情報工学実験Ⅱ報告書

	点

実験クール（1・2・3・4）	4	実験回（1・2・3）	3	クラス（A・B）	B

実験番号 3　実験題目 **サーバー構築**

グループ（A・B・C・D）	D	学籍番号	EP20050	氏名	小池　正基

	実験実施日	実験項目（実施概要）
1	2023／12／05	課題 1~6
2	2023／12／12	課題 7~14
3	2023／12／19	課題 15~16

共同実験者（学籍番号）

	中間①提出日	中間②提出日	完成提出日	完成再提出①	完成再提出②
提出期日（予定日）	2023／12／12	2023／12／19	2023／01／04	20　／　／	20　／　／
提出日	2023／12／12	2023／12／19	2023／01／09	20　／　／	20　／　／
ルーブリック評価点					

コメント

目次

1　課題 1

1.1　目的

VPN 接続とは Virtual Private Network の略称で，日本語にすると「仮想専用通信網」を意味する．VPN 接続を行うことで，不正なアクセスを防止し，第三者にデータを登用される心配が少なくなる．また，他のネットワークに安全にアクセス可能であるため，社内ネットワークや，海外などの離れた拠点に接続するために，頻繁に用いられる．

1.2　理論

VPN 接続を行うとき，送信側，受信側にそれぞれ設置した機器で「カプセル化」と呼ばれる処理を行うことで，第三者には見えない仮想的なトンネルを形成（トンネリング）して通信する．また，通信利用者が正規の利用者であることを確認するために認証を行う，トンネルに侵入された場合に備えて暗号化した通信を行うなどの技術も併用することで，より安全性の高い通信が担保されている．

トンネルにはプロトコル（通信規約）が設定されている．これは VPN 接続の手法を示すものであり，「IPsec」，「L2TP/IPsec」，「PPTP」などが主流である．それぞれのプロトコルにおいて通信手法が異なる．例えば，L2TP．PPTP はデータの暗号化を行わないが，IPsec，L2TP/IPsec はデータの内容を暗号化して通信する．

1.3　実験内容

今回の実験では，サーバ [157.110.45.50] に対して VPN 接続を行うことで，VPN 接続が可能であるか接続確認を行う．実験条件を表 1 に示す．

表 1: VPN 接続設定

VPN プロバイダ	Windows（ビルトイン）
接続名	VPN
サーバ名，アドレス	157.110.45.50
VPN の種類	事前共有キーを使った L2TP/IPsec
事前共有キー	chubu_admin
ユーザー名	ep20050
パスワード	実験者が定めたパスワード

以下に Windows10 を用いて VPN 接続を確立する方法を示す．ここで，コンソールは Windows PowerShell 7.3.10 を用いた．PowerShell による VPN 接続の作成には，「Add-VpnConnection」コマンドレットを用いる．Add-VPNConnection を用いて VPN 接続設定を追加するスクリプトをソースコード 1 に示す．

Source Code 1: VPN 設定スクリプト

```powershell
$vpnName = "VPN"
$serverAddress = "157.110.45.50"
$psk = "chubu_admin"

# VPN 接続の追加
Add-VpnConnection -Name $vpnName `
    -ServerAddress $serverAddress `
    -TunnelType L2tp `
    -EncryptionLevel Optional `
    -AuthenticationMethod Chap `
    -RememberCredential `
    -SplitTunneling `
    -L2tpPsk $psk `
    -Force
```

このプログラムを「ファイル名.ps1」といった拡張子で保存し，右クリックメニュー内の「PowerShell で実行」を選択することで実行することができる．プログラムの内容について簡単に説明する．プログラム内の Add-VpnConnection が VPN 接続の追加を行うコマンドである．ここで，引数として VPN の名前，アドレス，事前共有キー，暗号化設定を記述している．特に，EncryptionLevel（暗号化レベル）を Optional に，AuthenticationMethod（認証方式）を Chap にしなければ接続時にエラーが発生するため，注意が必要である．

ソースコード 1 を用いて VPN 設定が完了したら，VPN 接続を行うスクリプトであるソースコード 2 を実行して接続を行う．このとき，$VpnUser がユーザ名，$VpnPass がパスワードである．

Source Code 2: VPN 接続スクリプト

```powershell
$RasExec = "C:\windows\system32\rasdial.exe"
$VpnName = "VPN"
$VpnUser = "ep20050"
$VpnPass = "*******"

# 接続
cmd.exe /c $RasExec $VpnName $VpnUser $VpnPass


# 切断
# cmd.exe /c $RasExec $VpnName /Disconnect
```

1.4 実験結果

ソースコード 1，ソースコード 2 を実行し，VPN 接続を確認するまでのログをソースコード 3 に示す．

Source Code 3: VPN 接続ログ

```
PS C:\Users\zigza\OneDrive\ドキュメント\0_大学資料\4.1_秋\server> .\setting.ps1
WARNING: The currently selected encryption level requires EAP or MS-CHAPv2 logon security methods. Data encryption will not occur for Pap or Chap.
PS C:\Users\zigza\OneDrive\ドキュメント\0_大学資料\4.1_秋\server> .\connectVPN.ps1
VPN に接続中...
ユーザー名とパスワードを確認中...
ネットワークにコンピューターを登録中...
VPN に正常に接続しました。
コマンドは正常に終了しました。
PS C:\Users\zigza\OneDrive\ドキュメント\0_大学資料\4.1_秋\server> Get-VpnConnection | Where-Object { $_.ConnectionStatus -eq "Connected" }

Name                    : VPN
ServerAddress           : 157.110.45.50
AllUserConnection       : False
Guid                    : {6B45C4F5-5C40-4257-AFBA-051D6B055598}
TunnelType              : L2tp
AuthenticationMethod    : {Chap}
EncryptionLevel         : Optional
L2tpIPsecAuth           : Psk
UseWinlogonCredential   : False
EapConfigXmlStream      :
ConnectionStatus        : Connected
RememberCredential      : True
SplitTunneling          : True
DnsSuffix               :
IdleDisconnectSeconds   : 0
```

