f -exec sudo chmod 0644 {} \;
```

9. httpd を再起動する．

```
$ sudo systemctl restart httpd
```

10. c5.exp.info.kochi-tech.ac.jp にアクセスして，WordPress が表示されるか確認する．
11. サイトのタイトル，ユーザ名を設定して，WordPress 内で記事を編集し，公開する．

4.4 Basic 認証の適用

/var/www/html/basic への Web アクセスに，Basic 認証を適用する．

1. /var/www/html/basic/index.html を作成し，index.html に適当な Web ページを HTML で記述する．
2. Basic 認証用の ID とパスワード組みを作成する．

```
$ sudo htpasswd -c /etc/httpd/htpasswd groupc5
New password:
Re-type new password:
Adding password for user groupc5
$ cat /etc/httpd/htpasswd
groupc5:$apr1$ZsK7Iq9f$fq1F9XSDaBH2j/8RBHZZP/
$ sudo chown apache:apache /etc/httpd/htpasswd
$ sudo chmod 600 /etc/httpd/htpasswd
```

3. /etc/httpd/conf/httpd.conf に以下を追加し，Basic 認証を適用する．

src. 4 /etc/httpd/conf/httpd.conf

```
<Directory "/var/www/html/basic">
    AuthType Basic
    AuthName "auth"
    AuthUserFile /etc/httpd/htpasswd
    Require valid-user
</Directory>
```

4. httpd を再起動する．
5. c5.exp.info.kochi-tech.ac.jp/basic/index.html にアクセスし，Basic 認証が適用できているか確認する．

4.5 Web ページの SSL 化

Let's Encrypt を用いて HTTPS 通信を実現する．

1. 必要なパッケージをインストールし，Python3 をセットアップする．

```
$ sudo dnf install -y python3 augeas-libs pip
$ sudo python3 -m venv /opt/certbot/
$ sudo /opt/certbot/bin/pip install --upgrade pip
$ sudo /opt/certbot/bin/pip install certbot
$ sudo ln -s /opt/certbot/bin/certbot /usr/bin/certbot
```

2. Web サーバを停止する．

```
$ sudo systemctl stop httpd
```

3. certbot を実行し，質問に回答する．これにより，証明書が作成される．

```
$ sudo certbot certonly --standalone
```

- 証明書の期限切れを通知するメールアドレス：250373b@ugs.kochi-tech.ac.jp
- 利用規約に同意するか：Y
- メールアドレスを共有するか？ ：N
- 証明書に記載するドメイン名：c5.exp.info.kochi-tech.ac.jp

完了したら，証明書が/etc/letsencrypt/live/c5.kochi-tech.ac.jp 下に作成される．

4. /etc/httpd/conf.d/ssl.conf へ以下を追加し，証明書などのファイル所在を記す．

src. 5 /etc/httpd/conf.d/ssl.conf

```
– 略 –
<VirtualHost _default_:443>
 ErrorLog logs/ssl_error_log
 TransferLog logs/ssl_access_log
 LogLevel warn
 SSLEngine on
 ServerName      c5.exp.info.kochi-tech.ac.jp
 DocumentRoot    /var/www/html
 SSLHonorCipherOrder on
 SSLCipherSuite PROFILE=SYSTEM
 SSLProxyCipherSuite PROFILE=SYSTEM
 SSLCertificateFile      /etc/letsencrypt/live/c5.exp.info.kochi-tech.ac.jp/
   cert.pem
 SSLCertificateKeyFile   /etc/letsencrypt/live/c5.exp.info.kochi-tech.ac.jp/
   privkey.pem
 SSLCertificateChainFile /etc/letsencrypt/live/c5.exp.info.kochi-tech.ac.jp/
   chain.pem
 – 略 –
</VirtualHost>
– 略 –
```

5. httpd を起動する．

```
$ sudo systemctl start httpd
```

6. https://c5.exp.info.kochi-tech.ac.jp にアクセスし，SSL の適用を確かめる．

4.6 Docker のインストールとコンテナの構築

1. Docker のパッケージをインストールし，有効化する．権限も更新する．

```
$ sudo dnf install -y docker
$ sudo systemctl enable --now docker
$ sudo systemctl status docker
$ sudo usermod -aG docker ec2-user
```

2. Docker のコンテナを作成する．ここではイメージをレジストリから取得する．今回は，httpd のイメージを利用して，8080 番ポートで HTTP 通信を実現する．

```
$ docker pull httpd
$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
httpd        latest    d140b777a247   3 weeks ago   168MB
```

3. コンテナを作成する．8080 番ポートに来たアクセスをコンテナ内の 80 番ポートにフォワーディングする．

