つながる工場テストベッド事業

サーバー設定手順書

令和５年７月１日

国立研究開発法人産業技術総合研究所

目次

1. 概要

本手順書は、「つながる工場テストベッド」事業で用いるサーバー設定の手順を示す。本手順書では、実行コマンドおよびコンソール出力・ファイルの内容は、それぞれ以下のスタイルで表現する。

(1) 実行コマンドのスタイル

```
# 実行するコマンド（root（管理者）権限で実行する場合）
$ 実行するコマンド（一般ユーザー権限で実行する場合）
```

一般ユーザーとしてログインした後でroot権限を得るためには、suコマンドを用いる

```
$ su
パスワード: （root のパスワードを入力）
#
```

(2) コンソール出力またはファイルの内容のスタイル

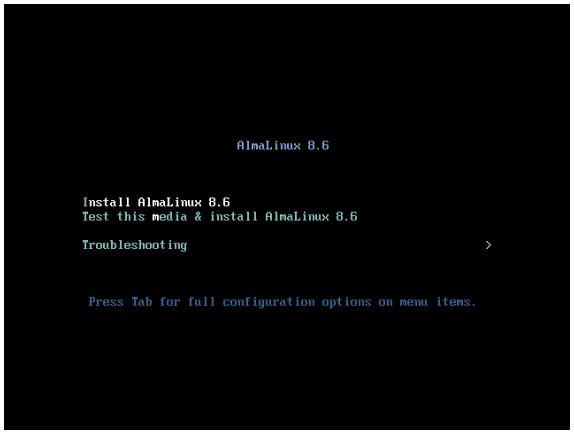
```
ファイルの内容
```

2. AlamLinux 8.6（OS）のインストールと初期設定

インストールDVDよりインストーラを起動し、画面の指示に従う。本サーバーのディスクアレイはソフトウェアRAIDで構成し、ブート領域にはRAID1、スワップ領域には標準パーティション、データ領域にはRAID5を用いる。その設定はインストーラで行う。

2.1. AlmaLinux 8.6のインストール

(1) インストーラ起動



(2) 言語選択

(3) 日付と時刻選択

(4) ソフトウェア選択

サーバー（GUI）を選択。必要なものは後からでも追加できる。取りあえず、開発ツールとシステムツールは選んでおいた方がよさそう。

(5) インストール先選択（RAID設定）

ディスクを3つとも選択し、ストレージの設定では「カスタム」を選択する。「データを暗号化する」を選択すると、起動するたびにパスフレーズの入力を求められることになる。したがって、ここでは選択しない。

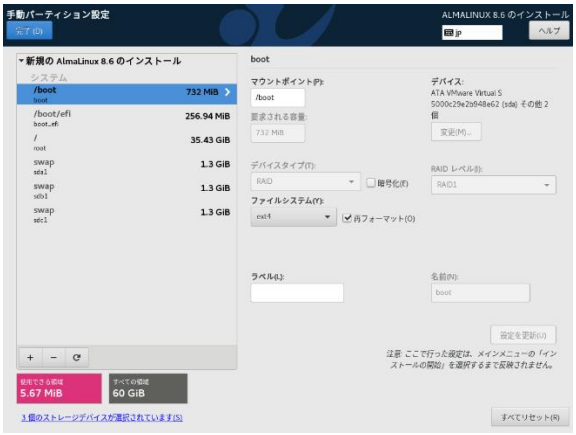
画面左下の「＋」ボタンをクリックして、以下のようにパーティションを作成する。

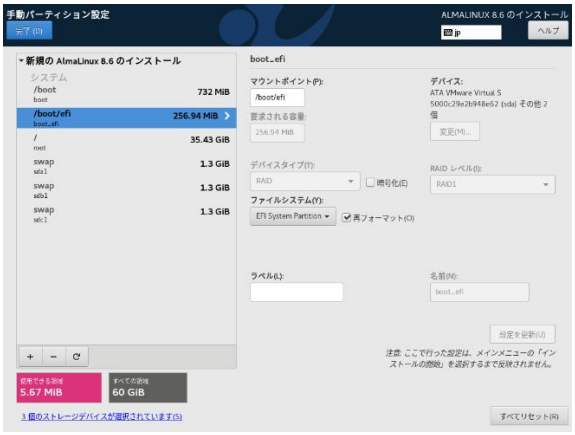
/boot　　768MB, RAID1, ext4

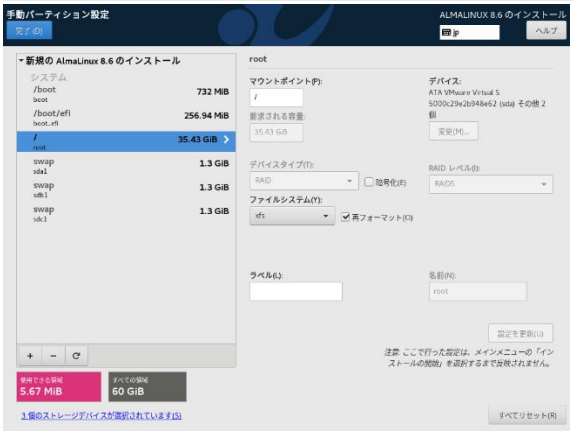
/boot/efi　256MB, RAID1, EFI System Partition

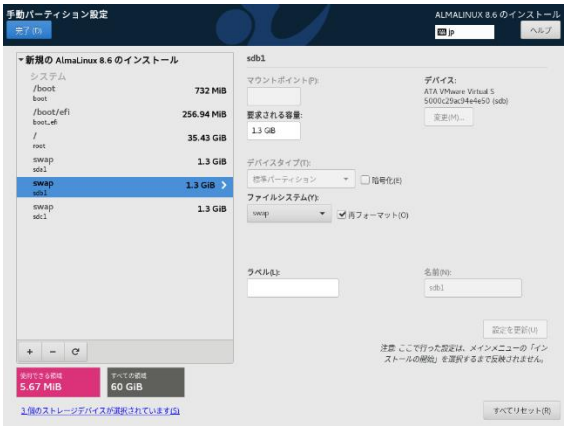
swap　　総計でRAMと同程度になるように各ディスクに設定，標準Partition, swap

/　　残り全部, RAID5, xfs

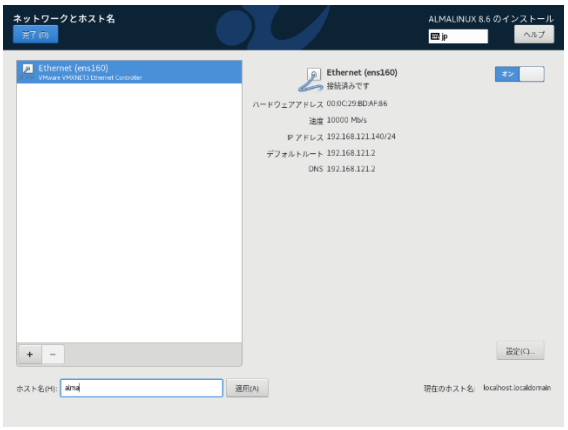




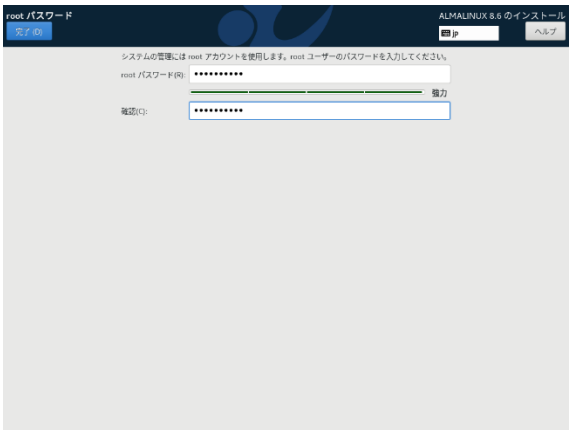


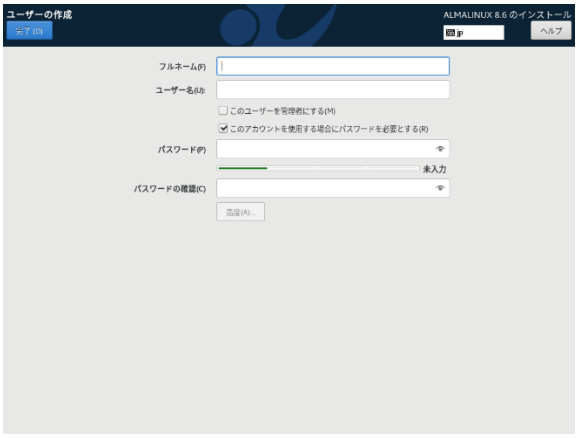


(6) ネットワーク設定
ホスト名は仮置きで構わない。

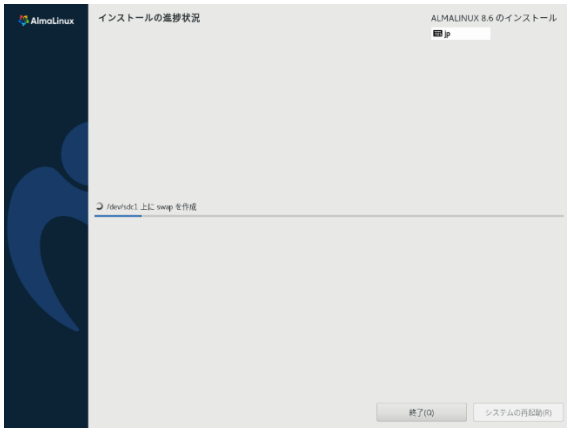


(7) ユーザーの設定

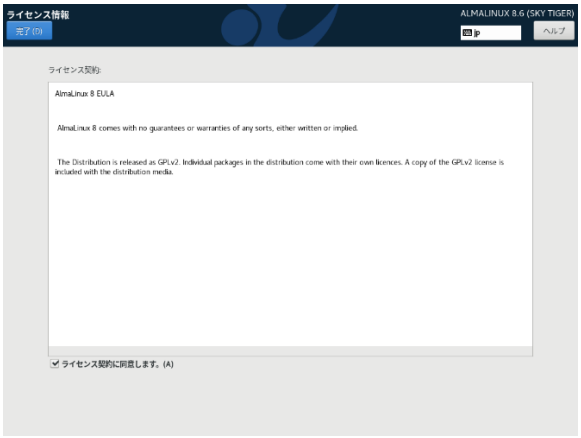




(8) インストール実行とシステム再起動



(9) 初期セットアップ（ライセンス契約同意）



初回のログイン時には、使用言語等の環境設定画面が表示される。画面の指示に従って設定を行う。「日本語（かな漢字）」を選択した場合、Windowsキー＋スペースキー押下で日本語入力を行えるようになる。

2.2. SELinuxの設定

SELinuxは、外部から攻撃された結果、プロセスの乗っ取りやroot権限の奪取が行われた場合においても、そのプロセスをMAC (Mandatory Access Controll)による強制アクセ

ス制御下に置き、行動を制限させることを目的とした安全機能モジュールである。攻撃そのものを防ぐものではない。

アプリケーションによっては動作を制限されることがあり[2]、SELinuxを無効化して運用しているシステムもあるが、安全上、SELinuxは有効化しておくことが望ましい。SELinuxの有効/無効設定は、/etc/sysconfig/selinuxファイルの編集によって行う。

/etc/sysconfig/selinuxファイルの編集

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=enforcing ←この行を書くと SELinux 有効
SELINUX=disabled ←この行を書くと SELinux 無効
# SELINUXTYPE= can take one of three values:
#       targeted - Targeted processes are protected,
#       minimum - Modification of targeted policy. Only selected processes are protected.
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

設定変更を反映させるには、OSを再起動する。

```
# reboot
```

2.3. ファイアウォールの設定

ファイアウォールとは、外部からの不正なアクセスを防ぐために、許可されない外部からの通信を阻止するシステムである。ファイアウォールの設定状態の確認や設定変更は、firewall-cmdコマンドで行う。CentOS 8のファイアウォールは、標準でいくつかの通信を許可している。

ファイアウォールの設定状態表示

```
# firewall-cmd --list-all
public (active)
        :
interfaces: <ネットワークカード名>
        :
services: cockpit dhcpv6-client ssh ←通信が許可されているサービス（常駐機能）
        :
```

通信が許可されているサービス（常駐機能）のうち、cockpitはWebブラウザ経由でサーバーを管理するアプリケーションである。便利な機能ではあるが、外部からの不正な操作を許す恐れもあるため、通信を許可しないように設定する。

---

[2] 例えば、後述のアンチウィルスソフトの常駐プロセスによるスキャンが制限される。

cockpitの通信許可削除

```
# firewall-cmd --remove-service=cockpit --permanent ←cockpit の通信許可の恒久的な削除
# firewall-cmd --reload ←上記変更の反映
```

　また、標準設定ではAllowZoneDriftingという属性が有効になっているが、これはセキュリティ上問題があり、将来のリリースからの削除が予定されている。設定ファイル/etc/firewalld/firewalld.confを編集し、AllowZoneDriftingを無効化する。

/etc/firewalld/firewalld.confの編集によるAllowZoneDriftingの無効化

```
                :
#AllowZoneDrifting=yes ←この行をコメントアウト
AllowZoneDrifting=no ←この行を追加
                :
```

2.4. OpenSSHによるリモート接続設定

　インストール時に設定したユーザーに対するssh (Secure Shell)リモート接続を、公開鍵方式による認証で行うように設定する。

2.4.1. 認証方式設定

　/etc/ssh/sshd_configファイルを編集し、公開鍵認証の有効化とパスワード認証の無効化を行う。

/etc/ssh/sshd_configファイルの編集[3]

```
                :
PermitRootLogin no ←この行の yes を no に変更
                :
PubkeyAuthentication yes ←この行をアンコメント
                :
PasswordAuthentication no ←この行の yes を no に変更
                :
```

　sshサービスを再起動する。これにより、公開鍵方式の認証によってのみ、ssh接続を行えるようになる。すなわち、公開鍵認証を設定しないユーザーに対するssh接続は行えなくなる。また、rootへのリモート接続も行えない。

sshサービスの再起動

```
# systemctl restart sshd
```

[3] OpenSSHを別途手動でインストールした場合、設定ファイルは/usr/local/etc/sshd_configになる。

2.4.2. 公開鍵および秘密鍵の生成

インストール時に設定したユーザーでログインし、ssh-keygenコマンドにより、公開鍵と秘密鍵を生成する。鍵の生成アルゴリズムには、Ed25519[4]を用いる。

公開鍵と秘密鍵の生成

```
$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/<ユーザー名>/.ssh/id_ed25519): ←Enter キー入力
Created directory '/home/<ユーザー名>/.ssh'.
Enter passphrase (empty for no passphrase): ←パスワードの設定
Enter same passphrase again: ←確認のため設定したパスワードを再入力
Your identification has been saved in /home/<ユーザー名>/.ssh/id_ed25519,
Your public key has been saved in /home/<ユーザー名>/.ssh/id_ed25519.pub,
The key fingerprint is:
SHA256: ……
The key's randomart image is:
+--[ED25519 256]--+
|            :            |
+---- [SHA256] -----+
```

生成された公開鍵を、信頼された公開鍵リストへ登録する。

公開鍵の登録とアクセス許可設定

```
$ cd ~/.ssh
$ cat id_ed25519.pub >> authorized_keys
$ chmod 700 .（このディレクトリを所有者のみ読み書き実行可に設定）
$ chmod 600 authorized_keys（authorized_keys を所有者のみ読み書き可に設定）
```

リモート接続を行う場合には、リモート接続を行うPCにid_ed25519（id_ed25519.pubではない方）をコピーしてアクセス許可設定を行った後、sshコマンドを実行する。

秘密鍵ファイル（id_ed25519）のアクセス許可設定（接続を行うPCがLinuxの場合）

```
$ chmod 600 id_ed25519（id_ed25519 を所有者のみ読み書き可に設定）
```

秘密鍵ファイル（id_ed25519）のアクセス許可設定（接続を行うPCがWindowsの場合）

```
> cacls id_ed25519 /G %USERNAME%:F
（id_ed25519 を所有者のみフルコントロール可に設定）
```

リモート接続（sshコマンド実行）

```
$ ssh -i id_ed25519 <接続先ユーザー名>@<サーバー名>
The authenticity of host '<サーバー名>' can't be established.
```

[4] 2022年9月現在、最も強固で効率的なデジタル署名アルゴリズムと言われている。

```
ECDSA key fingerprint is SHA256:NqCnLKNS8EiVpaRHqEzIJrDUzkMOhh3MNen4jN7Cxh4.
ECDSA key fingerprint is MD5:f3:cb:9e:10:90:00:fa:0c:4e:ab:9a:9c:d7:09:95:dd.
Are you sure you want to continue connecting (yes/no)? yes
（初回接続時のみこのメッセージが表示される。yes と書いて Enter キー入力。接続先サーバー情報は、.ssh ディレクトリ内の known_hosts ファイルに追記される。）
```