1.5 考察

ソースコード 3 内の Get-VpnConnection から，本実験において接続する VPN の情報を取得できていることが確認できる．Get-VpnConnection は VPN の情報を取得するもので，今回は Where-Object によって接続済みの VPN のみを指定してるため，正常に接続できたことがわかる．しかし，暗号化レベルの設定，認証方式の設定を間違えると，それぞれ
「リモート アクセス エラー 788 - セキュリティ層でリモート コンピューターと互換性のあるパラメーターをネゴシエートできなかったため，L2TP 接続に失敗しました．」
と，
「リモート アクセス エラー 628 - リモート コンピューターにより接続が途中で切断されました．」

が発生し，接続が不可能である．これは，ホストとクライアントの暗号化方式に差があると正常にデータのやりとりができないこと，第三者からの介入に対して脆弱になってしまうことが原因である．

また，Windows11 を用いて VPN 接続を行う場合，「リモートサーバが応答しないため，使用するコンピューターと VPN サーバ間のネットワーク接続を確立できませんでした」というエラーが発生し，Windows のレジストリ設定が必須である．この場合，`$HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\PolicyAgent$`内で「新規→DWOED（32 ビット）値」をクリックし，名前を「AssumeUDPEncapsulationContextOnSendRule」とし，[値のデータ] を 2 として再起動することで解決する．

2　課題 2,3,4,5

2.1　課題内容

インターネット環境 EP_Lab2，CWN，学外 VPN 接続，学外 VPN 未接続時について，コマンド tracert を用いて比較実験を行う．

2.2　目的

本課題では，Wi_Fi 環境「EP_Lab2」と「CWN」，「学外 (VPN 未接続)」，「学外 (VPN 接続)」の接続経路を tracert を用いて確認し，通信経路の違いを確認する．

2.3　理論，実験手順

tracert を PowerShell で実行することで，データパケットが目的地に到達するまでのルートを確認することができる．Wi_Fi 環境「EP_Lab2」と「CWN」，学外環境にそれぞれ接続し，PowerShell 内でコマンド tracert を用いて通信経路を確認し，結果を比較する．

2.4　実験結果

EP_Lab2 で tracert を実行したときの通信経路をソースコード 4 に示す（課題 2）．

Source Code 4: EP_Lab2 の通信経路

```
PS C:\Users\zigza> tracert 8.8.8.8

dns.google [8.8.8.8] へのルートをトレースしています
経由するホップ数は最大 30 です:

  1     *        *        *     要求がタイムアウトしました。
  2     *        *        *     要求がタイムアウトしました。
  3     *        *        *     要求がタイムアウトしました。
  4     *        *        *     要求がタイムアウトしました。
  5     *        *        *     要求がタイムアウトしました。
  6     *        *        *     要求がタイムアウトしました。
  7     *        *        *     要求がタイムアウトしました。
```

```
  8     *        *        *     要求がタイムアウトしました。
  9     *        *        *     要求がタイムアウトしました。
 10     6 ms     7 ms    17 ms  dns.google [8.8.8.8]

トレースを完了しました。
```

CWN で tracert を実行したときの通信経路をソースコード 5 に示す（課題 2）.

Source Code 5: CWN の通信経路

```
PS C:\Users\zigza> tracert 8.8.8.8

dns.google [8.8.8.8] へのルートをトレースしています
経由するホップ数は最大 30 です:

  1     4 ms     3 ms     3 ms  192.168.100.1
  2     4 ms     4 ms     3 ms  157.110.44.1
  3     5 ms     4 ms     4 ms  157.110.68.3
  4     7 ms     4 ms     3 ms  kasugai.isc.chubu.ac.jp [157.110.66.65]
  5     5 ms     4 ms     4 ms  150.99.192.89
  6     9 ms     9 ms     8 ms  150.99.16.249
  7     7 ms     7 ms     8 ms  103.246.232.96
  8     9 ms     9 ms     8 ms  108.170.243.97
  9     8 ms    10 ms     7 ms  142.250.239.221
 10     6 ms     7 ms     7 ms  dns.google [8.8.8.8]

トレースを完了しました。
```

学外インターネット環境から VPN 接続を行った結果をソースコード 6 に示す（課題 3）.

Source Code 6: 学外からの VPN 接続ログ

```
PS C:\Users\zigza\OneDrive\ドキュメント\0_大学資料\4.1_秋\server> .\connectVPN.ps1
VPN に接続中...
ユーザー名とパスワードを確認中...
ネットワークにコンピューターを登録中...
VPN に正常に接続しました。
コマンドは正常に終了しました。
PS C:\Users\zigza\OneDrive\ドキュメント\0_大学資料\4.1_秋\server> Get-VpnConnection
↪  | Where-Object { $_.ConnectionStatus -eq "Connected" }

Name                    : VPN
ServerAddress           : 157.110.45.50
AllUserConnection       : False
Guid                    : {6B45C4F5-5C40-4257-AFBA-051D6B055598}
TunnelType              : L2tp
AuthenticationMethod    : {Chap}
EncryptionLevel         : Optional
L2tpIPsecAuth           : Psk
```

```
UseWinlogonCredential : False
EapConfigXmlStream    :
ConnectionStatus      : Connected
RememberCredential    : True
SplitTunneling        : True
DnsSuffix             :
IdleDisconnectSeconds : 0
```

学外で VPN 接続を行わずに tracert を実行したときの通信経路をソースコード 7 に示す（課題 4）.

Source Code 7: 学外の通信経路（VPN 未接続）

```
PS C:\Users\zigza>  tracert 8.8.8.8

dns.google [8.8.8.8] へのルートをトレースしています
経由するホップ数は最大 30 です:

  1    <1 ms    <1 ms    <1 ms  192.168.3.1
  2    10 ms    10 ms    10 ms  softbank221110187114.bbtec.net [221.110.187.114]

  3    15 ms    23 ms    13 ms  softbank221110187113.bbtec.net [221.110.187.113]

  4    10 ms    10 ms    10 ms  10.2.9.93
  5    11 ms    10 ms    10 ms  10.9.203.114
  6    11 ms    10 ms    10 ms  72.14.214.193
  7    11 ms    10 ms    10 ms  108.170.243.129
  8    11 ms    11 ms    11 ms  108.170.235.7
  9    10 ms    10 ms    10 ms  dns.google [8.8.8.8]

トレースを完了しました。
PS C:\Users\zigza>
```

学外で VPN 接続を行い，tracert を実行したときの通信経路をソースコード 8 に示す（課題 5）.