```
$ docker run httpd -itd -p 8080:80
```

4. 作成したコンテナが動いているか確認する．

```
$ docker ps
CONTAINER ID   IMAGE     COMMAND              CREATED         STATUS
  PORTS                                  NAMES
0fd95973b539   httpd     "httpd-foreground"   8 minutes ago   Up 8 minutes
  0.0.0.0:8080->80/tcp, :::8080->80/tcp   wizardly_galois
```

5. Docker のコンテナへ，Web アクセスできることを確認する．http://c5.exp.info.kochi-tech.ac.jp:8080 へアクセスし，「It Works!」の文字を確認する．

4.7 TEX ファイルをコンパイルするシステムの構築

FTP で TEX ファイルを送信すると PDF を生成するシステムを作成する．FTP でファイル（main.tex）を所定箇所に置いたら，Bash がファイルの更新を検知して，Docker コンテナへ main.tex を転送する．Docker コンテナに送信された main.tex は，コンテナ内で main.pdf を生成し，EC2 インスタンスへ，main.pdf を転送する．転送された main.pdf を FTP を用いて，取得する．

1. FTP サーバをインスタンス上に構築する．vsftpd パッケージをインストール後，ユーザを作成する．

```
$ sudo yum install vsftpd -y
$ sudo adduser --home /home/ftp-user ftp-user
$ sudo passwd ftp_user
$ sudo chown ftp-user:ftp-user -R /home/ftp-user/
```

2. 設定ファイルを以下のように変更する．

src. 6 /etc/vsftpd/vsftpd.conf

```
anonymous_enable=NO
local_enable=YES
ascii_upload_enable=YES
ascii_download_enable=YES
write_enable=YES
local_umask=022
dirmessage_enable=YES
```

```
xferlog_enable=YES
xferlog_std_format=YES
listen=YES
listen_ipv6=NO
pam_service_name=vsftpd
userlist_enable=YES
tcp_wrappers=NO
use_localtime=YES
force_dot_files=YES
pasv_address=3.94.71.105
pasv_enable=YES
pasv_addr_resolve=YES
pasv_min_port=60000
pasv_max_port=60010
userlist_deny=NO
```

3. Docker のレジストリから TEX ファイルをコンパイルするソフトウェアを含むイメージを Pull する.

```
$ docker pull ghcr.io/being24/latex-docker
$ docker images
REPOSITORY                      TAG       IMAGE ID       CREATED       SIZE
ghcr.io/being24/latex-docker    latest    bc414bef2302   3 weeks ago   2.2GB
– 略 –
$ docker run --name latex bc414bef2302
$ docker ps
CONTAINER ID   IMAGE          COMMAND       CREATED      STATUS
  PORTS     NAMES
19b52251ed12   bc414bef2302   "/bin/bash"   7 days ago   Up 12 minutes
             latex
– 略 –
```

4. 事前に作成した更新を検知してコンパイルするシェルスクリプトを docker cp コマンドを用いて転送する.

```
$ docker cp comp.sh 19b52251ed12:/workdir
```

src. 7 19b52251ed12:/workdir/comp.sh

```
#!/bin/sh
echo "監視対象 main.tex"
INTERVAL=1 # 監視間隔
last=`ls --full-time main.tex | awk '{print $6"-"$7}'`
while true; do
    sleep $INTERVAL
    current=`ls --full-time main.tex | awk '{print $6"-"$7}'`
    if [ $last != $current ] ; then
        echo ""
        echo "updated: $current"
        last=$current
```

```
        uplatex -interaction=nonstopmode main.tex
        dvipdfmx main.dvi
    fi
done
```

5. main.tex が更新されたら，Docker のコンテナへファイルを転送するシェルスクリプトを/home/ftp-user に置く．

src. 8 /home/ftp-user/file_check.sh

```
#!/bin/sh
echo "監視対象 main.tex"
echo "実行コマンド docker cp main.tex 19b52251ed12:/workdir"
INTERVAL=1 #監視間隔，秒で指定
last=`ls --full-time main.tex | awk '{print $6"-"$7}'`
while true; do
    sleep $INTERVAL
    current=`ls --full-time main.tex | awk '{print $6"-"$7}'`
    if [ $last != $current ] ; then
        echo ""
        echo "updated: $current"
        last=$current
        docker cp main.tex 19b52251ed12:/workdir
        sleep 5
        docker cp 19b52251ed12:/workdir/main.pdf /home/ftp-user/main.pdf
        echo "compiled main.tex"
    fi
done
```

6. /home/ftp-user/file_check.sh を実行する．

```
$ sudo bash /home/ftp-user/file_check.sh &
```

7. コンテナ内に入って，/workdir/comp.sh を実行する．