2.4.3. IPアドレスによる接続許可の設定

インストール時のファイアウォールは、sshによる通信はすべて許可するように設定されている。特定のIPアドレスからのssh接続のみを許可するようにしたい場合には、許可するサービスからいったんsshを削除し、改めてIPアドレス指定での許可設定を行う。

特定IPアドレスからのアクセスのみを許可

```
# firewall-cmd --remove-service=ssh [--permanent]
（許可するサービスから ssh を削除。恒久的に変更する場合は、'--permanent'付与。）
# firewall-cmd --add-rich-rule='rule family=ipv4 source address=<IP アドレス> service name=ssh accept' [--permanent]
（指定 IP アドレスからのみ ssh 接続許可。恒久的に変更する場合は、'--permanent'付与。）
# firewall-cmd --reload（設定を恒久的に反映させる場合は実行）
```

一時的な設定変更ではなく、恒久的な変更とする場合には'--permanent'オプションを付けて、最後に'firewall-cmd --reload'を実行する。

IPアドレスを範囲指定する場合には、"192.168.0.0/16'のようにマスク指定を行う。マスク指定は、2進数表現の場合に一致する桁数を表す。この場合は、2進数表記で左から16桁まで一致するIPアドレス指定となるので、192.168.0.0～192.168.255.255を指定したことになる。

2.5. ソフトウェアパッケージの自動更新設定

AlmaLinuxのソフトウェアパッケージの管理には、dnfというパッケージ管理システムを使用する。まず、ソフトウェアパッケージを最新のものに更新しておく[5]。

ソフトウェアパッケージの更新

```
# dnf -y --nobest update
（-y オプションにより、ソフトウェア更新時の問い合わせにすべて yes と自動応答
  --nobest オプションにより、参照エラーが発生するパッケージをスキップ）
```

次に、以下の手順に従って、dnfによる自動更新設定を行う。

(1) dnf-automaticのインストール

```
# dnf -y install dnf-automatic
（-y オプションにより、インストール時の問い合わせのすべてに yes と自動応答）
```

[5] OSインストール直後の更新には、時間がかかることがある。

(2) 設定ファイル（/etc/dnf/automatic.conf）の編集

```
                    :
apply_updates = yes ←この行の no を yes に変更
                    :
```

(3) タイマーファイル（/usr/lib/systemd/system/dnf-automatic.timer）の編集

```
                    :
[Timer]
OnCalendar=*-*-* 6:00（毎日 6 時頃に更新。必要であれば日時を変更）
RandomizedDelaySec=60m（指定時刻からの遅延時間範囲を指定）
                    :
```

(4) 自動更新サービスの有効化と起動

```
# systemctl enable dnf-automatic.timer
# systemctl start dnf-automatic.timer
```

自動更新の日時をより詳細に指定したい場合には、cron[6]を利用する。

cronによる自動更新設定（dnf-automaticを使わない方法）

```
# systemctl stop dnf-automatic.timer（dnf-automatic.timer サービス停止）
# systemctl disable dnf-automatic.timer（dnf-automatic.timer サービス無効化）
# crontab -e（cron 設定ファイル編集画面(vi エディタ)表示）
（更新コマンドと実行時期を記述）
0 1 * * Sun /usr/bin/dnf -y --nobest update
（0 分 1 時毎週日曜日に/usr/bin/dnf -y --nobest update を実行）
crontab: installing new crontab（編集内容を保存完了メッセージ表示）
# crontab -l（cron 登録内容の表示）
0 1 * * Sun /usr/bin/dnf -y --nobest update
```

cron設定書式は以下の通り（半角スペース区切り）[7]。
1番目: 実行日時の分（上の場合は0分）*/nとすることで、n分間隔という指定も可
2番目: 実行日時の時（上の場合は1時）*/nとすることで、n時間間隔という指定も可
3番目: 実行日時の日（上の場合は指定なし）*/nとすることで、n日間隔という指定も可
4番目: 実行日時の月（上の場合は指定なし）*/nとすることで、n月間隔という指定も可
5番目: 曜日（上の場合は日曜日）数字指定も可（0から6が日曜日から土曜日に対応）
6番目以降: 実行コマンド

dnfにより更新を行った後、OSの再起動が必要となる場合がある。再起動が必要なパッケージは、needs-restartingコマンドで確認できる。

---

[6] Windowsのタスクスケジューラに相当するものと考えて差し支えない。
[7] cron設定情報は、ユーザーごとに分けられたファイルとして、/var/spool/cronディレクトリに保存される。

再起動が必要なパッケージの確認

```
# dnf needs-restarting -r
```

自動更新後、再起動も自動的に行うようにするには、cronで以下のように設定する。

cronによる自動更新と自動再起動設定

```
# crontab -e（cron 設定ファイル編集画面(vi エディタ)表示）
（実行時期と更新コマンドを記述）
0 1 * * Sun /usr/bin/dnf -y --nobest update; /usr/sbin/reboot
（0 分 1 時毎週日曜日に/usr/bin/dnf -y --nobest update を実行して再起動）
crontab: installing new crontab（編集内容を保存完了メッセージ表示）
# crontab -l（cron 登録内容の表示）
0 1 * * Sun /usr/bin/dnf -y --nobest update; /usr/sbin/reboot
```

2.6. 標準実行レベルのCUI（Character-baed User Interface）設定

インストール直後は、GUI（Graphical User Interface）で実行されている。サーバーとして運用する場合、GUIを使う必要はないので、標準の実行レベルをCUIに変更しておく。

標準実行レベルのCUIへの変更

```
# systemctl get-default（現在の実行レベルを表示）
graphical.target
# systemctl set-default multi-user.target（標準実行レベルを CUI に設定）
# systemctl get-default（実行レベルの変更を確認）
multi-user.target
# reboot（再起動による変更の反映）
```

一方、ローカルユーザーとしてログインして操作する場合には、GUIの方が便利である。GUIを起動するときには、ログイン後、startxコマンドを実行する。このとき、言語モードは英語になっていることに注意。日本語モードでGUIを起動するときには、言語モード指定を行ってから起動する。

GUIの起動（日本語モード指定）

```
$ export LANG=ja_JP.utf8;startx
```

ホームディレクトリ内の.bash_profileファイルの末尾に以下の1行を追加しておくと、startxのみの入力で、日本語モードのGUIが起動する。

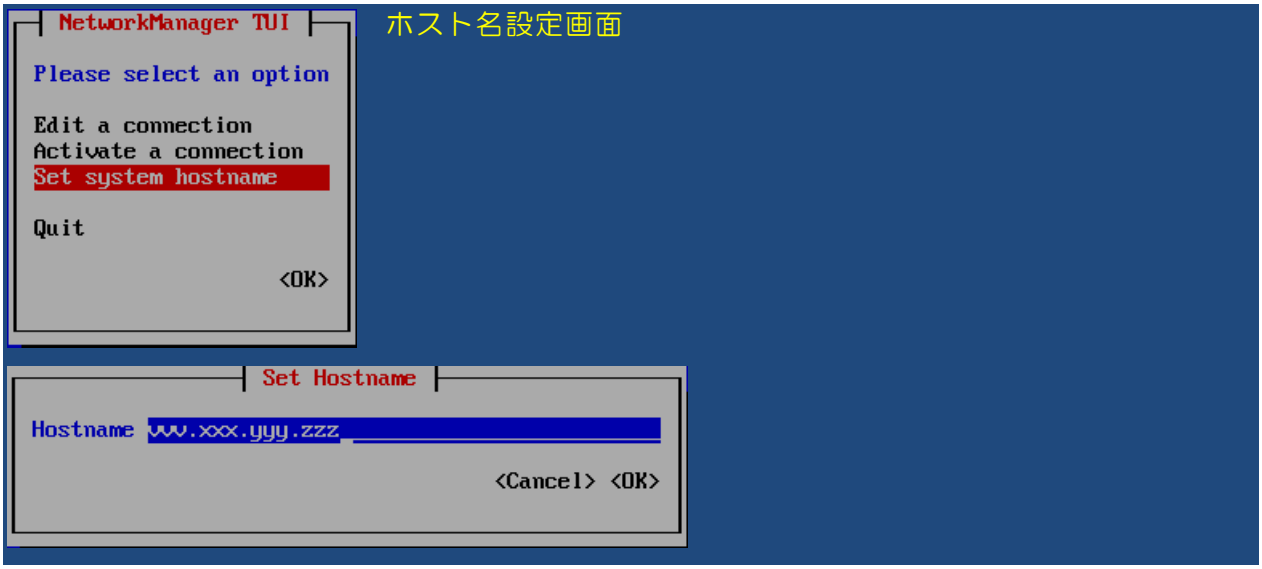
.bash_profileへの記述追加（alias設定）

```
alias startx='export LANG=ja_JP.utf8;startx'
```

2.7. ホスト名およびIPアドレスの変更

サーバーの移設などによりホスト名やIPアドレスを変更する場合には、nmtui（NetworkManager Text User Interface）コマンドを用いて設定を行う。

nmtuiコマンドの起動



3. 各種アプリケーションのインストールと設定

3.1. アンチウィルス（ClamAV）

オープンソースのアンチウィルスソフトClamAVをインストールし、自動更新と定時スキャンの設定を行う。

3.1.1. ClamAVのインストールと起動設定

ClamAVは、標準のリポジトリではなく、EPEL（Extra Packages for Enterprise Linux）リポジトリより提供されている。最初に、EPELを利用先のリポジトリとして登録する。

EPELリポジトリの登録

```
# dnf -y install epel-release
```

次に、ClamAV本体、ウィルススキャンデータベース、データベース更新機能、ClamAVサービスをインストールする。

ClamAV本体、ウィルススキャンデータベース、データベース更新機能、ClamAVサービス（常駐機能）のインストール

```
# dnf -y install clamav clamav-data clamav-update clamd
```

/etc/clamd.d/scan.confファイルを編集し、常駐動作の内容を設定する。

/etc/clamd.d/scan.confファイルの編集

```
                :
#Example ←この行をコメントアウト（例示であることの宣言を除去）
                :
LogFile /var/log/clamd.scan ←この行をアンコメント（ログファイル指定）
                :
```

```
LogFileMaxSize 2M ←この行をアンコメント（ログファイル最大サイズ指定）
                  :
LogTime yes ←この行をアンコメント（ログ時刻記述の要否指定）
                  :
LogRotate yes ←この行をアンコメント（古いログファイルの自動削除・上書要否指定）
                  :
ExtendedDetectionInfo yes ←この行をアンコメント（感染ファイル詳細情報出力要否指定）
                  :
LocalSocket /run/clamd.scan/clamd.sock ←この行をアンコメント（ローカルソケット使用）
                  :
FixStaleSocket yes ←この行をアンコメント（異常終了時のソケット削除可否）
                  :
ExcludePath ^/proc/ ←この行をアンコメント（スキャン対象外ディレクトリ）
ExcludePath ^/sys/ ←この行をアンコメント（スキャン対象外ディレクトリ）
ExcludePath ^/dev/ ←この行を追加（スキャン対象外ディレクトリ）
ExcludePath ^/.*/selinux/ ←この行を追加（スキャン対象外ディレクトリ）
ExcludePath ^/.*/audit/ ←この行を追加（スキャン対象外ディレクトリ）
ExcludePath ^/run/systemd/inaccessible/ ←この行を追加（スキャン対象外ディレクトリ）
ExcludePath ^/run/user/.*/gvfs ←この行を追加（スキャン対象外ディレクトリ）
ExcludePath ^/etc/.*shadow.* ←この行を追加（スキャン対象外ディレクトリ）
ExcludePath ^/etc/security/.*passwd.* ←この行を追加（スキャン対象外ディレクトリ）
ExcludePath ^/root/virus ←この行を追加（スキャン対象外ディレクトリ）
                  :
#SelfCheck 600 ←この行をコメントアウト（ウィルデータベース確認間隔（秒））
SelfCheck 21600 ←この行を追加（ウィルデータベース確認間隔（秒））
                  :
#User clamscan ←この行をコメントアウト（スキャン実行ユーザー）
User root ←この行を追加（スキャン実行ユーザーを root に指定）
                  :
OnAccessIncludePath <ディレクトリパス>
（オンアクセススキャン（リアルタイムスキャン）対象ディレクトリを適宜追記）
                  :
OnAccessExcludePath <ディレクトリパス>
（オンアクセススキャン対象外ディレクトリを適宜追記）
                  :
OnAccessPrevention yes ←この行をコメントアウト（感染ファイルへのアクセス禁止要否）
                  :
#OnAccessExcludeUname clamav ←この行をコメントアウト（スキャン実行ユーザー）
OnAccessExcludeUname root ←この行を追加（オンアクセススキャン対象外ユーザー指定）
```

ClamAVサービスの起動、動作状態確認、自動起動設定を行う。

ClamAVサービスの起動、動作状態確認、自動起動設定

```
# systemctl start clamd@scan（ClamAV スキャンサービスの起動）
```

```
# systemctl status clamd@scan（動作状態確認）
● clamd@scan.service - clamd scanner (scan) daemon
   Loaded: loaded (/usr/lib/systemd/system/clamd@.service; disabled; …
   Active: active (running) since …
        :
# systemctl enable clamd@scan.service（自動起動設定（サービス有効化））
# systemctl start clamonacc（ClamAV オンアクセススキャンサービス起動）
# systemctl status clamonacc（動作状態確認）
● clamonacc.service - Clam AntiVirus userspace daemon for OnAccess Scanning
   Loaded: loaded (/usr/lib/systemd/system/clamonacc.service; disabled; …
   Active: active (running) since …
# systemctl enable clamonacc.service（自動起動設定（サービス有効化））
```

SELinux（5ページ）を有効化している場合には、以下の設定も行っておく。

SELinuxを有効化した場合の追加設定[8]

```
# setsebool -P antivirus_can_scan_system 1
（アンチウィルスソフトにシステムファイルのスキャンを許可）
# setsebool -P clamd_use_jit 1
（ClamAV サービスに実行時コンパイラ（Just-In-Time Compiler）の使用を許可）
# reboot（SELinux への設定変更反映）
```

3.1.2. ウィルススキャンデータベースの更新設定

ウィルススキャンデータベースの更新は、freshclamというプログラムによって行われる。freshclamの動作は、設定ファイル/etc/freshclam.confで指定する。

/etc/freshclam.confの編集

```
                    :
#Example ←この行をコメントアウト（例示であることの宣言を除去）
                    :
UpdateLogFile /var/log/freshclam.log ←この行をアンコメント（ログファイル指定）
                    :
LogFileMaxSize 2M ←この行をアンコメント（ログファイル最大サイズ指定）
                    :
LogTime yes ←この行をアンコメント（ログ時刻記述の要否指定）
                    :
LogRotate yes ←この行をアンコメント（古いログファイルの自動削除・上書要否指定）
                    :
#DatabaseOwner clamupdate ←この行をコメントアウト（データベースの所有者）
DatabaseOwner root ←この行を追加（データベースの所有者を root に指定）
```

[8] この設定を行っても、SELinuxによってウィルススキャンを拒否されるディレクトリは存在し、エラーメッセージが出力される。/etc/clamd.d/scan.confでスキャン対象外に指定した” ^/.*/selinux/”と” ^/.*/audit/”などは、それに当たる。

```
                    :
#Checks 24 ←この行をコメントアウト（データベースの 1 日当たりの更新回数）
Checks 8 ←この行を追加（データベースの 1 日当たりの更新回数）
                    :
#NotifyClamd /path/to/clamd.conf ←この行をコメントアウト（更新通知要否＆通知先指定）
NotifyClamd /etc/clamd.d/scan.conf ←この行を追加（更新を clamd@scan へ通知）
                    :
```

freshclamの起動とサービスの有効化を行う。ウィルススキャンデータベースの更新は定期的に自動で行われ、ClamAVサービスに通知される。

freshclamサービスの有効化と起動

```
# systemctl start clamav-freshclam（freshclam の起動）
# systemctl enable clamav-freshclam（freshclam サービスの有効化）
```