Source Code 8: 学外の通信経路（VPN 接続）

```
PS C:\Users\zigza>  tracert 8.8.8.8

dns.google [8.8.8.8] へのルートをトレースしています
経由するホップ数は最大 30 です:

  1    <1 ms    <1 ms    <1 ms  192.168.3.1
  2    10 ms    10 ms    10 ms  softbank221110187114.bbtec.net [221.110.187.114]

  3    10 ms    13 ms    13 ms  softbank221110187113.bbtec.net [221.110.187.113]

```

```
  4    10 ms    11 ms    10 ms  10.2.9.93
  5    11 ms    10 ms    11 ms  10.9.203.114
  6    11 ms    10 ms    10 ms  72.14.214.193
  7    12 ms    11 ms    10 ms  108.170.243.129
  8    11 ms    11 ms    11 ms  108.170.235.7
  9    10 ms    10 ms    10 ms  dns.google [8.8.8.8]

トレースを完了しました。
PS C:\Users\zigza>
```

2.5 考察

課題 3 については，課題 1 と同様に接続が完了したことを確認できた．課題 2,4,5 について確認すると，CWN 及び EP_Lab2 は 10 台のルーターを経由して最終地点に接続したことがわかる．また，EP_Lab2 に限り通信経路が確認できなかった．これは，何らかのセキュリティによって応答をブロックしている可能性が考えられる．学外からの接続について確認すると，CWN と別経路であり，契約会社のソフトバンクのルーターを経由して目的地に到達している．学外のルーターは学内よりも 1 つ少ないルーター経由数であり，これは中部大学の学内ネットワークを経由しないため 1 つ少ないと考察できる．学外からの通信経路を VPN あり，なしの観点から見ると，同様のトレースを行っていることが確認できた．これは何らかの設定によって VPN を介さずに通信を行っている可能性がある．すべての結果を確認すると，CWN が最も通信速度が速く，次に学外環境，EP_Lab2 に続く結果となった．この結果から，CWN は複数人の利用に耐えうるよう高速に設計されている可能性がある．

3 課題 6

3.1 課題内容

https://www.isc.chubu.ac.jp/procedure/chubu/linuxaccount2.html への接続確認（学内，学外，VPN 接続時でそれぞれ確認し考察を行う）する実験を行い，実験の目的，理論＜方法論＞，実験手順，実験結果，考察を述べよ．

3.2 目的

学内，学外，VPN 接続時でそれぞれの通信経路が異なるかを確認する．

3.3 理論

学内，学外，VPN 接続時でそれぞれの通信経路が異なると仮定すると，Web ページ https://www.isc.chubu.ac.jp/procedure/chubu/linuxaccount2.html に接続し，学内ネットワークにアクセスできるか確認する．

3.4　実験手順

本実験では，学外ネットワーク，VPN, 学内ネットワークの 3 種類の方法で中部大学内ネットワークの Web ページにアクセスし，正常にアクセス可能か確認する．

3.5　実験結果

学内, 学外, VPN 接続時で https://www.isc.chubu.ac.jp/procedure/chubu/linuxaccount2.html にアクセスした結果を図 4 に示す．

図 1: 学内ネットワーク

図 2: 学外ネットワーク

図 3: VPN

図 4: 大学内ネットワークへのアクセス結果

3.6　考察

図 4 から，学外ネットワークからはアクセスできないのに対し，学内ネットワーク，VPN 接続時には正常にアクセスできていることがわかる．これは，学外からのアクセスに対し，VPN を用いることで学内ネットワークに疑似的に通信できているためであると推測できる．これにより，VPN 接続によって他のネットワークにアクセス可能であると結論付けることができる．

4　課題 7

SSH によるサーバ接続を確認する実験を行い，実験の目的，理論＜方法論＞，実験手順，実験結果，考察を述べよ．

4.1　目的

本課題では，ssh 接続によってサーバ接続を行うことで，安全なリモートコンピュータへのアクセスが可能であるか確認すると同時に，サーバに遠隔接続して操作を行うことで，複数人で行うプロジェクトへの理解を深めることを目的とする．

4.2　理論

ssh とは，Secure Shell（セキュアシェル）の略称で，リモートコンピュータと通信するためのプロトコルである．認証部分を含めネットワーク上の通信がすべて暗号化されるため，安全に通信することができる．従来手法の Telnet や FTP はパスワードを暗号化しないため，盗聴のリスクがあったが，ssh では公開鍵暗号を用いて，共通鍵を暗号化することで鍵交換を行うことで，安全かつ高速な通信が可能である．

4.3 実験手順

本課題は以下の手順に則って行う．ここで，AWS のサーバへはアクセス不可能であったため，著者の個人的な ssh 接続を行うことで実験の代わりとする．接続時の手順を以下に示す．

1. 学内 VPN に接続する．
2. Windows PowerShell を開く．
3. コマンドラインに「ssh [username]@192.168.168.81」と入力する．
4. パスワードを入力する．

4.4 実験結果

事件結果のログをソースコード 9 に示す．

Source Code 9: ssh 接続結果

```
PS C:\Users\zigza> ssh ********@192.168.168.81
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.19.0-45-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

Expanded Security Maintenance for Applications is not enabled.

140 のアップデートはすぐに適用されます。
これらの更新の 103 は、標準のセキュリティ更新です。
これらの追加アップデートを確認するには次を実行してください: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

*** System restart required ***
Last login: Thu Dec  7 20:20:54 2023 from 192.168.113.171
********@DLBox-114-28:~$
```

