

NTT東日本 – IPA シン・テレワークシステム Doc001.

プライベート版 構築マニュアル (スタンドアロン版)



未公開・
内部利用のみ

バージョン: **beta7preview11**

日付: 2021/01/26 (登)



NTT東日本
特 殊 局



独立行政法人
情報処理推進機構
産業サイバーセキュリティセンター
サイバー技術研究室

本バージョンについて・更新履歴

- 本ドキュメントは、以下のバージョンの「NTT東日本 – IPA シン・テレワークシステム」のソースコード一式に関する解説書です。
 - バージョン: [beta7preview11](#)
- バージョンごとの更新履歴・機能追加・バグ修正の履歴は、以下の URL のテキストファイル (README.md) に記載されています。
本ドキュメントをお読みいただく前に、必ずご参照ください。
 - <https://github.com/IPA-CyberLab/IPA-DN-Ultra/>



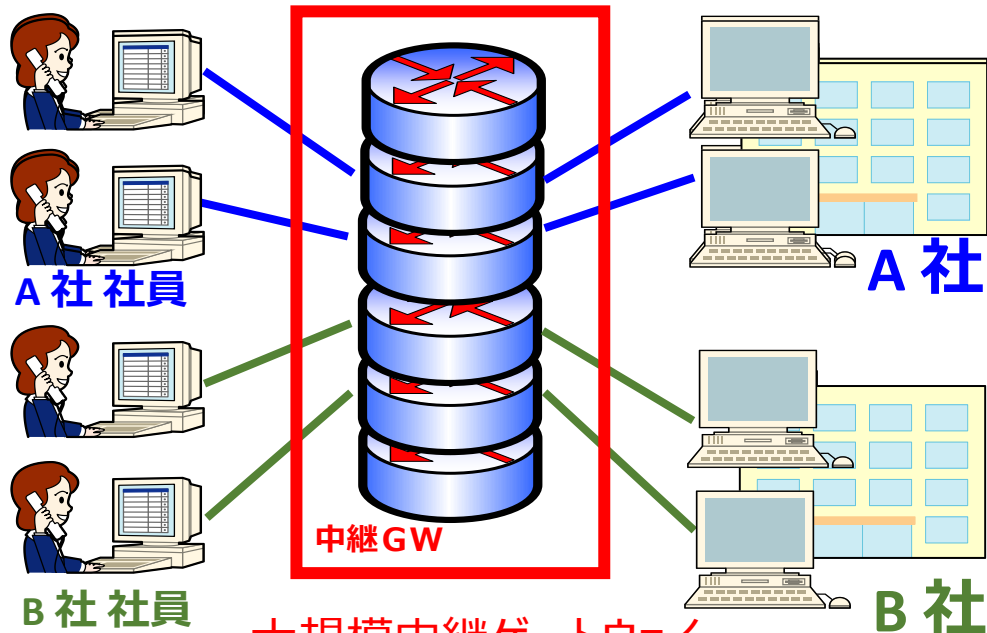
1. はじめに

- ソフトウェアの本バージョンは、現在未公開のソースコード・プレビュー版であり、本ドキュメントの配布先組織内における評価を目的として提供するものです。
- 本ドキュメントおよび本ドキュメントに記載されているクレデンシャル (認証情報) を用いて入手・ビルドしたプログラムの組織外への配布は、ご遠慮ください。
- IPA サイバー技術研究室では、シン・テレワークシステムの中継ゲートウェイの動作状況の統計データを収集しています。
シン・テレワークシステムの中継ゲートウェイは、統計情報を IPA の統計サーバーに送付します。
統計情報は、シン・テレワークシステムが日本国内でどの程度活用されているかを確認し、IPA サイバー技術研究室の活動に関する社会的理解や必要な予算の獲得等のために利用されます。
収集され保存される統計情報には、シン・テレワークシステムの中継ゲートウェイを経由するセッション数、接続コンピュータ台数、通信データ量、「中継ゲートウェイ」そのもの (統計情報の送付先) の固有の ID、IP アドレスおよびホスト名、OS のバージョン、中継ゲートウェイのバージョン番号が含まれます。
個人情報や認証情報、通信データ、ユーザー名、パスワード、秘密鍵等の秘密情報が収集されることはありません。
また、ヘッダファイル中の WIDE_STAT_POST_URL 設定項目を空文字 ("") に設定することにより、統計情報の送信をオプトアウトすることが可能です。