3.1.3. 定時ウィルススキャンの設定

ウィルススキャンと、ウィルス検出時のメール送信を行うスクリプトvirus_scan.shを作成し、定時に実行するように設定する。

(1) ウィルススキャンスクリプトの作成
/rootディレクトリ（rootのホームディレクトリ）に、virus_scan.shを作成する。

/root/virus_scan.shの作成

```
#!/bin/sh

CLAMDSCAN=/usr/bin/clamdscan（ウィルススキャンコマンド）
SCANDIR=/       （スキャン対象ディレクトリ。この場合は全ディレクトリ。）
VIRUS_MVDIR=/root/virus（ウィルス移動先ディレクトリ）
SCANRESULT=/tmp/scan_result.txt（スキャン結果出力先ファイル）
HOSTNAME=`hostname`     （ホスト名）
MAILADDR=<送信先メールアドレス>

$CLAMDSCAN $SCANDIR --move=$VIRUS_MVDIR > $SCANRESULT 2>&1
（スキャン結果を SCANRESULT で指定したファイルに出力）

if [ ! -z "$(grep FOUND$ $SCANRESULT)" ]; then（送信メールのタイトル設定）
        SUBJECT="[Virus Found]"
else
        SUBJECT="[Virus Not Found]"
fi

cat $SCANRESULT | mail -s "$SUBJECT $HOSTNAME" ¥（メール送信）
-S smtp=smtp://<SMTP サーバーホスト名>:<ポート番号> ¥
-S smtp-auth=login ¥
```

```
-S smtp-auth-user=<ユーザー名> ¥
-S smtp-auth-password=<パスワード> ¥
-S smtp-use-starttls ¥（通信保護に STARTTLS を使用する場合は記述）
-S nss-config-dir=/etc/openldap/certs/ ¥
-S ssl-verify=ignore ¥
-S from=<送信元メールアドレス> ¥
$MAILADDR

rm -f $SCANRESULT（スキャン結果ファイルを削除）
```

(2) ウィルス移動先ディレクトリの作成
検知したウィルスの移動先ディレクトリを作成する。

```
# mkdir /root/virus
```

(3) メール送信機能のインストール
メール送信パッケージmailxをインストールする。

```
# dnf -y install mailx
```

(4) ウィルススキャンスクリプトの実行ファイル化
作成したvirus_scan.shには、メールサーバーのユーザー名やパスワードが平文で記載されており、セキュリティ上、好ましくない。shcコマンドを用いて、virus_scan.shを実行ファイル化する。

```
# dnf -y install shc（shc のインストール）
# shc -r -f virus_scan.sh（virus_scan.sh の実行ファイル化）
# rm virus_scan.sh virus_scan.sh.x.c（元のスクリプトと中間ファイルの削除）
```

新しく生成されたvirus_scan.sh.xがウィルススキャンの実行ファイルである。

(5) ウィルススキャンの定時実行設定
cronにより、ウィルススキャンを提示に実行するように設定する。

```
# crontab -e（cron 設定ファイル編集画面(vi エディタ)表示）
                    :
0 3 * * * /root/virus_scan.sh.x（0 分 3 時に毎日/root/virus_scan.sh.x を実行）
crontab: installing new crontab（編集内容を保存完了メッセージ表示）
# crontab -l（cron 登録内容の表示）
          :
0 3 * * * /root/virus_scan.sh.x
```

3.2. 侵入検知システム（tripwire）

ファイルやディレクトリの状態を監視し、ファイルの変更や改竄等の不整合を検知するソフトウェアtripwireをインストールし、定期チェックを行うように設定する。

3.2.1. tripwireのインストールと初期設定

dnfコマンドによりtripwireをインストールし、サイトパスフレーズとローカルパスフレーズという2つのパスワードを設定する。

tripwireのインストールとパスフレーズの設定

```
# dnf -y install tripwire（tripwire のインストール）
# tripwire-setup-keyfiles
        :
Enter the site keyfile passphrase: ←サイトパスフレーズの入力
Verify the site keyfile passphrase: ←サイトパスフレーズの確認
        :
Enter the local keyfile passphrase: ←ローカルパスフレーズの入力
Verify the local keyfile passphrase: ←ローカルパスフレーズの確認
        :
Signing configuration file...
Please enter your site passphrase: ←サイトパスフレーズの入力
Wrote configuration file: /etc/tripwire/tw.cfg（設定ファイルへの署名完了）
        :
Signing policy file...
Please enter your site passphrase: ←サイトパスフレーズの入力
Wrote policy file: /etc/tripwire/tw.pol（ポリシーファイルへの署名完了）
```

3.2.2. 設定ファイルの編集と暗号署名

tripwireの設定ファイル/etc/tripwire/twcfg.txtを必要に応じて編集し、暗号署名を行う。

/etc/tripwire/twcfg.txtの編集

```
                :
REPORTLEVEL =3  ←0～4 の範囲でレポートの詳細レベルを指定（4 が最も詳細）
                :
```

設定ファイルの暗号署名（暗号署名版設定ファイルの生成）

```
# twadmin --create-cfgfile -S /etc/tripwire/site.key /etc/tripwire/twcfg.txt
Please enter your site passphrase: ←サイトパスフレーズを入力
Wrote configuration file: /etc/tripwire/tw.cfg（暗号署名版設定ファイル生成）
```

暗号署名版を作成したら、元のテキスト版設定ファイルは削除しておく。

テキスト版設定ファイルの削除

```
# rm /etc/tripwire/twcfg.txt
```

暗号署名版設定ファイルの内容をテキストファイルへ出力する場合は、twadminコマンドを使う。

暗号署名版設定ファイルのテキストファイル（/etc/tripwire/cfgprint.txt）への出力

```
# twadmin --print-cfgfile > /etc/tripwire/cfgprint.txt
```

tripwireの設定を変更する場合には、このようにして出力したテキストファイルを編集し、暗号署名を行う。

3.2.3. ポリシーファイルの編集と暗号署名

ポリシーファイルは、どのような監視を行うのかといった整合性チェックの内容を記述するものであり、サンプルファイルとして/etc/tripwire/twpol.txtが用意されている。

このサンプルファイルには存在していないファイルやディレクトリも記述されているために、これをそのまま使って以降の設定を行うと、「そのようなファイルやディレクトリはありません」というエラーが多数表示される。

スクリプトfixtwpol.shを作成して実行し、エラー発生原因となる行をtwpol.txtからコメントアウトしたファイルを生成する[9]。

fixtwpol.sh（エラー発生原因となる行をコメントアウトするスクリプト）

```
#!/bin/sh

error=$1.err
output=$1.new
cp $1 $output

tripwire --init 2> $error
paths="`cat $error | grep Filename | sed -r -e "s/[ #]+Filename: +//"`"

for path in $paths
do
        echo $path
        sed -r -e "s;$path([ ¥t]);#$path¥1;" $output > $output~
        cp $output~ $output
done

rm $error $output~
```

fixtwpol.shの実行

```
# chmod u+x fixtwpol.sh（実行権限を設定）
# ./fixtwpol.sh /etc/tripwire/twpol.txt
Please enter your local passphrase: ←ローカルパスフレーズを入力
Parsing policy file: /etc/tripwire/tw.pol
Generating the database...
*** Processing Unix File System ***
The object: "/boot/efi" is on a different file system...ignoring.
Wrote database file: /var/lib/tripwire/<ホスト名>.twd
```

[9] 手作業でtwpol.txtを編集しても良いが、それはかなり面倒な作業になる。

```
The database was successfully generated.
/usr/sbin/fixrmtab（コメントアウトされる行のファイルやディレクトリの表示）
        :
```

fixtwpol.shを実行すると、ポリシーファイル/etc/tripwire/twpol.txt.newが新しく生成される。これを使って、ポリシーファイルの暗号署名を行う。

ポリシーファイルの暗号署名（暗号署名版設定ファイルの生成）

```
# twadmin --create-polfile -S /etc/tripwire/site.key /etc/tripwire/twpol.txt.new
Please enter your site passphrase: ←サイトパスフレーズを入力
Wrote policy file: /etc/tripwire/tw.pol（暗号署名版ポリシーファイル生成）
```

3.2.4. tripwireデータベースの作成と動作確認

tripwireは、ファイルやディレクトリの状態をデータベースに記録しておき、その記録内容と実際の状態を比較することによってファイルの変更や改竄等の不整合を検知する。そのためのデータベースを作成し、動作確認を行う。

tripwireデータベースの作成

```
# tripwire --init
Please enter your local passphrase: ←ローカルパスフレーズを入力
Parsing policy file: /etc/tripwire/tw.pol
Generating the database...
*** Processing Unix File System ***
The object: "/boot/efi" is on a different file system...ignoring.
Wrote database file: /var/lib/tripwire/<ホスト名>.twd
The database was successfully generated.  （データベース作成完了）
```

tripwireの動作確認（整合性チェック実行）

```
# tripwire --check
Parsing policy file: /etc/tripwire/tw.pol
*** Processing Unix File System ***
Performing integrity check...
The object: "/boot/efi" is on a different file system...ignoring.
Wrote report file: /var/lib/tripwire/report/<ホスト名>-20200625-222203.twr（レポート保存先）


Open Source Tripwire(R) 2.4.3.7 Integrity Check Report
        :
Total violations found:  0（不整合 0）

===============================================================================
Object Summary:
===============================================================================
```

```
---------------------------------------------------------------------------
# Section: Unix File System
---------------------------------------------------------------------------

No violations.（不整合なし）

===========================================================================
Error Report:
===========================================================================

No Errors（エラーなし）

---------------------------------------------------------------------------
*** End of report ***
        :
Integrity check complete.（チェック完了）
```

動作確認を終えたら、テキスト版のポリシーファイル/etc/tripwire/twpol.txt.newは削除しておく。

テキスト版ポリシーファイル/etc/tripwire/twpol.txt.newの削除

```
# rm /etc/tripwire/twpol.txt.new
```

暗号署名されたポリシーファイルの内容をテキストファイルへ出力する場合には、twadminコマンドを使う。

暗号署名版ポリシーファイルのテキストファイル（/etc/tripwire/polprint.txt）への出力

```
# twadmin --print-polfile > /etc/tripwire/polprint.txt
```

整合性チェックの内容を変更する場合には、このようにして出力したテキストファイルを編集して、暗号署名を行う（3.2.3）。

3.2.5. データベースの更新

tripwireによる整合性チェックでは、管理者あるいはシステムが正当に変更を行ったファイルも不整合として検知される。以下の例では、tripwireデータベースのバックアップファイルが不整合（新規追加）として検知されている。

不整合検知の例

```
# tripwire --check
        :
Wrote report file: /var/lib/tripwire/report/<ホスト名>-20200626-120703.twr（レポート保存先）
        :
Total violations found:  1（不整合 1）
        :
```

```
Added:
"/var/lib/tripwire/<ホスト名>.twd.bak"（不整合の内容: ファイルの追加）
        :
```

　このような不適切な検知を回避するために、データベースを更新する必要がある。データベースの更新は、tripwire --update --twrfileに続けて、レポート保存先を指定することで行う。viエディタが起動するので、[x]の有無の確認を行ったのち、保存する。

tripwireデータベースの更新

```
# tripwire --update --twrfile /var/lib/tripwire/report/<ホスト名>-20200626-120703.twr
（vi エディタ起動）
                    :
Remove the "x" from the adjacent box to prevent updating the database
with the new values for this object.

Added:
[x] "/var/lib/tripwire/centos8r.twd.bak"（「x」のついている部分がデータベースに追加される）
                    :
Please enter your local passphrase: ←ローカルパスフレーズを入力
Wrote database file: /var/lib/tripwire/<ホスト名>.twd（データベース更新完了）
```

　保存されたレポートファイルはバイナリファイルなので、直接読むことはできない。テキスト形式で出力する場合には、twprintコマンドを使う。

tripwireレポートのテキスト出力

```
# twprint --print-report --twrfile /var/lib/tripwire/report/<ホスト名>-20200626-120703.twr
```

3.2.6. 定時チェックの設定

　tripwireによるチェックと、チェック結果のメール送信、tripwireデータベースの更新を行うスクリプトtripwire.shを/rootディレクトリに作成し、定時に実行するように設定する。

/root/tripwire.shの作成

```
#!/bin/sh

LOCALPASS="<ローカルパスフレーズ>"
RUNDATE=`date +%Y%m%d-%H%M%S`
TWRFILE=/tmp/tripwire_$RUNDATE.twr（レポートファイル）
TRIPWIRE=/usr/sbin/tripwire（tripwire コマンド）
HOSTNAME=`hostname`      （ホスト名）
MAILADDR=<送信先メールアドレス>
SUBJECT="[Tripwire Check Report]"

$TRIPWIRE --check --twrfile $TWRFILE | mail -s "$SUBJECT $HOSTNAME" ¥（メール送信）
```

```
-S smtp=smtp://<SMTP サーバーホスト名>:<ポート番号> ¥
-S smtp-auth=login ¥
-S smtp-auth-user=<ユーザー名> ¥
-S smtp-auth-password=<パスワード> ¥
-S smtp-use-starttls ¥（通信保護に STARTTLS を使用する場合は記述）
-S from=<送信元メールアドレス> ¥
-S nss-config-dir=/etc/openldap/certs/ ¥
-S ssl-verify=ignore ¥
$MAILADDR

$TRIPWIRE --update --twrfile $TWRFILE --local-passphrase $LOCALPASS --accept-all
（tripwire データベース更新）
rm -f $TWRFILE（レポートファイル削除）
```

/root/tripwire.shの動作確認

```
# cd /root
# chmod u+x tripwire.sh（実行権限を設定）
# ./tripwire.sh（スクリプトの実行）
```

/root/tripwire.shの実行ファイル化

```
# shc -r -f tripwire.sh（tripwire.sh の実行ファイル化）
# rm tripwire.sh tripwire.sh.x.c（元ファイルおよび中間ファイルの削除）
```

cronによる定時実行設定

```
# crontab -e（cron 設定ファイル編集画面(vi エディタ)表示）
                    :
30 4 * * * /root/tripwire.sh.x（30 分 4 時に毎日/root/tripwire.sh.x を実行）
crontab: installing new crontab（編集内容を保存完了メッセージ表示）
# crontab -l（cron 登録内容の表示）
          :
30 4 * * * /root/tripwire.sh.x
```

3.3. Java環境（OpenJDK）

Java環境のOpenJDKをインストールし、環境変数を設定する。

(1) OpenJDKのインストール

最新版[10]をインストールする場合には、dnfコマンドを以下のように実行する。

OpenJDK最新版のインストール

```
# dnf -y install java-latest-openjdk
```

[10] 2022年6月現在提供されている長期サポート版は、Java 8、11、17である。安定運用のためには、長期サポート版をインストールすることが望ましい。

ランタイムだけでなく、コンパイラを含む開発環境をインストールする場合には、以下のように実行する。

OpenJDK開発環境最新版のインストール

```
# dnf -y install java-latest-openjdk-devel
```

インストール可能なOpenJDKパッケージの一覧は、以下のコマンドを実行することで得られる。OpenJDKの特定のバージョンをインストールする場合には、そのバージョンのパッケージを指定する。

インストール可能なOpenJDKパッケージの一覧取得

```
# dnf list available | grep openjdk
java-1.8.0-openjdk.x86_64 …
        :
java-1.8.0-openjdk-devel.x86_64 …
        :
java-11-openjdk.x86_64 …
        :
java-11-openjdk-devel.x86_64 …
        :
java-latest-openjdk.x86_64 …
        :
java-latest-openjdk-devel.x86_64 …
        :
```

(2) 環境変数の設定

/etc/profile.d/sh.localの最下行に以下の1行を追加し、環境変数JAVA_HOMEを設定する。

/etc/profile.d/sh.localファイルへの行追加による環境変数JAVA_HOMEの設定

```
                :
export JAVA_HOME=/etc/alternatives/java_sdk[11]
```

環境変数の設定を反映させるには、OSを再起動する。

OSの再起動

```
# reboot
```

[11] AdoptOpenJDKやAmazon Corretto、Liberica JDKなどのJavaを別途インストールして使用する場合には、そのインストールディレクトリのパスを設定する。

3.4. Webサーバー（Apache HTTP Server）

3.4.1. Apache HTTP Serverのインストールと設定

Webサーバーソフトウェアapache HTTP Serverをインストール、ディレクトリ情報の非表示設定とサービス化を行う。

Apache HTTP Serverのインストール

```
# dnf -y install httpd
```

Apache HTTP Serverをdnfでインストールした場合、レスポンスで指定される文字コードはUTF-8となり、HTMLファイルでUTF-8以外の文字コードを指定していてもブラウザに無視され、文字化けが発生する。設定ファイル/etc/httpd/conf/httpd.confを編集し、AddDefaultCharsetをオフに設定する。

/etc/httpd/conf/httpd.confの編集による文字コード指定の解除

```
                    :
#AddDefaultCharset UTF-8←この行をコメントアウト
AddDefaultCharset off←この行を追加
                    :
```