4.5 考察

手順通りの操作によって，ssh 接続が完了し，ユーザー，PC 名がサーバのものに変化していることが確認できる．

5　課題 8

追加された作業用ユーザに sudo 権限が適切に付与されているか確認を行い，SSH における公開鍵・秘密鍵を用いた認証方式での SSH 接続を確立する実験を行う．実験の目的，理論＜方法論＞，実験手順，実験結果，考察を述べよ．

5.1　目的

sudo 権限を確認し，サーバのより専門的な操作が可能であるかを確かめる．

5.2　理論

sudo とは，UNIX 系の OS で用いられるコマンドの 1 つで，ユーザーが別のユーザーの権限レベルでプログラムを実行することができるコマンドである．一般的にはスーパーユーザー (root) の権限を利用するときに用いる．Ubuntu では，スーパーユーザーの完全な代替として用いられるため，ソフトウェアのインストールやシステムアップデート，ファイルの所有権変更などの専門的な操作の実行に必要である．

5.3　実験手順

以下のコマンドを実行することで sudo 権限の付与及び確認が可能である．

```
sudo usermod -aG sudo username                    //sudo 権限の付与
sudo -l                                           //sudo 権限の確認
```

また，以下のコマンドによって ssh の公開鍵，秘密鍵の設定，登録が可能である．

1. ubuntu を開き，「ssh-keygen」と入力する．
2. enter を押すことで鍵のペアを.ssh ディレクトリに保存する．
3.「ssh-copy-id [username]@192.168.168.81」によってサーバに公開鍵をコピーする．
4. パスワードを入力する．

5.4　実験結果

コマンド実行結果をソースコード 10 に示す．

Source Code 10: sudo 権限の確認

```
********@DLBox-114-28:~$ sudo usermod -aG sudo ********
********@DLBox-114-28:~$ sudo -l
既定値のエントリと照合中（ユーザー名 ********）（ホスト名
    DLBox-114-28）:
    env_reset, mail_badpass,

    ↪  secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin,
    use_pty
```

```
ユーザー ******** は DLBox-114-28 上で コマンドを実行できます
    (ALL : ALL) ALL
********@DLBox-114-28:~$
```

ssh の鍵生成及び受け渡しまでの実行結果をソースコード 11 に示す.

Source Code 11: ssh 鍵生成

```
masaki@DESKTOP-H24FTFV:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/masaki/.ssh/id_rsa):
Created directory '/home/masaki/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/masaki/.ssh/id_rsa
Your public key has been saved in /home/masaki/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:kAjWFkV2TR4kCUo8LuD2Luu8pQcRcaBrCsywXuDeJ+c masaki@DESKTOP-H24FTFV
The key's randomart image is:
+---[RSA 3072]----+
| o=+o+=.o=+      |
|oo.o=+ o.o..     |
|+o.oo.o   .      |
|=*o . .          |
|==oo    S        |
|*.o.             |
|.oo+ o           |
|..oo=            |
|.*=  E           |
+----[SHA256]-----+
masaki@DESKTOP-H24FTFV:~$ ssh-copy-id ********@192.168.168.81
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
 ↪  "/home/masaki/.ssh/id_rsa.pub"
The authenticity of host '192.168.168.81 (192.168.168.81)' can't be established.
ED25519 key fingerprint is SHA256:ini9amU7AeEtXfgKcxrOlR/pixsj9goiOJy5kTo9ClI.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out
 ↪  any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted
 ↪  now it is to install the new keys
********@192.168.168.81's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh '********@192.168.168.81'"
and check to make sure that only the key(s) you wanted were added.

masaki@DESKTOP-H24FTFV:~$ ssh ********@192.168.168.81
```

```
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.19.0-45-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

Expanded Security Maintenance for Applications is not enabled.

140 のアップデートはすぐに適用されます。
これらの更新の 103 は、標準のセキュリティ更新です。
これらの追加アップデートを確認するには次を実行してください: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

*** System restart required ***
Last login: Mon Dec 18 22:18:35 2023 from 192.168.113.171
********@DLBox-114-28:~$
```

5.5 考察

手順通りの操作によって，sudo 権限が与えられていることが確認できる．また，ssh の鍵生成及び受け渡しも正常に実行できたことが，ssh の接続時にパスワードを求められないことから確認できる．

6 課題 9

Docker 上のコンテナへの接続・Docker 上での操作を確認する実験を行い，実験の目的，理論＜方法論＞，実験手順，実験結果，考察を述べよ．

6.1 目的

本課題では，Docker を用いてアクセス可能なサーバを構築するため，Docker 環境を準備する．

6.2 理論

Docker とは，コンテナ型の仮想環境を作成，配布，実行するためのプラットフォームである．Docker は Linux のコンテナ技術が基になっており，表面上は VirtualBox などの仮想マシンと似た挙動であるが，Docker によって構築されるコンテナはホストマシンのカーネルを用いており，プロセス，ユーザを隔離することで仮想環境を構築するため，軽量で，起動，停止が高速であるという利点を持つ．また，Docker はミドルウェア（ソフトウェア）のインストール状態，各種環境設定をコード化して管理するため，これらを公開，共有可能であり，各自の環境のバージョンずれ防止や，開発環境準備の短縮化が望める．

6.3 実験手順

まず，Docker をホスト PC にインストールする．

```
sudo add-apt-repository universe
```

Docker で構築するコンテナは，基本的に目的に沿った image を Docker hub 上で探す．今回は Wordpress の image を用いる．image はホスト PC に pull することで使用可能となる．以下のコマンドによってホストに Wordpress の image を保存する．

```
docker pull Wordpress
```

保存した image を基にして，Docker コンテナを構築する．Docker コンテナ構築時のコマンドは，今回は以下のようになる．

```
docker run -dit --name ep20050 -p 20050:80 -v ./server:/var/www/html Wordpress
```