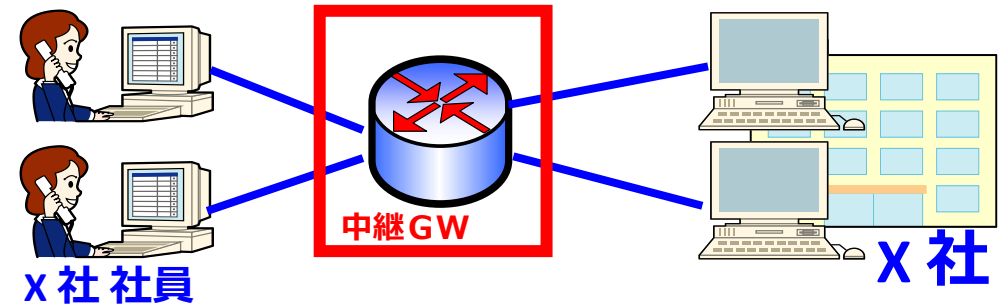
1-1.「シン・テレワークシステム プライベート版」とは

通常版 (2020.4 ~)



大規模中継ゲートウェイ。
IPA および NTT 東日本で構築・運営。
膨大な数のユーザー・組織で
単一のゲートウェイを共有。
IPA によるシステム停止時は全ユーザーが影響。

プライベート版



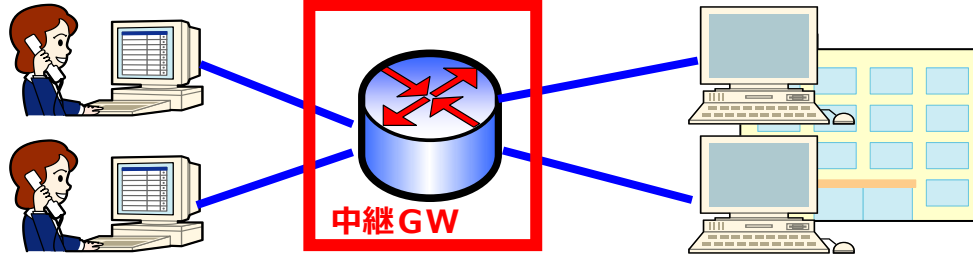
自社専用の中継ゲートウェイ。
単独で動作するソフトウェアとして提供。
IPA のシステムとは無関係で安定・永続的に動作可能。
利用する各社が構築。
(クラウド / DC / 自社内サーバールーム / 自宅サーバー 等、Linux が動作すればどこにでも構築可能)