また、Apache HTTP Serverの標準設定では、URLとしてディレクトリを指定した場合や該当ページが存在しない場合には、そのディレクトリ内のファイルの一覧が表示される。そのままだとサイトのファイル一覧が容易に閲覧され、攻撃のリスクが高くなる。これを回避するために、/etc/httpd/conf/httpd.confを以下のように編集する。

/etc/httpd/conf/httpd.confの編集によるディレクトリ情報の非表示設定

```
                    :
<Directory “/var/www/html”>
                    :
    #Options Indexes FollowSymLinks←この行をコメントアウト
    Options FollowSymLinks←この行を追加
                    :
</Directory>
                    :
```

Apache HTTP Serverの標準設定では、/var/www/html ディレクトリがWebブラウザからのアクセスを受け付ける最上位のディレクトリ（ルートディレクトリ）となっている。ここで行う非表示設定は、/var/www/html以下のすべてのディレクトリ、すなわちWebブラウザがアクセス可能なすべてのディレクトリに対して有効となる。

設定を終えたら、Apache HTTP Serverをサービス化してサービスを起動する。

Apache HTTP Serverのサービス化とサービス起動

```
# systemctl enable httpd
# systemctl start httpd
```

httpサービスの通信を許可するように、ファイアウォールを設定する。

httpサービスの通信許可設定

```
# firewall-cmd --add-service=http --permanent
# firewall-cmd --reload
```

3.4.2. Webページへのアクセス制限設定

特定のWebページに対するアクセスを、接続元のIPアドレスやホスト名によって制限する場合には、/etc/httpd/conf/httpd.confに<Directory>タグを記述して設定する。

下の例は、ルートディレクトリ（/var/www/html）直下のtestディレクトリへのアクセス制限を設定したものである。

/etc/httpd/conf/httpd.confへの<Directory>タグ記述によるアクセス制限設定

```
                    :
<Directory “/var/www/html/test”>
    Order deny,allow（拒否設定後、例外としての許可対象を記述。"allow,deny"だと逆になる。）
    Deny from all（全部拒否）
    Allow from localhost,192.168.1.1
                （localhost および IP アドレス 192.168.1.1 からのアクセスを許可）
    Allow from 172.16.1.0/24（IP アドレス範囲 172.16.1.0/24 からのアクセスを許可）
    Allow from xxx.yyy.zzz（ホスト xxx.yyy.zzz からのアクセスを許可）
</Directory>
                    :
```

3.5. サーブレットコンテナ（Tomcat）

3.5.1. Tomcatのインストールと設定

サーバー上のJavaプログラム（サーブレット）の実行管理を行うサーブレットコンテナTomcatのインストールと各種設定を行う。

(1) Tomcatダウンロード
Tomcatは、ホームページより最新版をダウンロードする[12,13]。

(2) ダウンロードファイルの展開とTomcat用ユーザーアカウントの設定
ダウンロードしたファイル（apache-tomcat-<バージョン番号>.tar.gz）を/usr/localディレクトリへ展開する。さらに、Tomcat用のユーザーアカウント「tomcat」を新しく作成し、Tomcatファイル一式の所有者を「tomcat」に設定する。

---

[12] URLはhttp://tomcat.apache.org/、最新版は、Tomcat 9がapache-tomcat-9.0.68.tar.gz、10がapache-tomcat-10.1.11.tar.gzである（2023年7月現在）。

[13] Tomcat 10では、サーブレットのパッケージ名がjavax.servlet.*からjakarta.servlet.*に変更された。Javax.servlet.*パッケージを使用しているアプリケーションをTomcat 10で動作させるには、warファイルをwebapps-javaeeフォルダへ配備する。

ダウンロードファイルの展開とTomcat用ユーザーアカウントの設定

```
# tar zxvf apache-tomcat-<バージョン番号>.tar.gz -C /usr/local（Tomcat 一式の展開）
# rm -f apache-tomcat-<バージョン番号>.tar.gz（ダウンロードファイルの削除）
# useradd -s /sbin/nologin tomcat（「tomcat」をログインなしユーザーとして登録）
# chown -R tomcat:tomcat /usr/local/apache-tomcat-<バージョン番号>
（Tomcat 一式の所有者設定）
# ln -s /usr/local/apache-tomcat-<バージョン番号> /usr/local/tomcat（リンク設定）
```

(3) 環境変数の設定
/etc/profile.d/sh.localの最下行に以下の2行を追加し、Tomcat環境変数を設定する。

/etc/profile.d/sh.localの編集によるTomcat環境変数の設定

```
                    :
export CATALINA_HOME=/usr/local/tomcat
export CATALINA_BASE=$CATALINA_HOME
```

環境変数の設定を反映させるには、OSを再起動する。

OSの再起動

```
# reboot
```

(4) Apache HTTP Serverとの連携設定
/usr/local/tomcat/conf/server.xmlを編集し、Apache HTTP Serverとの連携を行えるようにする。

/usr/local/tomcat/conf/server.xmlの編集によるApache HTTP Server連携機能有効化

```
                    :
<!-- ←この行を削除（コメントを外す）
<Connector protocol="AJP/1.3"
               address="::1"
               port="8009"
               redirectPort="8443" />
--> ←この行を削除（コメントを外す）
                    :
```

コメントを外して、以下のように編集。

```
                    :
<Connector protocol="AJP/1.3"
               address="localhost" ←address を「localhost」に修正
               port="8009"
               secret="<パスワード>" ←この行を追加
               redirectPort="8443" />
                    :
```

/etc/httpd/conf.dディレクトリに新しくajp.conf[14]を作成し、Apache HTTP Server側の設定を行う。

/etc/httpd/conf.d/ajp.confの新規作成

```
<Location /<Httpd ディレクトリ>/>
   ProxyPass ajp://localhost:8009/<Tomcat ディレクトリ>/ secret=<パスワード>
   （パスワードは、/usr/local/tomcat/server.xml と同じものを記述）
</Location>
```

この設定では、http(s)://localhost/<Httpdディレクトリ>/へのアクセスが発生した場合、その処理は/usr/local/tomcat/webapps/<Tomcatディレクトリ>/へ転送される。

SELinux（5ページ）を有効化している場合には、以下の設定も行っておく。

SELinuxを有効化した場合の追加設定

```
# setsebool -P httpd_can_network_connect 1（httpd によるネットワーク接続許可）
# reboot（OS 再起動による SELinux 設定の反映）
```

(5) Apache HTTP Serverのタイムアウト設定

Tomcatでの処理に時間が掛かる場合、標準設定のままではApache HTTP Serverとの通信が遮断されることがある。これを回避するため、/etc/httpd/conf/httpd.confの最下行に以下の2行を記述し、タイムアウトを設定する。

/etc/httpd/conf/httpd.confへのタイムアウト設定の記述追加

```
                    :
Timeout 600
ProxyTimeout 600
```

Apache HTTP Serverサービスを再起動する。

Apache HTTP Serverサービスの再起動

```
# systemctl restart httpd
```

(6) エラーページにおけるTomcatバージョン情報非表示設定

標準設定では、エラーページにはTomcatのバージョン情報が表示される。安全上、このような表示は好ましくないため、バージョン情報の記載を削除する。

バージョン情報の記載削除

```
# cd /usr/local/tomcat/lib
# jar xf catalina.jar org/apache/catalina/util/ServerInfo.properties
# vi org/apache/catalina/util/ServerInfo.properties
#server.info=Apache Tomcat/<バージョン番号> ←この行をコメントアウト
```

[14] 拡張子がconfであれば、ファイル名は任意でよい。

```
server.info=  ←この行を追加
#server.number=<バージョン番号> ←この行をコメントアウト
server.number=  ←この行を追加
#server.built=<日付> ←この行をコメントアウト
server.built=  ←この行を追加
# jar uf catalina.jar org/apache/catalina/util/ServerInfo.properties
# rm -rf org
```

(7) Tomcatのサービス化

Tomcatをサービス化するために必要なアプリケーションjsvcのインストールを行う。

jsvcのインストール

```
# cd /usr/local/tomcat/bin
# tar zxvf commons-daemon-native.tar.gz
# cd commons-daemon-<バージョン番号>-native-src/unix/
# ./configure
# make
# cp jsvc ../..
```

/usr/lib/systemd/system/tomcat.serviceを新規に作成し、Tomcatをサービスとして登録する。

/usr/lib/systemd/system/tomcat.serviceの新規作成

```
[Unit]
Description=Apache Tomcat Web Application Container
After=network.target

[Service]
User=tomcat
Group=tomcat
Type=forking
Environment=JAVA_HOME="<Java（JDK）インストールディレクトリのパス>"
Environment=CATALINA_HOME=/usr/local/tomcat
Environment=CATALINA_BASE=/usr/local/tomcat
ExecStart=/usr/local/tomcat/bin/daemon.sh start
ExecStop=/usr/local/tomcat/bin/daemon.sh stop
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

Tomcatサービスの登録と起動

```
# systemctl enable tomcat
# systemctl start tomcat
```

3.5.2. Webページへのアクセス制限設定

Webページに対するアクセスを、接続元のIPアドレスやホスト名によって制限する場合には、以下のような設定を行う。なお、IPアドレスによる制限とホスト名による制限を同時に行うことはできない。両方を設定した場合、IPアドレスによる制限のみが有効となる。

(1) すべてのページに対するアクセス制限を一括して行う場合
/usr/local/tomcat/conf/server.xmlに<Valve>タグを記述して設定する。<Valve>タグは、<Host>、<Engine>、<Context>タグの中に記述できる。ホスト名で制限する場合には、<Connector>タグの設定も必要である。

IPアドレスによる制限
/usr/local/tomcat/conf/server.xml への<Valve>タグの記述

```
<Valve className="org.apache.catalina.valves.RemoteAddrValve"
                allow="<IP アドレスの正規表現 1>|<IP アドレスの正規表現 2>|..."
                deny="<IP アドレスの正規表現 1>|<IP アドレスの正規表現 2>|..."/>
```

IPアドレスの記述例: 192.168.1.2と172.16.1.0～172.16.1.255を許可する場合
allow="192¥.168¥.1¥.2|172¥.16¥.1¥.¥d{1,3}"

ホスト名による制限
/usr/local/tomcat/conf/server.xml の<Connector>タグの設定と<Valve>タグの記述

```
                :
<Connector protocol="AJP/1.3"
           address="localhost"
           port="8009"
           secret="..."
           enableLookups="true" ←この行を追加
           redirectPort="8443" />
                :
<Valve className="org.apache.catalina.valves.RemoteHostValve"
                allow="<ホスト名の正規表現 1>|<ホスト名の正規表現 2>|..."
                deny="<ホスト名の正規表現 1>|<ホスト名の正規表現 2>|..."/>
```

ホスト名の記述例: xxx.yyy.zzzと*.aaa.bbbを許可する場合
allow="xxx¥.yyy¥.zzz|.+¥.aaa¥.bbb"

allowが設定されている場合は、設定されているもののみアクセスを許可する。allowが設定されていない場合は、denyに設定されていないもののみアクセスを許可する。denyが設定されている場合は、設定されているもののみアクセスを拒否する。denyが設定されていない場合は、アクセスはallowの設定に依存する。

(2) ウェブアプリ（コンテキスト）ごとにアクセス制限を行う場合
/usr/local/tomcat/conf/Catalina/localhost内に<コンテキスト名>.xmlファイルを作り、以下の内容を記述する。ホスト名で制限する場合には、

/usr/local/tomcat/conf/server.xml の中の<Connector>タグの設定も必要である。

IPアドレスによる制限
/usr/local/tomcat/conf/Catalina/localhost/<コンテキスト名>.xmlの作成

```
<Context path="<ルートディレクトリからのウェブアプリのパス>"
                docBase="<ウェブアプリディレクトリのファイルシステム上の絶対パス>">
               <Valve className="org.apache.catalina.valves.RemoteAddrValve"
                      allow=”<IP アドレスの正規表現 1>|<IP アドレスの正規表現 2>|…“
                      deny=”<IP アドレスの正規表現 1>|<IP アドレスの正規表現 2>|…“/>
</Context>
```

ホスト名による制限
/usr/local/tomcat/conf/server.xml の<Connector>タグの設定

```
                    :
<Connector protocol="AJP/1.3"
                address="localhost"
                port="8009"
                secret=”…”
                enableLookups="true" ←この行を追加
                redirectPort="8443" />
                    :
```

/usr/local/tomcat/conf/Catalina/localhost/<コンテキスト名>.xmlの作成

```
<Context path="<ルートディレクトリからのウェブアプリのパス>"
                docBase="<ウェブアプリディレクトリのファイルシステム上の絶対パス>">
               <Valve className="org.apache.catalina.valves.RemoteHostValve"
                      allow=”<ホスト名の正規表現 1>|<ホスト名の正規表現 2>|…”
                      deny=”<ホスト名の正規表現 1>|<ホスト名の正規表現 2>|…”/>
</Context>
```

いずれの場合も、ウェブアプリのディレクトリが/usr/local/tomcat/webapps直下にある場合は、pathとdocBaseの記述は不要である。

3.6. HTTPS化（Let’s Encrypt）

HTTPSとは、Webブラウザが用いるHTTP通信を、SSL/TLS（Secure Sockets Layer/Transport Layer Security）プロトコル（通信規約）により提供される安全な接続の上で行うための仕組みである。通信の安全確保のため、Apache HTTPサーバーとTomcatがHTTPSに対応できるように機能の追加と設定を行う。

3.6.1. Apache HTTPサーバーのHTTPS化

Let’s Encryptは、非営利団体のInternet Security Research Groupが運営している証明書認証局で、SSL/TLS証明書を無料で発行している。証明書の有効期間は90日間で、有効期間内であれば、証明書の更新はいつでも無料で行うことができる。以下の手順により、

Let's Encryptが発行する証明書を導入してApache HTTPサーバーのHTTPS化を行う[15]。

(1) ファイアウォールの設定
　　httpsサービスの通信許可設定を行う。

httpsサービスの通信許可設定

```
# firewall-cmd --add-service=https --permanent
# firewall-cmd --reload
```

(2) .well-knownディレクトリのProxy除外設定の追加
　　Let's Encryptの証明書取得過程では、.well-knownディレクトリへのアクセスが行われる。これが他へ転送されないように、設定を/etc/httpd/conf.d/ajp.conf（28ページ）に追加し、Apache HTTPサーバーを再起動する。

/etc/httpd/conf.d/ajp.confへの記述追加

```
                    :
<Location /.well-known/>（この行以下を追記）
   ProxyPass !
</Location>
```

Apache HTTPサーバーの再起動

```
# systemctl restart httpd
```

(3) certbotのインストール
　　証明書取得ツールcertbotをインストールする。certbotは、クロスプラットフォームのパッケージ管理システムsnappyでインストールする。

snappyによるcertbotのインストール

```
# dnf -y install snapd（snappy パッケージのインストール）
# systemctl enable --now snapd.socket（snapd サービスの有効化）
# systemctl start snapd.service（snapd サービスの開始）
# ln -s /var/lib/snapd/snap /snap（snap ディレクトリへのシンボリックリンク設定）
# snap install core && snap refresh core（snappy パッケージを最新版に更新）
# snap install certbot --classic（certbot のインストール）
```

(4) 証明書の取得
　　証明書を取得する。標準設定では、Apache HTTPサーバーのルートディレクトリは/var/www/htmlである。

[15] LAN（Local Area Network）内のプライベートIPアドレスで運用するローカルなサーバーをHTTPS化する場合は、自己署名証明書を発行して導入する。その手順は、付録 5で紹介する。

証明書の取得

```
# certbot certonly --webroot -w <HTTP サーバルートディレクトリ> -d <ホスト名>
          :
Enter email address (used for urgent renewal and security notices)
 (Enter 'c' to cancel): ←連絡先 E-mail アドレスを入力

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf. You must
agree in order to register with the ACME server at
https://acme-v02.api.letsencrypt.org/directory
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(A)gree/(C)ancel: A ←「A」と入力

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Would you be willing, once your first certificate is successfully issued, to
share your email address with the Electronic Frontier Foundation, a founding
partner of the Let's Encrypt project and the non-profit organization that
develops Certbot? We'd like to send you email about our work encrypting the web,
EFF news, campaigns, and ways to support digital freedom.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: ←Electronic Frontier Foundation からのメールニュース等を希望する場合は Y
          :
IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at:
    /etc/letsencrypt/live/<ホスト名>/fullchain.pem
    Your key file has been saved at:
    /etc/letsencrypt/live/<ホスト名>/privkey.pem
    Your cert will expire on 2020-11-27. To obtain a new or tweaked
    version of this certificate in the future, simply run certbot-auto
    again. To non-interactively renew *all* of your certificates, run
    "certbot-auto renew"
 - If you like Certbot, please consider supporting our work by:

    Donating to ISRG / Let's Encrypt:     https://letsencrypt.org/donate
    Donating to EFF:                      https://eff.org/donate-le
# ls /etc/letsencrypt/live/<ホスト名>（生成されたファイルを確認）
README    cert.pem    chain.pem    fullchain.pem    privkey.pem
```

(5) 証明書の登録

/etc/httpd/conf.d/ssl.confに証明書を登録する。

/etc/httpd/conf.d/ssl.confの編集

```
                    :
```

```
#ServerName www.example.com:443 ←この行をコメントアウト
                    :
#SSLCertificateFile /etc/pki/tls/certs/localhost.crt ←この行をコメントアウト
SSLCertificateFile /etc/letsencrypt/live/<ホスト名>/fullchain.pem ←この行を追加
                    :
#SSLCertificateKeyFile /etc/pki/tls/private/localhost.key ←この行をコメントアウト
SSLCertificateKeyFile /etc/letsencrypt/live/<ホスト名>/privkey.pem ←この行を追加
                    :
```

(6) httpからhttpsへのリダイレクト設定

新規に/etc/httpd/conf.d/le-redirect.conf[16]ファイルを作成してhttpからhttpsへのリダイレクト設定を記述する。

/etc/httpd/conf.d/le-redirect.confの作成

```
<VirtualHost _default_:80>
ServerName <ホスト名>

ServerSignature Off

RewriteEngine On
RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI}
[END,NE,R=permanent]

ErrorLog /var/log/httpd/redirect.error.log
LogLevel warn
</VirtualHost>
```

(7) Apach HTTPサーバーの再起動

以上の変更を反映させるため、Apache HTTPサーバーを再起動する

Apache HTTPサーバーの再起動

```
# systemctl restart httpd
```

(8) 証明書の自動更新設定

証明書の有効期間は90日間なので、期限が切れる前に更新[17]する必要がある。snappyでインストールした場合には、自動でタイマーが設定される。

snappyによって設定された自動更新タイマーの確認

```
# systemctl list-timers（システムタイマーの一覧表示）
NEXT  LEFT          LAST          PASSED       UNIT                         ACTIVATES
```

[16] 拡張子がconfであれば、ファイル名は任意でよい。

[17] 証明書の更新時には、httpおよびhttpsサービスの通信を任意のホストに対して許可しておく必要がある。

```
....      ....          ....          ....          ....                          ....
....      6h left       ....          7h ago        snap.certbot.renew.timer      ....
          :
```

自動更新の時期などを別途設定する場合には、タイマーの停止および無効化と、cronによる自動更新設定を行う。

自動更新タイマーの停止と無効化

```
# systemctl stop snap.certbot.renew.timer（タイマーの停止）
# systemctl disable snap.certbot.renew.timer（タイマーの無効化）
```

cronによるLet’s Encrypt証明書の自動更新設定

```
# crontab -e（cron 設定ファイル編集画面(vi エディタ)表示）
                    :
0 0 1 */2 * /var/lib/snapd/snap/bin/certbot renew --force-renewal（2 か月毎 1 日 0 分 0 時に更新）
crontab: installing new crontab（編集内容を保存完了メッセージ表示）
# crontab -l（cron 登録内容の表示）
          :
0 0 1 */2 * /var/lib/snapd/snap/bin/certbot renew --force-renewal
```

certbot renewコマンドに--force-renewalオプションを付けない場合は、有効期限が30日以内のときのみ更新を行う。また、動作確認のみを行う場合には、--dry-runオプションを付ける。

証明書の有効期限は、certbotあるいはopensslコマンドにより確認できる。

Let’s Encrypt証明書の有効期限確認

certbotコマンド

```
# certbot certificates
```

opensslコマンド

```
# openssl x509 -in /etc/letsencrypt/live/<ドメイン名>/cert.pem -noout -dates
```

3.6.2. TomcatのHTTPS化

TomcatのHTTPS化は、以下の手順により行う。

(1) ファイアウォールの設定

SSL/TLS通信で使用するポートの通信許可設定を行う。

使用ポートの通信許可設定

```
# firewall-cmd --add-port=8443/tcp --permanent
# firewall-cmd --reload
```

(2) Tomcat Native Libraryのインストール

SSL/TLS通信を行うために必要なTomcat Native Libraryのインストールを行う。

Tomcat Native Libraryのインストール

```
# dnf -y install openssl-devel（SSL/TLS ソフトウェア OpenSSL 開発キットのインストール）
# dnf -y install apr-devel
   （API ソフトウェア Apache Portable Runtime 開発キットのインストール）
# cd /usr/local/tomcat/bin
# tar zxvf tomcat-native.tar.gz
# cd tomcat-native-</バージョン番号>-src/native
# ./configure
# make
# make install
```

ライブラリは、/usr/local/apr/libに作成される。このディレクトリのパスを環境変数LD_LIBRARY_PATHに設定するように、/usr/lib/systemd/system/tomcat.serviceへ追記する。

/usr/lib/systemd/system/tomcat.service

```
[Unit]
                    :

[Service]
                    :
Environment=LD_LIBRARY_PATH=/usr/local/apr/lib ←この行を追加

[Install]
                    :
```

■ Tomcat 9 の場合

(i) Let’s Encrypt認証ファイルのTomcat環境へのコピー

Let’s Encrypt認証ファイル「/etc/letsencrypt/live/<ドメイン名>/*.pem」は読取権限設定により、Tomcatで読み込めないことがある。認証ファイルをTomcat環境へコピーし、読み込めるようにしておく。

Let’s Encrypt認証ファイルのコピー

```
# cp -fb /etc/letsencrypt/live/<ドメイン名>/*.pem /usr/local/tomcat/conf ←ファイルのコピー
# chown tomcat:tomcat /usr/local/tomcat/conf/*.pem ←所有者を tomcat に変更
```

(ii) 設定ファイル/usr/local/tomcat/conf/server.xmlの編集

/usr/local/tomcat/conf/server.xmlを編集し、SSL/TLS通信機能を有効化する。

/usr/local/tomcat/conf/server.xmlの編集によるSSL/TLS通信機能有効化

```
    <!-- ←この行を削除（コメントを外す）
    <Connector port="8443" protocol="org.apache.coyote.http11.Http11AprProtocol"
```

```
                maxThreads="150" SSLEnabled="true" >
        <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
        <SSLHostConfig>
            <Certificate certificateKeyFile="conf/localhost-rsa-key.pem"
                         certificateFile="conf/localhost-rsa-cert.pem"
                         certificateChainFile="conf/localhost-rsa-chain.pem"
                         type="RSA" />
        </SSLHostConfig>
    </Connector>
    --> ←この行を削除（コメントを外す）
```

コメントを外して、以下のように編集する。

```
    <Connector port="8443" protocol="org.apache.coyote.http11.Http11AprProtocol"
                maxThreads="150" SSLEnabled="true" >
        <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
        <SSLHostConfig protocols="TLSv1.2"> ←許可するプロトコルをカンマ区切りで記述
            <Certificate certificateKeyFile="/usr/local/tomcat/conf/privkey.pem"
                         certificateFile="/usr/local/tomcat/conf/cert.pem"
                         certificateChainFile="/usr/local/tomcat/conf/chain.pem"
                         （Let's Encrypt 認証ファイル 3 つの名前をフルパスで記述）
                         type="RSA" />
        </SSLHostConfig>
    </Connector>
```

(iii) Tomcat再起動

設定を反映するため、Tomcatを再起動する。

Tomcat再起動

```
# systemctl daemon-reload ←サービス設定ファイルの再読み込み
# systemctl restart tomcat
```

■ Tomcat 10の場合

(i) Tomcat用証明書の作成

Let's Encrypt証明書を元に、Tomcat用証明書を作成して導入する。

Tomcat用証明書の作成

```
# cd /etc/letsencrypt/live/<ホスト名>
# openssl pkcs12 -export -in cert.pem -inkey privkey.pem -certfile chain.pem -out cert.p12 -
passout pass:<Tomcat 証明書用パスワード>←指定したパスワードは設定ファイルで参照される
# chown tomcat:tomcat cert.p12
# mv cert.p12 /usr/local/tomcat/conf
```

(ii) 設定ファイル/usr/local/tomcat/conf/server.xmlの編集

/usr/local/tomcat/conf/server.xmlを編集し、SSL/TLS通信機能を有効化する。

/usr/local/tomcat/conf/server.xmlの編集によるSSL/TLS通信機能有効化

```
<!-- ←この行を削除（コメントを外す）
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
           maxThreads="150" SSLEnabled="true">
    <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
    <SSLHostConfig>
        <Certificate certificateKeystoreFile="conf/localhost-rsa.jks"
                     type="RSA" />
    </SSLHostConfig>
</Connector>
--> ←この行を削除（コメントを外す）
```

コメントを外して、以下のように編集する。

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
           maxThreads="150" SSLEnabled="true">
    <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
    <SSLHostConfig protocols="TLSv1.2"> ←許可するプロトコルをカンマ区切りで記述
        <Certificate certificateKeystoreFile="/usr/local/tomcat/conf/cert.p12"
                     certificateKeystorePassword="<Tomcat 証明書用パスワード>"
                     type="RSA" />
    </SSLHostConfig>
</Connector>
```

(iii) Tomcat再起動

設定を反映するため、Tomcatを再起動する。

Tomcat再起動

```
# systemctl daemon-reload ←サービス設定ファイルの再読み込み
# systemctl restart tomcat
```

3.7. データベース（MariaDB）

3.7.1. MariaDBのインストールと設定

データベース管理ソフトウェアMariaDBのインストールと初期設定を行う。

(1) MariaDBリポジトリの登録

MariaDBウェブサイト[18]からDownload＞MariaDB Repositoriesをクリックし、表示画面にて、以下の項目を選択する。

Choose a distribution: Red Hat EL 8 (x86_64)

Choose a MariaDB Server Version: 10.6[19]

リポジトリ登録ファイル/etc/ yum.repos.d/MariaDB.repoファイルを作成し、画面に表示されたリポジトリ情報をコピーして貼り付ける。

---

[18] https://mariadb.org/（2022年6月現在）

[19] 10.6までは5年間の長期サポート、10.7以降は1年間の短期サポートとされている。

/etc/yum.repos.d/MariaDB.repoファイルの作成

```
# vi /etc/yum.repos.d/MariaDB.repo
# MariaDB 10.6 RedHat repository list - created 2022-06-28 06:46 UTC
# https://mariadb.org/download/
[mariadb]
name = MariaDB
baseurl = https://ftp.yz.yamagata-u.ac.jp/pub/dbms/mariadb/yum/10.6/rhel8-amd64
module_hotfixes=1
gpgkey=https://ftp.yz.yamagata-u.ac.jp/pub/dbms/mariadb/yum/RPM-GPG-KEY-MariaDB
gpgcheck=1
#
```

(2) MariaDBのインストール

```
# dnf -y install MariaDB-server
```

(3) 文字コードの設定およびサービスの起動・有効化

```
# vi /etc/my.cnf.d/server.cnf
                    :
[mysqld]
                    :
character_set_server = utf8mb4 ←この行を追加
collation_server = utf8mb4_general_ci ←この行を追加
                    :
#
```

(4) MariaDBのサービス化とサービス起動

```
# systemctl enable mariadb
# systemctl start mariadb
```

(5) MariaDBのrootパスワード設定と匿名ユーザーの削除
rootのパスワード設定およびパスワードなしでログインできる匿名ユーザー[20]の削除を行う。

```
# mariadb -u root
        :
MariaDB [(none)]> alter user 'root'@'localhost' identified by '<root 用パスワード>'; ←入力
Query OK, 0 rows affected (0.003 sec)

MariaDB [(none)]> drop user ''@'localhost'; ←匿名ユーザーの削除
Query OK, 0 rows affected (0.003 sec)
```

[20] 匿名ユーザーを残しておくと、追加したユーザーのパスワード設定が機能しないことがある。

```
MariaDB [(none)]> drop user ''@'<ホスト名>'; ←匿名ユーザーの削除
Query OK, 0 rows affected (0.003 sec)

MariaDB [(none)]> exit（MariaDB からログアウト）
Bye
#
```

(6) MariaDBユーザーの追加とユーザーデータベースの作成
リモートログインの可能なユーザーを追加し、そのユーザー用のデータベースを作成する。ここでは、「aist」というユーザーを作成し、aist用のデータベース「cf」を作成するものとする。

ユーザー「aist」とaist用データベース「cf」の作成

```
$ mariadb -u root -p（root で MariaDB にログイン）
Enter password: ←パスワード入力
        :
> create database cf;（データベース作成）
> grant all on cf.* to aist@'%' identified by '<aist 用のパスワード>';（ユーザー「aist」作成）
> exit（MariaDB からログアウト）
```

3.7.2. SSHを用いたMariaDBサーバーへのリモート接続

MariaDBが使用している3306番ポートへの接続許可をファイアウォールで設定することで、他のPCからMariaDBへリモート接続ができるようになる。

3306番ポートへの接続許可設定

```
# firewall-cmd --add-port=3306/tcp
```

MariaDBへのリモート接続

```
$ mariadb -u <MariaDB ユーザー名> -p -h <MariaDB ホスト名／IP アドレス>
Enter password: ←パスワード入力
```

しかしながら、この通信は保護されておらず、安全上問題がある。ここでは、sshによるポート転送を利用したリモート接続を紹介する。

ポート転送とは、ローカルPCのポートへの通信をリモートホストの指定したポートへ転送するものである。転送先をMariaDBホストの3306番ポートに指定することで、ローカルPCのポートへ送ったMariaDBコマンドを、リモートのMariaDBサーバーへ転送することができる。この間の通信はSSHによって保護される。

以下に、その手順を示す。なお、ポート転送によるリモート接続を行う場合、ファイアウォールにおける3306番ポートへの接続許可設定は不要である。

(1) ポート転送用公開鍵および暗号鍵の生成と登録
ポート転送をパスワード入力なしで行えるように、そのための公開鍵と暗号鍵を新たに生成して登録する。そしてセキュリティ確保のため、その鍵で接続した場合には、どのコマンドも実行できないように設定する。

公開鍵と秘密鍵の生成、公開鍵の登録とコマンド実行制限の設定

```
$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/<ユーザー名>/.ssh/id_ed25519): /home/<ユーザー名>/.ssh/id_portfwd ←鍵ファイル名入力
Enter passphrase (empty for no passphrase): ←Enter キー入力
Enter same passphrase again: ←Enter キー入力
        :
$ cd ~/.ssh
$ cat id_portfwd.pub >> authorized_keys
$ vi authorized_keys
                    :
command="" ssh-ed25519 … ←最後の行の先頭に「command="" 」を記入して保存
$
```

(2) 秘密鍵ファイルのコピーとアクセス許可設定
秘密鍵ファイルid_portfwdを、リモート接続を行うPCにコピーし、アクセス許可設定を行う。

秘密鍵ファイル（id_portfwd）のアクセス許可設定（接続を行うPCがLinuxの場合）

```
$ chmod 600 id_portfwd（id_portfwd を所有者のみ読み書き可に設定）
```

秘密鍵ファイル（id_portfwd）のアクセス許可設定（接続を行うPCがWindowsの場合）

```
> cacls id_portfwd /G %USERNAME%:F
（id_portfwd を所有者のみフルコントロール可に設定）
```

(3) ポート転送実行
sshコマンドを実行し、ローカルの33060番ポート[22]をMariaDBホストの3306番ポートへ転送する。

sshによるポート転送実行

```
$ ssh -i id_portfwd -fNL 33060:localhost:3306 <接続先ユーザー名>@<MariaDB ホスト名／IP アドレス>
```

(4) MariaDBサーバーへのリモート接続
mariadbコマンドによりリモート接続を実行する。

mariadbコマンドによるMariaDBサーバーへのリモート接続

```
$ mariadb --protocol=TCP -P 33060 -u <MariaDB ユーザー名> -p
Enter password: ←パスワード入力
```

[22] 使われていない番号であれば、任意の番号でよい。

3.8. ネットワークプログラム用JavaScript環境（Node.js/Node-RED）

3.8.1. Node.jsとNode-REDのインストールと設定

Node.jsはネットワークアプリケーションを構築するためのJavaScript環境で、Webサーバーのような、多くの接続を同時に処理することが求められるアプリケーションの開発に適している。Node-REDは、Node.jsのプログラムを開発するためのビジュアルプログラミングツールであり、その操作はWebブラウザ上で行う。