コマンドの内容について解説する．Docker run は Docker コンテナの構築に用いるコマンドである．このコマンドに続く単語郡は構築時の設定である．-dit はコンテナがバックグラウンドで動作させる，ユーザによる入力を可能にする，端末によってコンテナが操作できるようにする設定である．–name はコンテナ名， -p はポート設定であり，それぞれホストのポートとコンテナのポートが対応する．コンテナ上のポートが 80 なのは http の標準ポートであるためである．また，ホスト側のポートは実験者の学籍番号である 20050 を用いる．-v は，コンテナ内のデータを保存する設定である．これを用いることで，データをコンテナ外に保存することができるため，コンテナを消してもデータを保存できる．今回はホスト PC の/server ディレクトリ以下を Docker コンテナの/var/www/html に連動させ，操作可能としている．このとき，ホスト，コンテナ両方からディレクトリにアクセスすることができる．

6.4 実験結果

docker のイメージのダウンロードからコンテナの構築までの流れをソースコード 12 に示す．

Source Code 12: Docker コンテナ構築

```
masaki@DESKTOP-H24FTFV:~$ docker pull wordpress
Using default tag: latest
latest: Pulling from library/wordpress
Digest: sha256:eebf33e0f8b7a7c82e2093404c8e6a9299672d17fdbcc9211c7e9a902468586b
Status: Image is up to date for wordpress:latest
docker.io/library/wordpress:latest
masaki@DESKTOP-H24FTFV:~$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
wordpress    latest    4e6f8f468c68   11 days ago    739MB
python       3.11.3    0a6cd0db41a4   6 months ago   920MB
```

```
masaki@DESKTOP-H24FTFV:~$ ls
jikkenii  server  vscode-cpptools
masaki@DESKTOP-H24FTFV:~$ docker run -dit --name ep20050 -p 20050:80 -v
↪  ./server:/var/www/html wordpress
6ed4fecb4fc6939ffd23becbdae6574af3a5b16dc9b0a5209442ce2f94eeb782
masaki@DESKTOP-H24FTFV:~$ docker ps -a
CONTAINER ID   IMAGE       COMMAND                  CREATED          STATUS
↪  PORTS                    NAMES
6ed4fecb4fc6   wordpress   "docker-entrypoint.s…"   12 seconds ago   Up 9 seconds
↪  0.0.0.0:20050->80/tcp    ep20050
masaki@DESKTOP-H24FTFV:~$
```

6.5 考察

docker images コマンドによってイメージ Wordpress が入手できたことを確認できた．また，docker ps -a コマンドによって新規コンテナ ep20050 が実行されていることを確認できた．ここで，一度 connect: permission denied というエラーが発生した．これは，Docker のグループにユーザ-が追加されていないために起こる現象であるため，「sudo usermod -aG docker $USER」によって現在のユーザーを docker グループに追加している．

7 課題 10

Docker 上に Apache をインストールし，必要なサーバのポートならびにホストのポートのバインドを施し，Http サーバの動作を確認する実験を行い，実験の目的，理論＜方法論＞，実験手順，実験結果，考察を述べよ．

7.1 目的

Apache をインストールすることで，Docker 上に http サーバ機能を実装し，その動作を確認する．

7.2 理論

Apache とは，Web サーバソフトの一種であり，正式名称は「Apache HTTP Server」である．Web サーバとは Web ページを表示するためのデータを補完するサーバのことを指す．名前の通り http 関連のデータ管理に優れ，アプリケーションサーバに特化したものに「Apache Tomcat」がある．2022 年 6 月時点で，Apache は全ウェブサイトの 31.4 %で使用されており，Nginx に次いで 2 番目に人気のあるウェブサーバである．

7.3 実験手順

Apache のインストール及び Http サーバの動作確認は以下の操作で可能である．

1. sudo ufw enable によってファイアウォールを有効化する．
2. sudo ufw allow 20050 によってポートを開放する．
3. sudo ufw reload によってファイアウォールを更新し，変更点を反映する．
4. docker exec -it ep20050 /bin/bash によって課題 9 で作成したコンテナにアタッチする．
5. コンテナ内で apt install systemctl, ufw, Apache によってアプリケーションをインストールする．
6. サーバ側のポート番号 80 を手順 1 から 3 までの手順と同様にして開放する．
7. systemctl start apache2 によって Apache2 を起動する．
8. systemctl status apache2 によって Apache2 の状態を確認する．

7.4 実験結果

実験手順を実行し，Apache の起動及び動作確認を行った結果をソースコード 13 に示す．

Source Code 13: Apache の起動

```
root@35e22d5e3fa6:/var/www# systemctl start apache2
root@35e22d5e3fa6:/var/www# systemctl status apache2
apache2.service - The Apache HTTP Server
    Loaded: loaded (/usr/lib/systemd/system/apache2.service, enabled)
    Active: active (running)
root@35e22d5e3fa6:/var/www#
```

7.5 考察

systemctl status apache2 の結果から，Apache2 が正常にインストールできたことが確認できる．また，ufw の起動時に以下のエラーが発生した．

```
Problem loading ipv6 (skipping)
Problem running '/etc/ufw/before.rules'
Problem running '/etc/ufw/after.rules'
Problem running '/etc/ufw/user.rules'
```

このエラーを解決するため，Docker コンテナを以下の設定でもう一度構築した．

```
docker run -dit --name ep20050 -p 20050:80 --cap-add=NET_ADMIN -v
↪  ./server:/var/www/html Wordpress
```

既存の Docker 構築設定に新しく–cap-add=NET_ADMIN を追加することで，ネットワーク管理機能を追加している．

8　課題 11

Docker 上に PHP をインストールし，PHP の動作を確認する実験を行い，実験の目的，理論＜方法論＞，実験手順，実験結果，考察を述べよ．

8.1　目的

PHP をインストールすることで，動的な web ページを実装可能にする．

8.2　理論

PHP とは正式名称「PHP: Hypertext Preprocessor」であり，動的なコンテンツの作成が可能なオープンソースの汎用スクリプト言語である．PHP を用いることでコードがサーバーで実行され，結果がクライアントに送信されるため，JavaScript と違ってクライアントがコードの内容を知ることはできないという利点がある．今回は，コンテナ内で PHP をインストールし，index.php によって php が実装されているか確認する．