本ドキュメントでは、利用企業側の視点で、
プライベート版の構築方法を述べる。

1-2. プライベート版には 2 種類が存在

(1) スタンドアロン版

- ・ 中小企業で構築
- ・ 大企業の部署単位で構築



1 台の Linux ホストが
中継 GW になる。

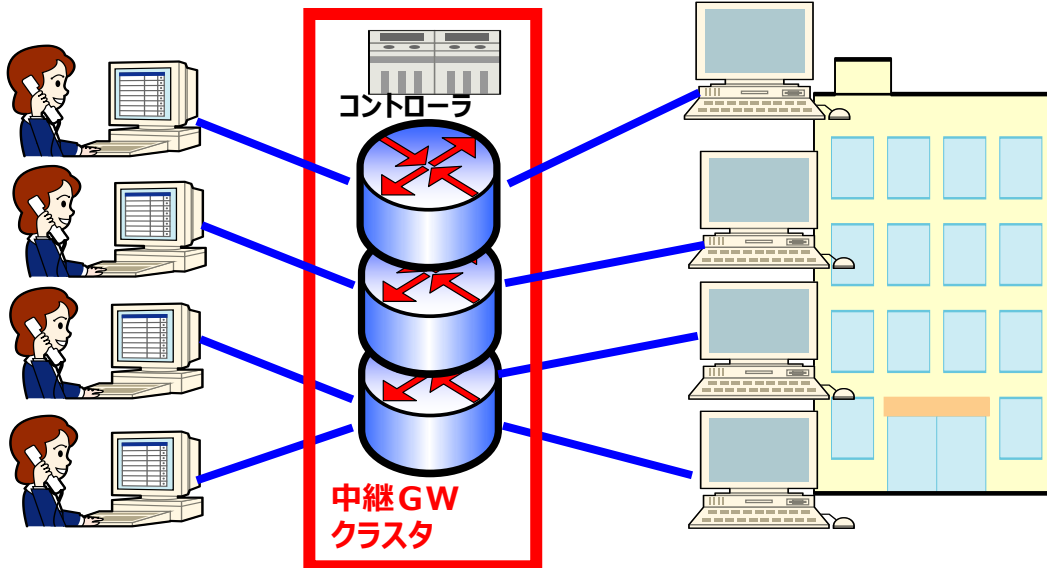
本ドキュメントの対象。

サーバー数に制限。
最大 500 ホストまで
同時起動可能。

(2) ハイパースケール版

※ 開発中

- ・ 大企業全体で構築
- ・ 商用サービスとして
第三者に提供



数台～数百台規模の GW クラ
スタを構築できる。

サーバー数が事実上
無制限。
(1 システムあたり最大
30 万台程度を想定)

500 ～ 1000 ホスト / 1GW
で構成。
例えば 300 台の GW を設置す
ることによって 30 万台程度をサポート。



2. 必要な資源と時間



2. 構築用環境・所要時間

必要なリソースは、わずか 2 台のホストのみ。**所要時間は、1 時間程度。(慣れれば 10 分)**

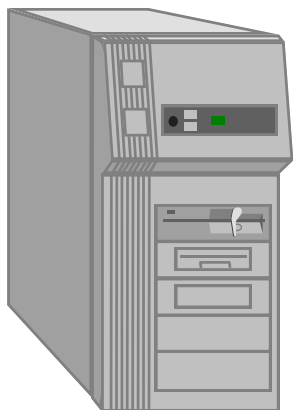
1. 作業用マシン (Windows 10 or 8.1) × 1 台



- メモリ 4GB 以上。
- Administrators 権限。
- 2. のサーバーと SSH 通信できること。

- 本ドキュメントでは Windows 10 を利用。

2. 標準的な Linux が動作するインターネットサーバー × 1 台

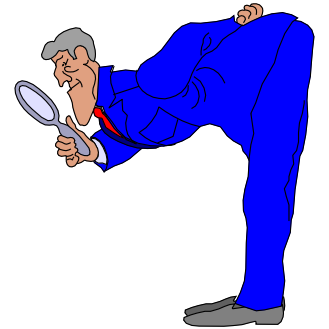
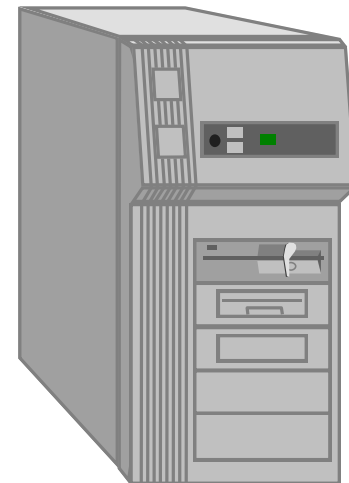


- Intel x64 または ARM 64bit であること。
(Raspberry Pi 等の組み込み機器でも可)
- AWS, Azure, GCP, さくらのクラウド 等の VM も利用可能。
- VMware, Hyper-V, KVM も利用可能。
- Linux コンテナ内で動作可能。(LXD を検証済み)
- メモリ 1GB 以上。CPU 1 コア以上。
- グローバル IPv4 アドレスが 1 個必要。
(静的 IP を推奨するが、別途 Dynamic DNS を利用するならば動的 IP でも可)

- 本ドキュメントでは Ubuntu 18.04 を利用。
- 他の Linux でも動作するが、OS ディストリビューション固有の不具合が発生した場合は、利用側の責任で対応すること。
- Linux コンテナを利用する場合は、多種多様な構成方法があるため、利用側の責任で対応すること。



3. 中継ゲートウェイを構築しよう



3-1. 中継ゲートウェイ (Linux) の構築の流れ

1. Linux サーバーの準備とインターネットへの接続
(本マニュアルでは、AWS を例に説明。)
↓
2. 中継ゲートウェイプログラムのダウンロードとビルド
↓
3. 中継ゲートウェイプログラムの起動



3-2. AWS の Ubuntu Linux サーバーの起動

- 以下、AWS の無料枠を活用して Ubuntu Linux サーバーを起動する。
 - AWS のアカウントは、個人でもクレジットカードがあれば即時に取得可能。
 - 無料枠の詳細は、AWS の Web サイトを参照。
 - CPU 時間は無料であるが、パケット通信料は発生する。短時間の実験的利用で、数十円程度課金されることが考えられるが、自己責任で行なうこと。
 - AWS で無料枠範囲外のホストを立ち上げる等すると課金されるので、十分注意する。
 - 本実験専用に AWS アカウントを作成した場合、不要になったら、アカウントを削除することをお勧めする。