ここでは、Node-REDのインストールと設定を行う[23]。

(1) Node-REDのインストール

```
# bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-
installers/master/rpm/update-nodejs-and-nodered)

Root user detected. Typically install as a normal user. No need for sudo.

Are you really sure you want to install as root ? (y/N) ? y（root として実行）
Would you like to add Node-RED port 1880 to the firewall public zone ? [y/N] ? N
（Node-RED のポートは公開しない）

This script will do an install of node.js, Node-RED and the service packages to auto-run
Node-RED

Are you really sure you want to do this ? [y/N] ? y（Node-RED インストール実行）

Running nodejs and Node-RED install for user <ユーザー名> at /root on "almalinux"

  Stop Node-RED                ✓
  Install Node.js LTS          ✓    Node v14.19.3     Npm 6.14.17
  Install Node-RED core        ✓    2.2.2
  Add shortcut commands        ✓
  Update systemd script        ✓
  Not adding firewall rule     -


Any errors will be logged to     /var/log/nodered-install.log
All done.
  You can now start Node-RED with the command    node-red-start
  Then point your browser to localhost:1880 or http://{your_ip-address}:1880

Started … - Finished …
```

(2) Node-REDサービスの有効化と起動

[23] Node-REDと同時にNode.jsもインストールされる。

```
# systemctl enable nodered
# systemctl start nodered
```

(3) Node-REDログイン画面の追加

Node-REDの設定情報は、suコマンドによってrootになる前の元のログインユーザーのホームディレクトリ直下に生成される.node-redディレクトリに作成される。元のユーザーに戻って.node-redディレクトリへ移動し、settings.jsファイルを編集してNode-REDログイン画面を追加する。

Node-REDログイン画面の追加

```
# exit（root からログアウト）
$ cd ~/.node-red
$ node-red admin hash-pw（Node-RED ログイン画面用パスワードハッシュ値の生成）
Password: ←パスワード入力
$2b$08$.... ←この文字列（ハッシュ値）をコピー
$ vi settings.js
                    :
（以下の部分をアンコメントして、パスワードの部分にハッシュ値を貼り付け）
    adminAuth: {
            type: "credentials",
            users: [{
                    username: "admin",
                    password: "...",
                    permissions: "*"
            }]
    },
                    :
$ su（root 権限獲得）
Password: ←root のパスワード入力
# systemctl restart nodered（Node-RED サービス再起動）
```

(4) Node-REDログイン画面の確認

ブラウザを起動し、URLに「http://localhost:1880/」を指定して、Node-REDの画面が表示されることと、ユーザー名「admin」に指定したパスワードでログインできることを確認する。

(5) MySQLノードのインストール

MySQLやMariaDBへアクセスするために必要なノード（プログラムモジュール）であるMySQLノードをインストールする。これも.node-redディレクトリで実行する。

MySQLノードのインストール

```
$ cd ~/.node-red（元のログインユーザーで実行）
$ npm install node-red-node-mysql（MySQL ノードのインストール）
npm notice created a lockfile as package-lock.json. You should commit this file.
```

```
+ node-red-node-mysql@1.0.1
added 16 packages from 20 contributors and audited 16 packages in …s
found 0 vulnerabilities
$ systemctl restart nodered（Node-RED サービス再起動）
```

Node-REDのインストールを行った後、dnfコマンドでパッケージのアップデートを行うと、以下のようなメッセージが表示されることがある。

アップデート実行時のメッセージ

```
# dnf -y --nobest update
正しくない設定値: failovermethod=priority in /etc/yum.repos.d/nodesource-el8.repo;  設定: id
"failovermethod"  を伴う  OptionBinding  は存在しません
          :
#
```

放置しても問題はないが、気になる場合には、/etc/yum.repos.d/nodesource-el8.repoファイルを以下のように編集する。

/etc/yum.repos.d/nodesource-el8.repoファイルの編集

```
[nodesource]
                    :
#failovermethod=priority  ←この行をコメントアウト
                    :
[nodesource-source]
                    :
#failovermethod=priority  ←この行をコメントアウト
                    :
```

付録 1. RAID管理

RAID（Redundant Arrays of Inexpensive Disks）とは、複数のディスクを1つのドライブのように扱い、分散処理によるデータ読み書きの高速化や、データ復旧機能の実装による安全化を図る技術である。RAIDの構成形態はいくつかあり、よく使われるものとして、RAID0、RAID1、RAID5が挙げられる。

(1) RAID0（ストライピング）
　　分散処理による高速化
(2) RAID1（ミラーリング）
　　データ複製保存による安全化
(3) RAID5（パリティディスク分散）
　　データおよびエラー訂正＆データ復旧情報の分散保存／処理による高速化と安全化

CentOS 8におけるRAID構成の確認およびハードウェア情報の確認を行う主なコマンドは以下の通り。

RAID構成の確認

```
# df -m (ファイルシステム構成表示、RAID システムは/dev/md***で示される)
# cat /proc/mdstat（RAID 状態表示、故障しているファイルシステムには(F)が表示される）
# mdadm -D /dev/md***（RAID の運用状態詳細表示）
```

ハードウェア情報の確認

```
# cat /proc/cpuinfo（CPU 情報）
# cat /proc/memsinfo（メモリ情報）
# cat /proc/scsi/scsi（ストレージ情報）
# smartctl -a /dev/sd**（ハードディスク型番等情報）
# ls -l /dev/disk/by-id（ハードディスクのモデル名とシリアル番号取得）
# lsblk（ハードディスク構成情報ツリー形式表示）
# blkid -o list（ハードディスクの UUID 情報取得）
# fdisk -l（ハードディスク一覧情報表示, BIOS & MBR でも UEFI & GPT[24]でも使える）
# gdisk -l /dev/sd*（ハードディスク情報表示, GPT の場合）
# badblocks -v -s /dev/sd**（ハードディスク不良セクタチェック）
# fsck -t -y -f -c /dev/sd**（ハードディスクの調査と修復）
```

RAID5におけるハードディスク交換手順の例を以下に示す。

(1) RAID構成確認。

```
# cat /proc/mdstat
        :
    md126 : active raid1 sdc2[2] sda2[0] sdb2[1]（3 つのディスクで正常運転中）
```

[24] BIOS (Basic Input/Output System)とUEFI (Unified Extensible Firmware Interface)は、いずれもOSを起動するための組込ソフトウェア。MBR (Master Boot Record)とGPT (Globally Unique IDentifier Partition Table)は、ディスクパーティション形式を表す。2007年以降のPCでは、UEFI & GPTの構成が主流。

```
        :
    md127 : active raid5 sdc3[3] sda3[0] sdb3[1]  （3 つのディスクで正常運転中）
```

(2) ハードディスクのシリアルナンバーとデバイス名/dev/sd*の対応確認

```
# ls -l /dev/disk/by-id
合計 0
lrwxrwxrwx. 1 root root  9  6 月  7 13:16 ata-<モデル名>_<シリアル番号> -> ../../sda
        :
```

(3) シャットダウンしてハードディスクを1つ除去した後、再起動して構成確認。

```
# cat /proc/mdstat
        :
    md126 : active raid1 sda2[0] sdb2[1]
          1047552 blocks super 1.2 [3/2] [_UU]
      （3 つのディスクのうち 2 つで運転中との表示）
        :
    md127 : active raid5 sda3[0] sdb3[1]
          35629056 blocks super 1.2 level 5, 512k chunk, algorithm 2 [3/2] [_UU]
      （3 つのディスクのうち 2 つで運転中との表示）
# mdadm -D /dev/md126
      （縮退運転（degraded）であること、ディスクが 1 つ除去されたことが示される）
# mdadm -D /dev/md127
      （縮退運転（degraded）であること、ディスクが 1 つ除去されたことが示される）
```

(4) シャットダウンしてハードディスクを1つ追加後、再起動。起動できないときは、既存ディスクの接続ポートを前に移動し、追加ディスクをそれより後のポートに接続してみる。（ブート領域の認識の問題らしい。）

```
# cat /proc/mdstat
        :
    md126 : active raid5 sda3[1] sdb3[3]
          35629056 blocks super 1.2 level 5, 512k chunk, algorithm 2 [3/2] [_UU]
      （3 つのディスクのうち 2 つで運転中との表示）
        :
    md127 : active raid1 sdb2[2] sda2[1]
          1047552 blocks super 1.2 [3/2] [_UU]
      （3 つのディスクのうち 2 つで運転中との表示）
# mdadm -D /dev/md126
      （縮退運転（degraded）であること、ディスクが 1 つ除去されたことが示される）
# mdadm -D /dev/md127
      （縮退運転（degraded）であること、ディスクが 1 つ除去されたことが示される）
```

(5) ハードディスク一覧情報を表示。シリアル番号に基づいて、新しく追加したディスクのデバイス名（例えば/dev/sdc）を確認する。

```
# ls -l /dev/disk/by-id
```

(6) 既存ディスクの構成を参考にして、新しく追加したディスクにパーティション作成。

```
# fdisk /dev/sdc（UEFI の場合には、gdisk /dev/sdc を使う。コマンドはほぼ同じ。）
  コマンド: n（新規作成）
  Partition type:
  Select: p（基本パーティション）
  パーティション番号: 1（1 から順番に付与）
  最初: 2048（既存ディスクと同じ）
  Last sector: 4204543（既存ディスクと同じ）
        :
        ＜以下同様に必要な分パーティション作成＞
        :
  コマンド: t（システム ID 設定）
  パーティション番号: 1（作成したパーティションの番号）
  Hex Code: 82（Linux スワップ ID 指定。gdisk の場合は 8200）
        :
  Hex Code: fd（RAID システム ID 指定。gdisk の場合は FD00）
        :
  コマンド: a（ブート可フラグ設定。gdisk の場合は不要）
  パーティション番号: 2（/boot パーティションの番号）
  コマンド: p（設定内容確認）
  コマンド: w（設定内容をハードディスクへ書き込み）
# partprobe（パーティション書き込みに失敗した場合のみ実行）
```

(7) 新しく作成したパーティションをRAIDに追加。

```
# mdadm --manage /dev/md126 --add /dev/sdc3（RAID へ組み込み）
# mdadm --manage /dev/md127 --add /dev/sdc2（RAID へ組み込み）
# watch cat /proc/mdstat（組み込み進捗状況の確認）
```

(8) 現在割り当てられているスワップ領域のUUIDを確認。

```
# blkid -o list
device          fs_type         label   mount point     UUID
------------------------------------------------------------------------------------------------
        :
/dev/sda1       swap                    <swap>          <スワップ領域/dev/sda1 の UUID>
        :                                               （この UUID をメモ）
/dev/sdb1       swap                    <swap>          <スワップ領域/dev/sdb1 の UUID>
        :                                               （この UUID をメモ）
/dev/sdc1       swap                    not mounted     …
        :
```

(9) 新しく作成したスワップ領域を割り当て。

```
# mkswap /dev/sdc1
        :
```

```
   … UUID=<新しく作成したスワップ領域の UUID>
# swapon /dev/sdc1
```

(10)新しく作成したスワップ領域の登録。
　　/etc/fstabを編集し、新しく作成したスワップ領域を登録する。

/etc/fstabの編集による新規作成スワップ領域の登録

```
# vi /etc/fstab
          :
UUID=….       swap   swap   …
（(8)でメモした UUID のどれでもない swap の UUID を、新しく作成したスワップ領域の
UUID で書き換え）
UUID=….       swap   swap   …
UUID=….       swap   swap   …
          :
```

(11)追加したディスクへのブートローダーインストール（UEFIの場合には不要）

```
# grub2-install /dev/sdc
```

　ホットスワップ（シャットダウンせずにハードディスクを交換すること）を行う場合には、実際に取り外す前に、コマンドで不良指定と取り外し処理をしておくこと。

```
# mdadm --manage /dev/md126 --fail /dev/sda3（不良マーク付与）
# mdadm --manage /dev/md127 --fail /dev/sda2（不良マーク付与）
# mdadm --manage /dev/md126 --remove /dev/sda3（RAID から不良パーティションを削除）
# mdadm --manage /dev/md127 --remove /dev/sda2（RAID から不良パーティションを削除）
```

付録 2. ファイアウォール設定

ファイアウォールの設定は、ゾーンと呼ばれるグループごとに行われる。各ゾーンでは、アクセス許可の内容の他、その内容が適用されるNIC（Network Interface Card）が記述される。ファイアウォール関連でよく使うコマンドは以下の通り。

(1) ファイアウォールサービスの起動／停止／再起動／状態表示

```
# systemctl [start/stop/restart/status] firewalld.service
```

(2) 標準のゾーン名を取得

```
# firewall-cmd --get-default
```

(3) 有効化されているゾーン名を取得

```
# firewall-cmd --get-active
```

(4) 指定したゾーンの設定状態を表示（--zoneオプションなしの場合は標準ゾーン）

```
# firewall-cmd --list-all --zone=<ゾーン名>
```

(5) すべてのゾーンの設定状態を表示

```
# firewall-cmd --list-all-zones
```

(6) 登録可能なサービスの一覧表示

```
# firewall-cmd --get-services
```

(7) 登録されているサービスの一覧表示（--zoneオプションなしの場合は標準ゾーン）

```
# firewall-cmd --list-services --zone=<ゾーン名>
```

(8) サービスの登録追加（--zoneオプションなしの場合は標準ゾーンに追加）

```
# firewall-cmd --add-service=<サービス名> --zone=<ゾーン名>
```

(9) サービスの登録削除（--zoneオプションなしの場合は標準ゾーンから削除）

```
# firewall-cmd --remove-service=<サービス名> --zone=<ゾーン名>
```

(10)登録されているポートの一覧表示（--zoneオプションなしの場合は標準ゾーン）

```
# firewall-cmd --list-ports --zone=<ゾーン名>
```

(11)ポートの登録追加（--zoneオプションなしの場合は標準ゾーンに追加）

```
# firewall-cmd --add-port=<ポート番号>/[tcp/udp] --zone=<ゾーン名>
```

(12)ポートの登録削除（--zoneオプションなしの場合は標準ゾーンから削除）

```
# firewall-cmd --remove-port=<ポート番号>/[tcp/udp] --zone=<ゾーン名>
```

(13)登録されているIPアドレスの一覧表示（--zoneオプションなしの場合は標準ゾーン）

```
# firewall-cmd --list-sources --zone=<ゾーン名>
```

(14)IPアドレスの登録追加（--zoneオプションなしの場合は標準ゾーンに追加）

```
# firewall-cmd --add-source=<IP アドレス> --zone=<ゾーン名>
```

IPアドレスは、例えば”192.168.0.0/16”のようにマスクを指定することで、範囲指定を行える。マスク指定は、2進数表現の場合に一致する桁数を表す。この場合は、2進数表記で左から16桁まで一致するIPアドレス指定となるので、192.168.0.0～192.168.255.255を指定したことになる。

(15)IPアドレスの登録削除（--zoneオプションなしの場合は標準ゾーンから削除）

```
# firewall-cmd --remove-source=<IP アドレス> --zone=<ゾーン名>
```

ゾーン名をdropとすると、アクセス拒否設定になる。

(16)特定のIPアドレスから特定のサービスへのアクセス許可登録追加（--zoneオプションなしの場合は標準ゾーンに追加）

```
# firewall-cmd --add-rich-rule=’rule family=ipv4 source address=<IP アドレス> service name=<サービス名> accept’ --zone=<ゾーン名>
```

(17)特定のIPアドレスから特定のサービスへのアクセス許可登録削除（--zoneオプションなしの場合は標準ゾーンから削除）

```
# firewall-cmd --remove--rich-rule=’rule family=ipv4 source address=<IP アドレス> service name=<サービス名> accept’ --zone=<ゾーン名>
```

(18)特定のIPアドレスから特定のポートへのアクセス許可登録追加（--zoneオプションなしの場合は標準ゾーンに追加）

```
# firewall-cmd --add-rich-rule=’rule family=ipv4 source address=<IP アドレス> port port=<ポート番号> protocol=[tcp/udp] accept’ --zone=<ゾーン名>
```

(19)特定のIPアドレスから特定のポートへのアクセス許可登録削除（--zoneオプションなしの場合は標準ゾーンから削除）

```
# firewall-cmd --remove--rich-rule=’rule family=ipv4 source address=<IP アドレス> port port=<ポート番号> protocol=[tcp/udp] accept’ --zone=<ゾーン名>
```

(20)修正の恒久的な反映

```
# firewall-cmd <コマンド> --permanent
# firewall-cmd --reload
```

(21)ping（ICMP）の遮断

```
# firewall-cmd --set-target=DROP --permanent
# firewall-cmd --reload
```

(22)ping（ICMP）処理を標準に設定

```
# firewall-cmd --set-target=default --permanent
# firewall-cmd --reload
```