8.3　実験手順

PHP の実装及び PHP の動作確認は以下の操作で可能である．

1. PHP は Docker image Wordpress に導入済みである．「php -v」によってバージョン確認が可能である．
2. もし未インストールであれば， apt install php libapache2-mod-php php-mysql によってインストールを行う．
3. /var/www/html/index.php を編集し，2 行目に phpinfo(); を挿入する．
4. http://localhost:20050 にアクセスし，PHP の動作確認を行う．

8.4　実験結果

PHP の動作確認を行った結果を図 5 に示す．

図 5: http://localhost:20050 へのアクセス結果

8.5 考察

図 5 から，PHP の実装が正常に完了し，phpinfo(); によって PHP の情報が画面上に表示されたことを確認できる．

9 課題 12

Docker 上に MariaDB をインストールし，その動作を確認する実験を行い，実験の目的，理論＜方法論＞，実験手順，実験結果，考察を述べよ．

9.1 目的

Docker コンテナ内に MariaDB をインストールし，動作確認を行うことで，データベース管理機能を実装する．

9.2 理論

MariaDB は，主要な Linux ディストリビューションで採用されているデータベースである．MySQL5.5 から派生したデータベースであり，データを木構造を用いて保存する，階層型の形式を取っている．この形式と対照的にデータを表形式として保存するリレーショナル型があり，SQL などが該当する．本課題では，Docker コンテナ上に MariaDB をインストールし，データベース機能を Web サーバに実装する流れを理解する．

9.3 実験手順

MariaDB のインストールは以下の手順で実行可能である．

1. apt install mariadb-server mariadb-client によって MariaDB サーバ，クライアントのインストールを行う．
2. mysqld_safe & によって MariaDB を直接起動する．
3. mysql_secure_installation によってセキュリティ関連の設定を行う．
4. mysql によって MariaDB に接続する．

MariaDB にログイン後，以下のコマンドを実行することでデータベース及びユーザの作成を行う．

```
CREATE DATABASE EP20050;
CREATE USER ep20050@'localhost' IDENTIFIED BY '20050';
GRANT ALL PRIVILEGES ON EP20050.* to ep20050@'localhost';
FLUSH PRIVILEGES;
```

9.4 実験結果

MariaDB の接続確認結果をソースコード 14 に示す．

Source Code 14: MariaDB の接続確認

```
root@35e22d5e3fa6:/var/www# mysql
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 12
Server version: 10.11.4-MariaDB-1~deb12u1 Debian 12

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE EP20050;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]>  CREATE USER ep20050@'localhost' IDENTIFIED BY '20050';
Query OK, 0 rows affected (0.008 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON EP20050.* to ep20050@'localhost';
Query OK, 0 rows affected (0.006 sec)

MariaDB [(none)]>  FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> exit
Bye
root@35e22d5e3fa6:/var/www#
```

9.5 考察

ソースコード 14 から，正常にデータベースを作成できたことが確認できる．今回の MariaDB の実装において，単純に mysql_secure_installation を起動した場合，

```
ERROR 2002 (HY000): Can't connect to local server through socket
↪  '/run/mysqld/mysqld.sock' (2)
```

というエラーが発生し，MariaDB の起動においても，

```
root@35e22d5e3fa6:/var# service mysql status
mysql: unrecognized service
```

というエラーが発生したため，mysqld_safe & を用いて直接起動した．

10 課題 13

Docker 上に Wordpress をインストールし，その動作を確認する実験を行い，実験の目的，理論＜方法論＞，実験手順，実験結果，考察を述べよ．

10.1 目的

Docker コンテナ上に Wordpress をインストールし，起動確認を行うことで CMS（コンテンツマネジメントシステム）の理解を深める．

10.2 理論

Wordpress とは，PHP で開発されているオープンソースのブログウェアソフトである．データベースには MySQL が用いられる．世界中の Web サイトで利用されており，2023 年 6 月時点，全ウェブサイトの 43.1%で使用されている．PHP による動的なページ作成が可能である点，テンプレートが豊富な点，プラグインを導入できる点などの特徴があり，誰でも簡単に Web ページを作成可能である．本課題では，Wordpress の導入と動作確認を行う．

10.3 実験手順

Docker コンテナを作成する際，Wordpress のイメージを選んでいるため，Wordpress はインストール済みである．そのため，Wordpress の初期設定から行う．以下の手順によって Wordpress の初期設定を行う．

1. MariaDB におけるユーザー名，パスワード，データベース名を記録する．
2. /var/www/html/wp-config-sample.php をコピーし，同ディレクトリに wp-config.php として保存する．
3. wp-config.php に MariaDB のデータベース情報を記入する．このとき，DB_HOST は 127.0.0.1 である．
4. http://localhost:20050 に接続し，必要事項を記入してユーザー登録を行う．
5. ログインすると，Wordpress の初期画面が確認できる．

このときの wp-config.php の内容（一部抜粋）をソースコード 15 に示す．

Source Code 15: wp-config.php

```php
<?php
/**
 * The base configuration for WordPress
 *
 * The wp-config.php creation script uses this file during the installation.
 * You don't have to use the web site, you can copy this file to "wp-config.php"
 * and fill in the values.
 *
 * This file contains the following configurations:
 *
 * * Database settings
 * * Secret keys
 * * Database table prefix
 * * ABSPATH
 *
 * @link https://wordpress.org/documentation/article/editing-wp-config-php/
 *
 * @package WordPress
 */

// ** Database settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'EP20050'); //Change

/** Database username */
define('DB_USER', 'ep20050'); //Change

/** Database password */
define('DB_PASSWORD', '20050'); //Change

/** Database hostname */
define('DB_HOST', '127.0.0.1'); //Change

/** Database charset to use in creating database tables. */
define('DB_CHARSET', 'utf8');

/** The database collate type. Don't change this if in doubt. */
define('DB_COLLATE', '');
```

10.4　実験結果

最初に http://localhost:20050 に接続した結果を図 6 に，ログイン情報を記入して Wordpress の初期画面に遷移した結果を図 7 に示す．