```
$ docker exec -it latex bash
root@19b52251ed12:/workdir# bash comp.sh
```

8. 適当な TEX 文書を main.tex に書く．
9. FTP クライアントから，EC2 インスタンスへ FTP 接続し，main.tex ファイルを置く．5 秒くらい経ったら，main.pdf を取得する．

```
$ ftp ftp-user@c5.exp.info.kochi-tech.ac.jp
Connected to c5.exp.info.kochi-tech.ac.jp.
220 (vsFTPd 3.0.5)
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> put main.tex
local: main.tex remote: main.tex
229 Entering Extended Passive Mode (|||60007|)
150 Ok to send data.

  0 | | 0 0.00 KiB/s --:-- ETA 100 |*********************************************|    94      312.23 KiB/s    00:00 ETA
226 Transfer complete.
94 bytes sent in 00:00 (0.24 KiB/s)
```

```
ftp> get main.pdf
local: main.pdf remote: main.pdf
229 Entering Extended Passive Mode (|||60003|)
150 Opening BINARY mode data connection for main.pdf (6269 bytes).

  0 | | 0 0.00 KiB/s --:-- ETA 100 |***********************************************|  6269        8.37 MiB/s    00:00 ETA
226 Transfer complete.
6269 bytes received in 00:00 (32.85 KiB/s)
ftp> quit
221 Goodbye.
$
```

10. `main.pdf` が `main.tex` をコンパイルしたものであるか確認する．

5 考察

オートスケーリングとその問題について取り上げる．オートスケーリング（オートスケール）とは，3 節で解説したスケーリングを，自動で行うしくみである．AWS でのオートスケーリングは以下のように説明されている．

> “AWS Auto Scaling は、安定した予測可能なパフォーマンスを可能な限り低コストで維持するためにアプリケーションをモニタリングし、容量を自動で調整します。” [6]

オートスケーリングでは，利用状況に応じて，スケールアップ，スケールダウン，スケールアウト，スケールインする．オートスケーリングを用いることで，仮想化の「コストや管理の手間を削減できる」というメリット最大限引き出せる．オートスケーリングのデメリットを説明するために，「ステートレス」と「ステートフル」の言葉を導入する．

ステートレス “利用者の状態をシステムが保持しており、その内容に応じて処理結果を変えること。”

ステートフル “利用者の状態をシステムの内部に保持せず、入力された値だけを使用して処理すること。同じ入力に対しては、常に同じ結果が得られる。”

[7, p.64]

ここで，オートスケーリングでスケールインやスケールアウトした場合を考える．ステートレスでは，サーバの増減にかかわらず，どのサーバでも処理を引き継げる．これは，ユーザの状態をシステムが保持していないことに起因する．しかし，ステートフルの場合は，ユーザの入力状況によってサーバの処理が変わる．スケールインやスケールアウトしたとき，ユーザの状態がほかのサーバに引き継がれていない場合は障害が生じる．これを解決するためには，ユーザの状態をサーバ間で共有するしくみが必要だ．加えて，オートスケーリングによるスケールインでは，サーバが「削除」されるので，サーバにデータが保存される場合はその注意が必要だ．予期せぬデータの消失を防ぐために，サーバを削除する前にスケールアウトで作成したサーバと，削除前の差分をバックアップとして自動的に保存する方法を提案する．これにより，万が一データが消失した場合でもデータの復元が簡単であるほか，差分で保存することによりバックアップファイルサイズを小さくできる．

参考文献

[1] 佐々木拓郎, 林晋一郎, 小西秀和, 佐藤瞬. Amazon Web Services パターン別構築・運用ガイド: 一番大切な知識と技術が身につく. SB クリエイティブ, 2015.

[2] 三省堂編. スーパー大辞林. 三省堂, 2020.

[3] アイティーエム株式会社. 仮想化技術について解説. https://www.itmanage.co.jp/column/virtualization-server-integration/, Last confirmed date: August 6th, 2023.
[4] 松尾啓志. オペレーティングシステム（情報工学レクチャーシリーズ）. 情報工学レクチャーシリーズ. 森北出版, 2018.
[5] SAKURA internet. サーバのスケールアップ・スケールダウン・スケールアウト・スケールイン. https://knowledge.sakura.ad.jp/6217/, Last confirmed date: August 6th, 2023.
[6] Inc Amazon Web Services. AWS auto scaling. https://aws.amazon.com/jp/autoscaling/, Last Confirmed date: August 6th, 2023.
[7] 増井敏克. IT 用語図鑑 ―ビジネスで使える厳選キーワード 256―. 翔泳社, 2019.