付録 3. OpenSSHによる認証および遠隔操作

付録 3.1. 公開鍵認証と暗号化通信

クライアントに秘密鍵ファイル、サーバーにそれと対応する公開鍵ファイルを保存する。サーバーへ接続するとき、クライアントでは秘密鍵ファイルを使って電子署名を作成し、サーバーへ送信する。サーバーでは、公開鍵ファイルを使って、送られてきた電子署名を検証する。電子署名が正しいと判断されれば、クライアントとサーバー間の通信が確立し、共通鍵方式による暗号化通信が行われる。この時の共通鍵は一時的なもので、通信が終了すれば破棄される。

公開鍵情報は、サーバーの接続先ユーザーのホームディレクトリに作られる.sshディレクトリの中のauthorized_keysファイルに記述する。各行が1つの公開鍵に対応しており、複数の公開鍵を登録することができる。

本手順書では、サーバーで公開鍵と秘密鍵を作成し、秘密鍵をクライアントへコピーするという手順を紹介した。一方、クライアントで公開鍵と秘密鍵を作成し、公開鍵のみをサーバーへ送って登録することもできる。

付録 3.2. OpenSSHでよく使うコマンドおよび設定

(1) ssh サービスの起動／停止／再起動（CentOS の場合）

```
# systemctl [start/stop/restart] sshd
```

(2) ssh によるリモート接続先

```
$ ssh -i <秘密鍵ファイル名> [-v] <接続先ユーザー名>@<サーバー名>
（-v オプションを付けると、サーバーとクライアント間のやり取りを見ることができる。）
```

-v オプションによる出力の抜粋

```
debug1: Local version string SSH-2.0-OpenSSH_7.4
debug1: Remote protocol version 2.0, remote software version OpenSSH_8.0
（ローカルおよびリモートの ssh のバージョン）
debug1: match: OpenSSH_8.0 pat OpenSSH* compat 0x04000000
（通信プロトコルのバージョン決定）
debug1: kex: algorithm: curve25519-sha256
debug1: kex: host key algorithm: ecdsa-sha2-nistp256
（共通鍵を交換するためのアルゴリズムの決定）
debug1: kex: server->client cipher: chacha20-poly1305@openssh.com MAC: <implicit> …
debug1: kex: client->server cipher: chacha20-poly1305@openssh.com MAC: <implicit> …
（共通鍵暗号化アルゴリズム＆認証方式の決定）
```

(3) ssh による接続先コマンドの実行

```
$ ssh -i <秘密鍵ファイル名> <接続先ユーザー名>@<サーバー名> <コマンド>
```

(4) scp コマンドによる接続先との間のファイルコピー

```
$ scp -i <秘密鍵ファイル名> <コピーするファイル名>
<接続先ユーザー名>@<サーバー名>:<コピー先ディレクトリ名／ファイル名>（絶対パス／ホ
```

ームディレクトリからの相対パス）
$ scp -i <秘密鍵ファイル名> <接続先ユーザー名>@<サーバー名>:<コピーするファイル名>
<コピー先ディレクトリ名／ファイル名>（絶対パス／現在のディレクトリからの相対パス）
接続先ファイルを指定する部分の「:」の前後にはスペースを入れないことに注意。

(5) .ssh/authorized_keys へのオプション記述による接続元や実行コマンドの制限
from="<IP アドレス 1>,<IP アドレス 2>,…" ssh-ed25519 …（接続元 IP アドレスの制限）
command="<コマンド 1>,<コマンド 2>,…" ssh-ed25519 …（実行コマンドの制限）
from="…", command="…", … ssh-ed25519 …（複数オプションの組み合せ）
オプションは行ごと（公開鍵ごと）に設定。

付録 3.3. Windows 10/11におけるOpenSSHサーバーの設定

CentOS 8の場合、最小構成のインストールでもOpenSSHサーバーが設定されている。Windows 10/11 PCをOpenSSHサーバーとして使用する場合は、Windows 10設定画面から、もしくは、OpenSSHパッケージのダウンロードにより、インストールとサービス化を行う。（クライアントは、最初からインストールされている。）

なお、管理者権限を持ったユーザーへのssh接続は行えないことに注意。

■ Windows 10（1809 以降）の設定画面からインストールする方法

(1) 画面左下のスタートアイコンから、「設定」-「アプリ」-「アプリと機能」-「オプション機能」-「機能の追加」を選択する。
(2) 「OpenSSH サーバー」をクリックし、表示された「インストール」ボタンをクリックする。
(3) Windows 10 のスタートメニューから、「Windows 管理ツール」-「サービス」を選択する。
(4) 表示された画面の一覧から「OpenSSH SSH Server」を選択し、右クリックで表示されたメニューから、「プロパティ」を選択する。
(5) 「スタートアップの種類」から、「自動」あるいは「自動 (遅延開始)」を選択する。Windows 開始時に、OpenSSH サーバーが自動で起動するようになる。
(6) 「サービスの状態」の中の、「開始」ボタンをクリックする。OpenSSH サーバーが起動する。サーバー設定ファイル（sshd_config）は、C:¥ProgramData¥ssh フォルダに作成される。

■ Windows 11 の設定画面からインストールする方法

(1) 画面左下のスタートアイコンから、「設定」-「アプリ」-「オプション機能」-「オプション機能を追加する」-「機能を表示」を選択する。
(2) 「OpenSSH サーバー」を選択し、表示された「次へ」ボタンをクリックする。
(3) 「インストール」ボタンをクリックする。
(4) Windows 11 のスタートメニューから、「すべてのアプリ」-「Windows ツール」-「サービス」を選択する。
(5) 表示された画面の一覧から「OpenSSH SSH Server」を選択し、右クリックで表示されたメニューから、「プロパティ」を選択する。

(6) 「スタートアップの種類」から、「自動」あるいは「自動 (遅延開始)」を選択する。Windows 開始時に、OpenSSH サーバーが自動で起動するようになる。
(7) 「サービスの状態」の中の、「開始」ボタンをクリックする。OpenSSH サーバーが起動する。サーバー設定ファイル（sshd_config）は、C:¥ProgramData¥ssh フォルダに作成される。

■ OpenSSH パッケージダウンロードによりインストールする方法
(1) https://github.com/PowerShell/Win32-OpenSSH/releases より Powershell/Win32-OpenSSH の ZIP ファイルをダウンロードし、適当な場所に展開する。展開先のパスには、空白文字や日本語文字を含まないようにする。ここでは、C:¥OpenSSH に展開するものとする。
(2) スタートメニューから Windows PowerShell>Windows PowerShell とたどり、右クリック>その他>管理者として実行を選択し、PowerShell を起動する。
(3) OpenSSH の展開先フォルダ（この場合は C:¥OpenSSH）へ移動する。PowerShell の実行ポリシーの確認と設定を行い、インストールスクリプトを実行する。

```
> cd C:¥OpenSSH（展開先フォルダへ移動）
> Get-ExecutionPolicy（実行ポリシーの表示）
Restricted（実行禁止）
> Set-ExecutionPolicy RemoteSigned（実行ポリシーを、'リモートは署名が必要'に変更）
実行ポリシーの変更
        :       （変更することの確認を求められる。Y（はい）と入力）
> .¥install-sshd.ps1（インストールスクリプト実行）
[SC] SetServiceObjectSecurity SUCCESS
[SC] ChangeServiceConfig2 SUCCESS
[SC] ChangeServiceConfig2 SUCCESS
sshd and ssh-agent services successfully installed（インストール完了）
```

(4) ホストキーの生成

```
> mkdir C:¥ProgramData¥ssh
> .¥ssh-keygen -A
```

ホストキーファイルは、C:¥ProgramData¥ssh フォルダに生成される。
(5) ホストキーアクセス許可の設定

```
> .¥FixHostFilePermissions.ps1
        :       （確認の求めに対してすべて Y（はい）と入力）
```

(6) Windows 10 のスタートメニューから、「Windows 管理ツール」-「サービス」を選択する。
(7) 表示された画面の一覧から「OpenSSH SSH Server」を選択し、右クリックで表示されたメニューから、「プロパティ」を選択する。
(8) 「スタートアップの種類」から、「自動」あるいは「自動 (遅延開始)」を選択する。Windows 開始時に、OpenSSH サーバーが自動で起動するようになる。
(9) 「サービスの状態」の中の、「開始」ボタンをクリックする。OpenSSH サーバーが起動する。サーバー設定ファイル（sshd_config）は、C:¥ProgramData¥ssh フォルダに作成される。

さらに以下の手順により、Windowsのファイアウォールに、OpenSSHサーバー通信用

の規則を追加する。

(1) スタートメニューから Windows システムツール>コントロールパネル>Windows Defender ファイアウォールと選択する。
(2) 画面左側の「詳細設定」をクリックする。
(3) 画面左上側の「受信の規則」を選択した後、画面右上側の「新しい規則...」を選択する。
(4) 「規則の種類」で「ポート」を選択する。
(5) 「プロトコルおよびポート」で「TCP」と「特定のローカルポート」を選択し、「22」と記入する。
(6) 「操作」で「接続を許可する」を選択する。
(7) 「プロファイル」で「ドメイン」、「プライベート」、「パブリック」のすべてを選択する。
(8) 「名前」で「OpenSSH」と記入する。（名前は任意で良い。）

■　公開鍵の登録とアクセス許可設定

sshd_configファイル（C:¥ProgramData¥sshフォルダに作られる）の設定、ならびに公開鍵と秘密鍵の生成は、それぞれ2.4.1と2.4.2に述べた手順で行える。C:¥ProgramDataフォルダは、標準では見えないことに注意。エクスプローラの「表示」タブから、「隠しファイル」にチェックを入れると見えるようになる。

公開鍵の登録とアクセス許可設定は、コマンドプロンプトより以下のコマンドを入力して行う。

```
> cd %USERPROFILE%¥.ssh
> type id_ed25519.pub >> authorized_keys
> cacls authorized_keys /G %USERNAME%:F
（authorized_keys を所有者のみフルコントロール可に設定）
> cacls authorized_keys /E /G SYSTEM:F
（authorized_keys を Windows システムもフルコントロール可に設定）
```

付録 4. cronによるコマンドの定期実行

コマンドを定期実行させる方法は、crontabコマンドを使う方法と、/etc/cron.dディレクトリに定期実行用スクリプトを保存する方法の2通りある。

(1) crontabコマンドを使う方法

本文中でも紹介した方法で、以下のように実行する。これは、dnfによるパッケージ更新の設定例を示している。

```
# crontab -e（cron 設定ファイル編集画面(vi エディタ)表示）
（実行時期とコマンドを記述）
0 1 * * Sun /usr/bin/dnf -y –nobest update
（0 分 1 時毎週日曜日に/usr/bin/dnf -y update を実行）
crontab: installing new crontab（編集内容の保存完了メッセージ表示）
```

cron設定書式は以下の通り（半角スペース区切り）。

1番目: 実行日時の分（上の場合は0分）*/nとすることで、n分間隔指定も可
2番目: 実行日時の時（上の場合は1時）*/nとすることで、n時間間隔指定も可
3番目: 実行日時の日（上の場合は指定なし）*/nとすることで、n日間隔指定も可
4番目: 実行日時の月（上の場合は指定なし）*/nとすることで、n月間隔指定も可
5番目: 曜日（上の場合は日曜日）数字指定も可（0から6が日から土に対応）
6番目以降: 実行コマンド

cron設定情報は、ユーザーごとに分けられたファイルとして、/var/spool/cronディレクトリに保存される。

(2) /etc/cron.dディレクトリに定期実行用スクリプトを保存する方法

以下のようにして、/etc/cron.dディレクトリに定時実行用のスクリプト（テキストファイル）を作成する。これは、dnfによるパッケージ更新の設定例を示している。

```
# cd /etc/cron.d
# vi dnf.sh（vi エディタ起動）
（実行時期、実行ユーザーとコマンドを記述）
0 1 * * Sun root /usr/bin/dnf -y --nobest update
（0 分 1 時毎週日曜日に root 権限で/usr/bin/dnf -y --nobest update を実行）
# chmod a+x dnf.sh（保存後、実行権限を設定）
```

cron設定書式は以下の通り（半角スペース区切り）。

1番目: 実行日時の分（上の場合は0分）*/nとすることで、n分間隔指定も可
2番目: 実行日時の時（上の場合は1時）*/nとすることで、n時間間隔指定も可
3番目: 実行日時の日（上の場合は指定なし）*/nとすることで、n日間隔指定も可
4番目: 実行日時の月（上の場合は指定なし）*/nとすることで、n月間隔指定も可
5番目: 曜日（上の場合は日曜日）数字指定も可（0から6が日から土に対応）
6番目: 実行ユーザー
7番目以降: 実行コマンド

付録 5. LANで運用するローカルなサーバーのHTTPS化

LAN内のプライベートIPアドレスで運用するApache HTTPサーバーをHTTPS化する場合には、x509.v3のsubjectAltName（SAN）拡張を利用し、IPを用いた自己署名証明書を発行して導入する。その手順を以下に記す。なお、このサーバーへブラウザからアクセスした場合には、「自己署名証明書なので信用できない」といった内容の警告が出る。そのときには、信用できる証明書あるいはサイトとしてブラウザに登録する。

(1) mod_sslのインストール

```
# dnf -y install mod_ssl
```

(2) httpsサービスの通信許可設定

```
# firewall-cmd --add-service=https --permanent
# firewall-cmd --reload
```

(3) サーバー用の秘密鍵の作成

```
# openssl genrsa -out localhost.key 2048（数値は鍵のビット数。デフォルトは 512）
Generating RSA private key, 2048 bit long modulus
...+++
.....................................+++
e is 65537 (0x10001)
#
```

(4) CA（Certification Authority）へのCSR（Certificate Signing Request）ファイル作成

```
# openssl req -new -key localhost.key -out localhost.csr
          :
Country Name (2 letter code) [XX]: ←適当なものを入力
State or Province Name (full name) []: ←適当なものを入力
Locality Name (eg, city) [Default City]: ←適当なものを入力
Organization Name (eg, company) [Default Company Ltd]: ←適当なものを入力
Organizational Unit Name (eg, section) []: ←適当なものを入力
Common Name (eg, your name or your server's hostname) []: ←適当なものを入力
Email Address []: ←適当なものを入力

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []: ←適当なものを入力
An optional company name []: ←適当なものを入力
#
```

(5) 自己署名のサーバー証明書（CRTファイル）の作成

```
# openssl x509 -days 3650 -in localhost.csr -out localhost.crt -req -signkey localhost.key
（数値”3650”は証明書の有効日数。デフォルトは 30 日。）
Signature ok
```

```
subject=/C=jp/L=Default City/O=Default Company Ltd
Getting Private key
#
```

(6) Apache HTTPサーバーへの証明書の導入

```
# mv localhost.key /etc/pki/tls/private（秘密鍵の移動）
# mv localhost.crt /etc/pki/tls/certs（サーバー証明書の移動）
# vi /etc/httpd/conf.d/ssl.conf（秘密鍵とサーバー証明書の登録）
                    :
#ServerName www.example.com:443 ←この行をコメントアウト
                    :
SSLCertificateFile /etc/pki/tls/certs/localhost.crt ←この行をアンコメント
                    :
SSLCertificateKeyFile /etc/pki/tls/private/localhost.key ←この行をアンコメント
                    :
#
```

SELinux（5ページ）を有効化している場合には、以下の設定が必要な場合もある。

SELinuxを有効化した場合の追加設定

```
# setsebool -P httpd_read_user_content 1（httpd によるファイルアクセス許可）
# reboot（OS 再起動による SELinux 設定の反映）
```

(7) リダイレクト設定

新規に/etc/httpd/conf.d/san-redirect.confファイルを作成してhttpからhttpsへのリダイレクト設定を記述する。

/etc/httpd/conf.d/san-redirect.confの作成

```
# vi /etc/httpd/conf.d/san-redirect.conf
<VirtualHost _default_:80>
ServerSignature Off

RewriteEngine On
RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]

ErrorLog /var/log/httpd/redirect.error.log
LogLevel warn
</VirtualHost>
#
```

(8) Apache HTTPサーバーの再起動

```
# systemctl restart httpd
```

(9) 自己署名のサーバー証明書を元に、Tomcat用p12証明書ファイルを作成

```
# openssl pkcs12 -export -inkey localhost.key -in localhost.crt -out localhost.p12
Enter Export Password: ←p12 証明書用パスワード設定
Verifying - Enter Export Password: ←p12 証明書用パスワード確認再入力
#
```

(10)Tomcatへのp12証明書ファイル導入

```
# chown tomcat:tomcat localhost.p12 ←証明書ファイルの所有者を tomcat に変更
# mv localhost.p12 /usr/local/tomcat/conf
#
```

(11)Tomcatポートの通信許可設定とSSL/TLS通信機能の有効化

ポートの通信許可設定

```
# firewall-cmd --add-port=8443/tcp --permanent
# firewall-cmd --reload
```