1. AWS コンソール <https://console.aws.amazon.com/> へアクセス。初めての場合は、ユーザー登録。(クレジットカードが必要)
2. 「リージョン」を、「アジアパシフィック (東京) ap-northeast-1」に必ず切替える。(そうしないと、遅延が大きくなる。)



3. 「サービス -> EC2 -> ネットワーク & セキュリティ -> キーペア」より、キーペアを作成する。ファイル形式は pem 形式がよい。

キーペアを作成

キーペア
プライベートキーとパブリックキーの組合せで構成されるキーペアは、セキュリティ認証情報としてインスタンス接続時の ID 証明に使用されます。

名前
testkey
名前には最大 255 文字の ASCII 文字を使用できます。先頭または末尾のスペースを含めることはできません。

ファイル形式
☒ pem
OpenSSH で使用する場合
☐ ppk
PuTTY で使用する場合

タグ (オプション)
リソースにタグが関連付けられていません。
タグを追加
You can add 50 more tags.

キャンセル キーペアを作成

4. pem ファイルが自動的に Web ブラウザでダウンロードされるので、どこか安全な場所 (デスクトップ等) に保存しておく。



5. 「サービス -> EC2 -> インスタンス -> インスタンス」より、「インスタンスを起動」を実行する。
6. 「ステップ 1: Amazon マシンイメージ (AMI)」の検索ボックスで、「Ubuntu」と検索する。
7. 「Ubuntu Server 18.04 LTS (HVM), SSD Volume Type」の「64 ビット (x86)」を選択する。

Q Ubuntu

SSM パラメータによる検索


クイックスタート (8)

マイ AMI (0)

AWS Marketplace (493)

コミュニティ AMI (34907)

☐ 無料利用枠のみ ⓘ

 **Ubuntu Server 20.04 LTS (HVM), SSD Volume Type** - ami-0f2dd5fc989207c82 (64 ビット x86) / ami-0d1f7bec0e294ef80 (64 ビット Arm)


無料利用枠の対象

Ubuntu Server 20.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

ルートデバイスタイプ: ebs 仮想化タイプ: hvm ENA 有効: はい

☒ 64 ビット (x86)
☐ 64 ビット (Arm)

選択

 **Ubuntu Server 18.04 LTS (HVM), SSD Volume Type** - ami-023a7615a07affbe5 (64 ビット x86) / ami-034efdc866e4b4fd7 (64 ビット Arm)

無料利用枠の対象

Ubuntu Server 18.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

ルートデバイスタイプ: ebs 仮想化タイプ: hvm ENA 有効: はい

☒ 64 ビット (x86)
☐ 64 ビット (Arm)

選択

8. 「ステップ 2: インスタンスタイプの選択」で「t2.micro 無料利用枠の対象」を選択する。

	ファミリー	タイプ	vCPU ⓘ	メモリ (GiB)	インスタンスストレージ (GB) ⓘ	EBS 最適化利用 ⓘ	ネットワークパフォーマンス ⓘ	IPv6 サポート ⓘ
<input type="checkbox"/>	t2	t2.nano	1	0.5	EBS のみ	-	低から中	はい
<input checked="" type="checkbox"/>	t2	t2.micro 無料利用枠の対象	1	1	EBS のみ	-	低から中	はい



9. 「ステップ 3: インスタンスの詳細の設定」はデフォルトのままでよい。
10. 「ステップ 4: ストレージの追加」はデフォルトのままでよい。
11. 「ステップ 5: タグの追加」はデフォルトのままでよい。(タグを付けてもよい。)
12. **これが肝心。**「ステップ 6: セキュリティグループの設定」で、新しいセキュリティグループを作成する。ルールは、以下のようにする。
- **SSH (デフォルト)** ※ 任意の場所から接続できるようにする
 - **HTTPS (追加すること)** ※ 任意の場所から接続できるようにする
 - **すべての ICMP - IPv4 (追加すること)** ※ 任意の場所から接続できるようにする