図 6: 初回接続結果

図 7: Wordpress

10.5　考察

wp-config.php では，MariaDB のデータベース EP20050 を Wordpress に用いるという設定を行っている．そのため，MariaDB で定めたユーザー名およびパスワード，接続先アドレスを使用する必要がある．このとき，接続先アドレスに *localhost* を用いると，”Error establishing a database connection”とだけ表示され，データベースへの接続が失敗する．127.0.0.1 で成功する理由は，*localhost* と指定した場合，システムが自身を指定する形式となっているため，UNIX ソケットを介して接続を試みるが，127.0.0.1 は IPv4 ローカルホストアドレスであり，TCP/IP を介してデータベースに接続するためである．

11　課題 14

学籍番号のポートで Wordpress を公開する実験を行い，実験の目的，理論＜方法論＞，実験手順，実験結果，考察を述べよ．

11.1　目的

学籍番号のポートで Wordpress を公開することで，よりポートについての理解を深める．

11.2　理論

課題 13 で作成した Wordpress 環境を流用する．docker コンテナ構築時に指定したポートを用いることで，外部からアクセスすることが可能である．よって，新規に操作を行う必要はない．

11.3　実験手順

web ブラウジングアプリを用いて，http://localhost:20050 にアクセスする．

11.4　実験結果

アクセス結果を図 8 に示す．

図 8:　Wordpress

11.5 考察

図 8 のアドレスバーから，ポート 20050 によってアクセスできていることが確認できる．これは，サーバ側の http のポート 80 とホストのポート 20050 を対応させていたため実現していると考察できる．

12 課題 15

設定を施した Docker コンテナについてイメージ化を行い，他のサーバ環境で構築した環境を復元する実験を行い，実験の目的，理論＜方法論＞，実験手順，実験結果，考察を述べよ．

12.1 目的

今回実験を行った Docker コンテナについてイメージ化を行い，他の PC によって環境を復元する．この操作により，同様の環境をより簡単に他の端末に移行することができるか確認する．

12.2 理論

チームで開発を行うときや，自分の開発したプログラムを他者に共有するとき，端末間でのアプリのバージョンの違いや，システム構成によって動作しない場合がある．

これを解決するため，Docker を用いた仮想環境の構成をイメージとして保存し，他者に共有することができる．このときのシステム構成はコードとして記録され，環境変数などの設定，ファイルシステム，インストールされるアプリケーションを共有することができる．

しかし，ボリュームを用いて Docker コンテナ外のデータを利用している場合，その部分のデータはイメージ化できないため，注意が必要である．今回の課題では，ボリュームを用いているため，ボリューム先は個別にコピーするものとする．

12.3 実験手順

具体的な手順を以下に示す．

1. 課題 9 によって実行した Docker コンテナを用いる．
2. docker commit コマンド用いてコンテナをイメージ化する．
3. docker images コマンドによってコンテナのイメージ化が正常に完了したか確認する．
4. docker save コマンドを用いてイメージを.tar ファイルに圧縮する．
5. 他端末にファイルを共有する．
6. docker load -i を用いて共有ファイルをイメージ化する．
7. docker images コマンドによってイメージが正常に認識されているか確認する．
8. $docker run -dit --name new\_ep20050 -p 20050:80 -v ./server:/var/www/html ep20050$ を入力し，コンテナを起動する．

9. docker コンテナ内で $mysqld_safe\&$ を入力し，データベースを起動する．
10. http://localhost:20050 にアクセスし，コンテナが正常に実行されているか確認する．

12.4 実験結果

docker コンテナのイメージ化，別端末での実行結果をソースコード 16 に示す．

Source Code 16: docker コンテナの移行

```
masaki@DESKTOP-H24FTFV:~/server$ docker commit 35e22d5e3fa6 ep20050
sha256:1b7725567d529b8f8dd617c21a0b304ebd52e0be5f97bb657e66ce2077bf60b1
masaki@DESKTOP-H24FTFV:~/server$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
ep20050      latest    1b7725567d52   23 seconds ago   1.57GB
wordpress    latest    4e6f8f468c68   3 weeks ago      739MB
python       3.11.3    0a6cd0db41a4   7 months ago     920MB
masaki@DESKTOP-H24FTFV:~/server$ docker save -o ep20050.tar ep20050:latest
masaki@DESKTOP-H24FTFV:~/server$ ls
db-test.php  index.php    readme.html      wp-admin        wp-comments-post.php
↪  wp-config-sample.php  wp-content   wp-includes    wp-load.php   wp-mail.php
↪  wp-signup.php     xmlrpc.php
ep20050.tar  license.txt  wp-activate.php  wp-blog-header.php  wp-config-docker.php
↪  wp-config.php     wp-cron.php  wp-links-opml.php  wp-login.php  wp-settings.php
↪  wp-trackback.php
masaki@DESKTOP-H24FTFV:~/server$

--------------------------------------------------------------------------------
                                他端末へ移行
--------------------------------------------------------------------------------

masaki@laptop-lenovo:~/server$ docker load -i ep20050.tar
9f80ba795da1: Loading layer [==================================================>]
↪  51.23MB/51.23MB
b69431a68981: Loading layer [==================================================>]
↪  10.75kB/10.75kB
b897cf086a42: Loading layer [==================================================>]
↪  8.192kB/8.192kB
3cb98ef87e89: Loading layer [==================================================>]
↪  13.39MB/13.39MB
c748a7d1b3bf: Loading layer [==================================================>]
↪  4.096kB/4.096kB
186da2a860bc: Loading layer [==================================================>]
↪  49.53MB/49.53MB
3b0a8c0fae0c: Loading layer [==================================================>]
↪  12.8kB/12.8kB
fe7e6ae67a95: Loading layer [==================================================>]
↪  4.608kB/4.608kB
8d7421982230: Loading layer [==================================================>]
↪  4.608kB/4.608kB
9930490430fd: Loading layer [==================================================>]
↪  69.66MB/69.66MB
```

```
83785679e5f3: Loading layer [==================================================>]  99.61MB/99.61MB
90c5bb9656b5: Loading layer [==================================================>]  5.632kB/5.632kB
f46cd08c0025: Loading layer [==================================================>]  4.608kB/4.608kB
b43c32a62d13: Loading layer [==================================================>]  91.65kB/91.65kB
0ee05dab08d0: Loading layer [==================================================>]  72.08MB/72.08MB
e56fc25040b8: Loading layer [==================================================>]  8.704kB/8.704kB
d9a83783bb44: Loading layer [==================================================>]  6.656kB/6.656kB
010ccac097de: Loading layer [==================================================>]  831.3MB/831.3MB
Loaded image: ep20050:latest
masaki@laptop-lenovo:~/server$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED             SIZE
ep20050      latest    1b7725567d52   About an hour ago   1.57GB
wordpress    latest    7f432c511dec   3 weeks ago         740MB
ubuntu       latest    b6548eacb063   4 weeks ago         77.8MB
masaki@laptop-lenovo:~/server$ docker run -dit --name ep20050 -p 20050:80 -v ./server:/var/www/html ep20050
9adf78e833daac95adfe9848c7a456bb346abadd1c6be8d3d5b3e15203a2393c
masaki@laptop-lenovo:~/server$ docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS
   NAMES
9adf78e833da   ep20050   "docker-entrypoint.s…"   6 seconds ago   Up 4 seconds   0.0.0.0:20050->80/tcp, :::20050->80/tcp   ep20050
masaki@laptop-lenovo:~/server$

--------------------------------------------------------------------------------
                                コンテナ内
--------------------------------------------------------------------------------
root@9adf78e833da:~# mysqld_safe &
[1] 591
root@9adf78e833da:~# 231229 16:02:06 mysqld_safe Logging to syslog.
231229 16:02:06 mysqld_safe Starting mariadbd daemon with databases from /var/lib/mysql
```