/usr/local/tomcat/conf/server.xmlの編集によるSSL/TLS通信機能有効化

```
    <!-- ←この行を削除（コメントを外す）
    <Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
               maxThreads="150" SSLEnabled="true">
        <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
        <SSLHostConfig>
            <Certificate certificateKeystoreFile="conf/localhost-rsa.jks"
                         type="RSA" />
        </SSLHostConfig>
    </Connector>
    --> ←この行を削除（コメントを外す）
```

コメントを外して、以下のように編集する。

```
    <Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
               maxThreads="150" SSLEnabled="true">
        <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
        <SSLHostConfig>
            <Certificate certificateKeystoreFile="/usr/local/tomcat/conf/localhost.p12"
                         certificateKeystorePassword="< p12 証明書用パスワード>"
                         type="RSA" />
        </SSLHostConfig>
    </Connector>
```

あとは、「3.6.2 TomcatのHTTPS化」の(2)と同様にして、Tomcat Native Libraryをインストールする。

付録 6. バックアップ用外付けHDDの接続

外付けのHDDをUSBで接続した場合、通常、そのHDDは自動で認識され、ファイル管理ソフトを使ってコピーなどのファイル操作を行うことができる。しかしながら、ファイル管理ソフトの画面上での操作は、自動化するには不向きである。

ここでは、接続したHDDを指定したディレクトリに自動で配置（マウント）し、rsyncコマンドを用いてバックアップを行う方法を紹介する。

なお、HDDのフォーマットはexFAT[25]とする。

(1) exFAT読み書き用パッケージのインストール

```
# dnf -y install https://forensics.cert.org/cert-forensics-tools-release-el8.rpm
# dnf -y --nobest install exfat-utils
```

(2) HDDマウント先ディレクトリの作成

HDDをマウントするディレクトリを作成する。ここでは、/mnt/backupディレクトリへマウントするものとする。

HDDマウント先ディレクトリ（/mnt/backup）の作成

```
# mkdir /mnt/backup
```

(3) HDDの接続とUUIDの確認

HDDを接続し、blkidコマンドによりデバイス名とUUIDを確認する。

blkidコマンドによるデバイス名とUUIDの確認

```
# blkid -o list
device          fs_type         label    mount point    UUID
------------------------------------------------------------------------------------------------
        :
/dev/sd**       exfat                    ....           <HDD の UUID>
        :
```

(4) HDDマウント先の登録とOS再起動

/etc/fstabファイルを編集し、HDDマウント先を登録する。

/etc/fstabファイルの編集によるHDDマウント先の登録

```
# vi /etc/fstab
                :
UUID=<HDD の UUID>      /mnt/backup    exfat    rw,nofail       0       0 ←この行を追加
# reboot
```

(5) HDDマウントの確認

---

[25] マイクロソフトが導入したファイルシステムで、SDカードのフォーマットとして広く使われている。Windows、Mac、Linuxのいずれでも読み書き可能である。

blkidコマンドにより、HDDが/mnt/backupにマウントされていることを確認する。外しておいたHDDをOS起動後に接続した場合も、同じように/mnt/backupにマウントされる。

blkidコマンドによるHDDマウントの確認

```
# blkid -o list
device          fs_type         label   mount point    UUID
-------------------------------------------------------------------------------------------------
        :
/dev/sd**       exfat                   /mnt/backup            <HDD の UUID>
        :
```

(6) バックアップの実行

rsyncコマンドによりバックアップを実行する。ここでは、/home、/var/www/html、/usr/local/tomcat/webappsのバックアップを取る例を示す。

rsyncコマンドによるバックアップ実行例

```
# rsync -rlptD --delete /home /var/www/html /usr/local/tomcat/webapps /mnt/backup
```

rsyncコマンドの主なオプションは以下の通り。

```
-a, --archive               アーカイブモード(-rlptgoD オプションと同義)
-r, --recursive             ディレクトリで再帰的に実行する
-l, --links                 ソフトリンクを維持する
-p, --perms                 パーミッションを維持する
-t, --times                 タイムスタンプを維持する
-g, --group                 グループを維持する
-o, --owner                 オーナーを維持する (rootのみ)
-D, --devices               デバイスを維持する (rootのみ)
-u, --update                アップデートのみ許可 (上書き禁止)
-n, --dry-run               実行時の動作だけを表示
    --delete                送信側にないファイルを削除
    --timeout=TIME          IO タイムアウトを設定(秒)
-z, --compress              受信ファイルを圧縮compress file data
  --exclude=PATTERN パターン一致するファイルを除外
  --exclude-from=FILE ファイルに記述されたパターンと一致するファイルを除外
```

ただし、バックアップ先のフォーマットがFATやexFATの場合には、rootであってもグループやオーナーの操作を行えないため、-gオプションや-oオプションを指定するとエラーが発生する。また、シンボリックリンクはサポートされていないためにスキップされ、-lオプションを付けた場合にはシンボリックを検知したときにエラーメッセージが表示される。

付録 7. SELinuxに起因するプロセス動作制限の簡易的な解決法

SELinuxによりプロセスの動作を拒否されることが時折発生する[26]。ここでは、SELinuxのログを読み込んで解決策を表示するコマンドaudit2allowを用いて、そのような動作拒否を解決する簡易的な方法を紹介する。

(1) 動作が制限されているプロセスの確認
/var/log/audit/audit.logから、SELinuxによる動作拒否を示す行を抽出する。

/SELinuxによる動作拒否を示す行の抽出

```
# grep denied /var/log/audit/audit.log
```

(2) audit2allowによる解決策案の表示

```
# audit2allow -a -l -m mylocal（"mylocal"の部分は適当な名前でよい）
module mylocal 1.0;
        :
#!!!! This avc can be allowed using the boolean '<パラメータ名>'
        :
```

「This avc can be allowed using the boolean '<パラメータ名>'」と表示された場合には、そのパラメータを1に設定することによって解決される可能性がある。setseboolコマンドで、値を1に設定してOSを再起動する。

```
# setsebool -P <パラメータ名> 1
# reboot
```

(3) モジュールパッケージの作成とインストール
SELinuxパラメータ値の設定で解決しない場合には、モジュールパッケージの作成とインストールを行う。

モジュールパッケージの作成とインストール

```
# audit2allow -M mylocal < /var/log/audit/audit.log
（許可ルール定義ファイル mylocal.te とモジュールパッケージ mylocal.pp の出力）
****************** 重要 **********************
  このポリシーパッケージを有効にするには、以下を実行して下さい:

  semodule -i mylocal.pp
# semodule -i mylocal.pp（モジュールパッケージのインストール）
```

semoduleコマンドの主なオプションは以下の通り。
-l　　モジュールの表示

[26] いったんSELinuxを無効化して再起動し、再度SELinuxを有効化して起動し直すと、SELinuxのラベル付けが更新されて、プロセスの動作が拒否されなくなることもある。

-r <モジュール名>	モジュールの削除
-d <モジュール名>	モジュールを削除せずに無効化
-u <モジュールパッケージ>	モジュールの更新

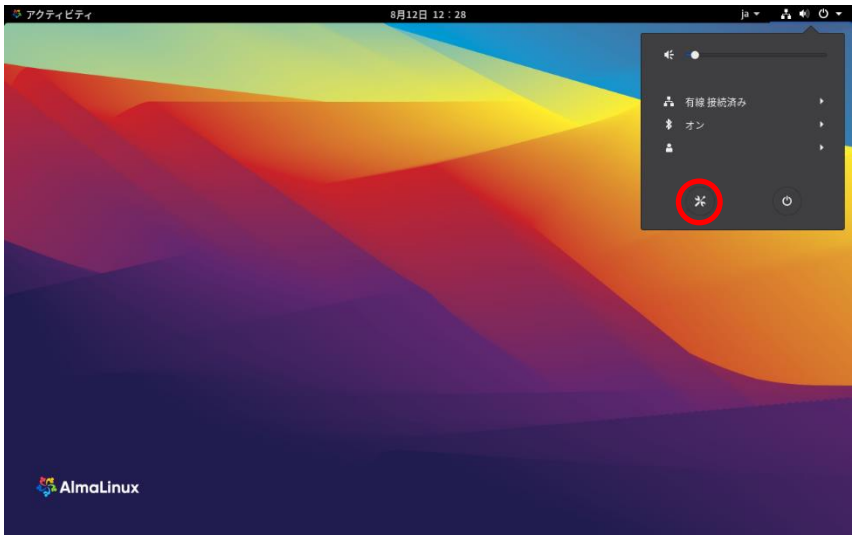
付録 8. L2TP (Layer 2 Tunneling Protocol) VPN接続の設定

NetworkManager-l2tp VPNプラグインをインストールすることにより、NetworkManagerを用いてL2TP (Layer 2 Tunneling Protcol) VPN接続を行えるようになる。

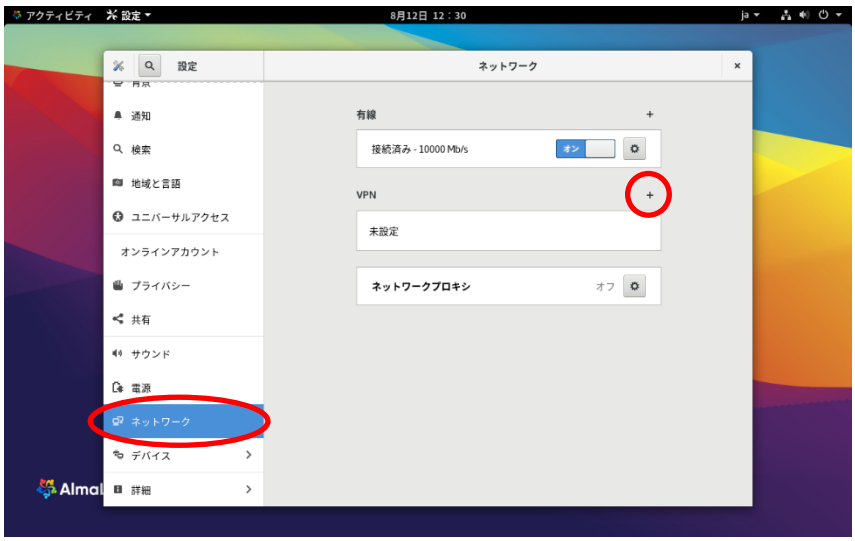
(1) NetworkManager-l2tp VPNプラグインのインストール

```
# dnf -y install xl2tpd NetworkManager-l2tp NetworkManager-l2tp-gnome
```

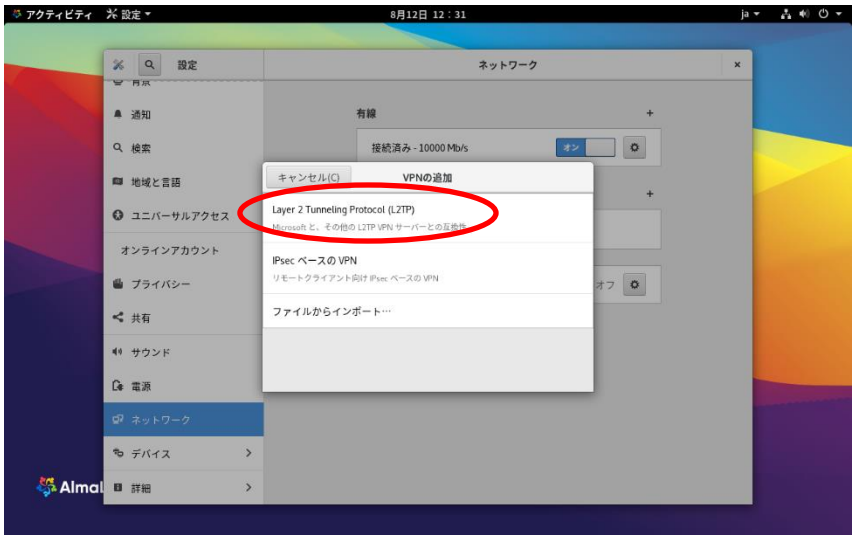
(2) 設定画面の表示



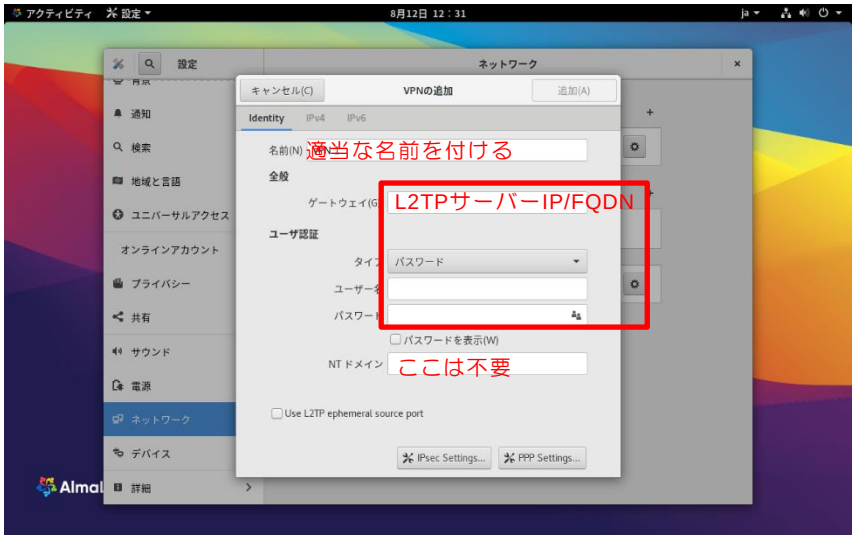
(3) VPN接続追加画面の表示



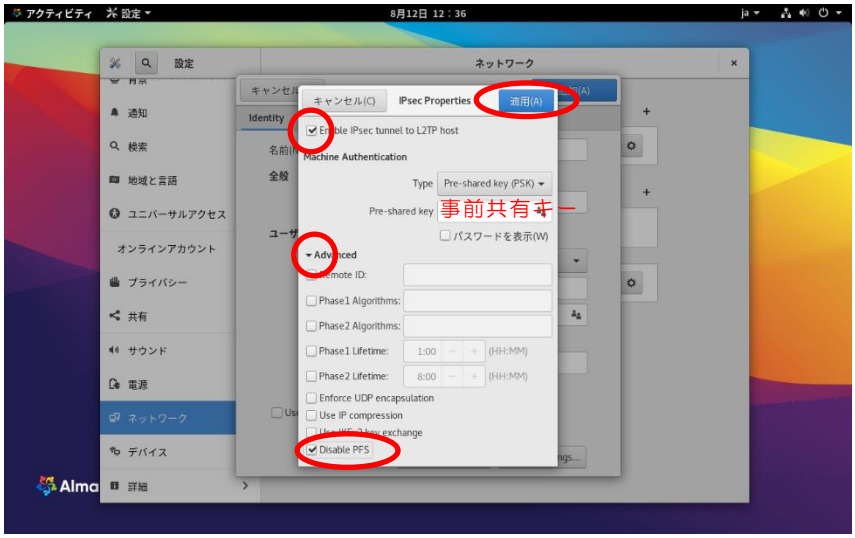
(4) Layer 2 Tunneling Protocol (L2TP)オプションの選択



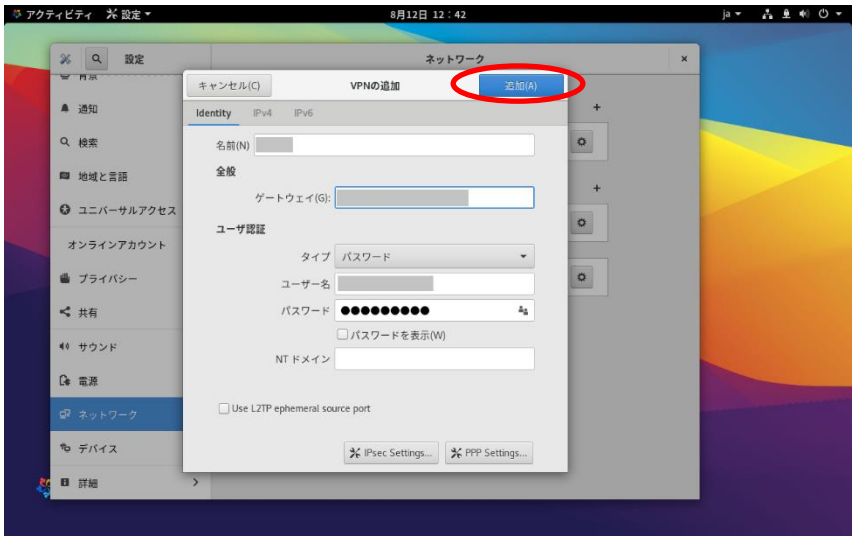
(5) VPN接続詳細情報の設定



(6) IPsec設定



(7) VPN接続の追加



(8) VPN接続オン