http://localhost:20050 にアクセスした結果を図 9 に示す.

図 9: アクセス結果

12.5 考察

ソースコード 16 によって docker commit 及び別端末での docker コンテナの実行が正常に完了したことを確認できる．また，図 9 から，wordpress のアクセスも正常に完了したことを確認できる．これらの操作を他のプロジェクトに利用することで，より簡単に環境の共有が可能であると言える．

13 課題 16

Docker を利用して独自のアプリケーションサーバを構築し，その動作を確認する実験を行い，実験の目的，理論＜方法論＞，実験手順，実験結果，考察を述べよ．

13.1 目的

今までの課題によって作成した wordpress サーバーを用いて，独自のアプリケーションサーバーを構築し，動作確認を行う．

13.2 理論

本課題では，wordpress を用いてイベント予約フォームを作成する．wordpress 内でプラグイン「Events Manager」をインストールし，外観テーマとして Cocoon に設定する．テーマは「テーマを追加」ボタンからインストール可能であり，ローカルからテーマをアップロードすることも可能である．また，「プラグイン」タブから「プラグインを追加」を選択することでプラグインを追加できる．ここから「Events Manager」をインストールして，イベントの予約システムを構築する．

13.3 実験手順

具体的な手順を以下に示す．

1. wordpress サーバーにアクセスする．
2.「外観」メニューから「新しいテーマを追加」を選択する．
3.「テーマのアップロード」から，ローカルフォルダに保存してある「Cocoon」をアッ

プロードする．

4. 「テーマ」メニューから，「Cocoon Child」を選択し，有効化する．
5. 「イベント」メニューから，「イベントの新規作成」を選択する．
6. 必要事項を記入し，「公開」から「更新」をクリックする．
7. 公開されたイベントページにアクセスし，予約が可能か確認する．

13.4 実験結果

作成したイベント予約ページを図 10 に示す．

図 10: 予約ページ

「予約」ボタンを押した結果を図 11 に示す．

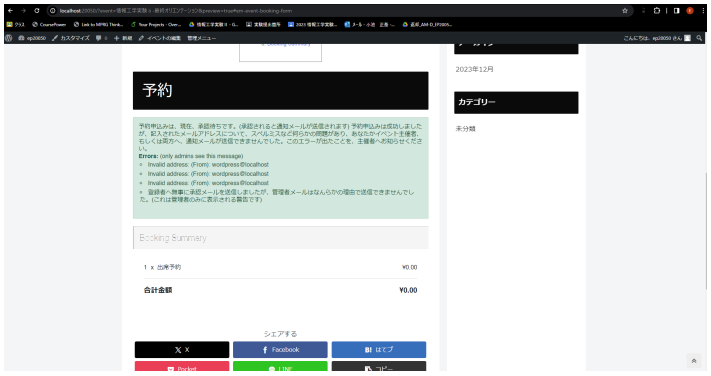

図 11: 予約完了画面

13.5 考察

図 10，図 11 から，イベント予約ページが完成し，予約が可能であることを確認した．また，テーマ「Cocoon」が正常に適用されていることも確認できた．テーマのアップロードの途中で，アップロードの容量制限問題が発生した．これを解決するため，wp-config.php を編集し，以下のコードを追加した．

```
@ini_set('upload_max_size' , '64M' );
@ini_set('post_max_size', '64M');
@ini_set('max_execution_time', '300');
```

このコードを追加することで，ファイルのアップロード制限が 2MB から 64MB に拡張され，テーマのアップロードが可能となった．これらの操作によって，設定されたイベントに対して参加するためのフォームを作成することができた．しかし，この参加フォームではまだ金銭支払いの機能が未実装であり，まだまだ未完成であると言える．今後はこのような追加機能を追加できるよう勉強し，実装していきたい．

参考文献

[1] Windows 10 で vpn 接続設定を行う方法と注意点. https://atmarkit.itmedia.co.jp/ait/articles/2108/26/news041.html, 2021. accessed: 2023-12-10.

[2] Windows10 での vpn 接続方法について. https://faq.interlink.or.jp/faq2/View/wcDisplayContent.aspx?sso_step=1&id=604. accessed: 2023-12-10.

[3] Powershell で vpn 設定を追加してみた. https://tech.yamatozaitaku.com/2020/05/powershell-vpn/, 2020. accessed: 2023-12-10.