セキュリティグループの割り当て: ☒ 新しいセキュリティグループを作成する

☐ 既存のセキュリティグループを選択する

セキュリティグループ名:

説明:

タイプ ⓘ	プロトコル ⓘ	ポート範囲 ⓘ	ソース ⓘ	説明 ⓘ	
SSH ▼	TCP	22	任意の場所 ▼ 0.0.0.0/0, ::/0	例: SSH for Admin Desktop	×
HTTPS ▼	TCP	443	任意の場所 ▼ 0.0.0.0/0, ::/0	例: SSH for Admin Desktop	×
すべての ICMP ▼	ICMP	0 - 65535	任意の場所 ▼ 0.0.0.0/0, ::/0	例: SSH for Admin Desktop	×
ルールの追加					



13. 「ステップ 7: インスタンス作成の確認」で、いよいよ、「起動」をクリックする。先ほど作成したキーペアを選択して起動する。

ステップ 7: インスタンス作成の確認
 インスタンスの作成に関する詳細を確認してください。各セクションの変更に戻ることができます。[作成] をクリックして、インスタンスにキーペアを割り当て、作成処理を完了します。

⚠ インスタンスのセキュリティを強化してください。セキュリティグループ launch-wizard-2 は世界に向けて開かれています。
 このインスタンスには、どの IP アドレスからもアクセスできる可能性があります。セキュリティグループのルールを更新して、既知の IP アドレスからのみアクセスできるようにすることをお勧めします。
 また、セキュリティグループの追加ポートを開いて、実行中のアプリケーションやサービスへのアクセスを容易にすることもできます。たとえば、ウェブサーバー用に HTTP (80) を開きます。 [セキュリティグループの編集](#)

AMI の詳細

AMI の編集

▼ インスタンスタイプ

インスタンスタイプの編集

インスタンスタイプ	ECU	vCPU	メモリ (GiB)	インスタンスストレージ (GB)	EBS 最適化利用	ネットワークパフォーマンス
t2.micro	-	1	1	EBS のみ	-	Low to Moderate

▼ セキュリティグループ

セキュリティグループの編集

▼ インスタンスの詳細

インスタンスの詳細の編集

▼ ストレージ

ストレージの編集

▼ タグ

タグの編集

キャンセル

戻る

起動

既存のキーペアを選択するか、新しいキーペアを作成します。

キーペアは、AWS が保存するパブリックキーとユーザーが保存するプライベートキーファイルで構成されます。組み合わせて使用することで、インスタンスに安全に接続できます。Windows AMI の場合、プライベートキーファイルは、インスタンスへのログインに使用されるパスワードを取得するために必要です。Linux AMI の場合、プライベートキーファイルを使用してインスタンスに SSH で安全に接続できます。

注: 選択したキーペアは、このインスタンスに対して権限がある一連のキーに追加されます。「パブリック AMI から既存のキーペアを削除する」の詳細情報をご覧ください。

既存のキーペアの選択

キーペアの選択

dnctest1

☒ 選択したプライベートキーファイル (dnctest1.pem) へのアクセス権があり、このファイルなしではインスタンスにログインできないことを認識しています。

キャンセル

インスタンスの作成

14. AWS コンソールの「EC2」の「インスタント」で、13 で起動したインスタンスが「実行中」になるまで待つ。(時々画面更新する) 数分間で完了するはずである。

<input type="checkbox"/>	Name ▼	インスタンス ID	インスタンス... ▼	インスタンス... ▼	ステータスチ...	Alarm status	ア
<input type="checkbox"/>	-	i-00c05e4c098693796	<div> <div>✓ 実行中</div> <div>🔍</div> </div>	t2.micro	<div> <div>✓ 2/2 のチェ...</div> </div>	アラーム... +	ap



15. 「サービス -> EC2 -> ネットワーク & セキュリティ -> Elastic IP」で、静的パブリック IPv4 アドレスを 1 つ割当てを受ける。
IP アドレスの割当ては、即時に完了する。

EC2 > Elastic IP アドレス > Elastic IP アドレスの割り当て

Elastic IP アドレスの割り当て

パブリック IP アドレスを割り当てるパブリック IPv4 アドレスプールを選択して、Elastic IP アドレスを割り当てます。実行中のインスタンスに 1 つの Elastic IP (EIP) アドレスを無料で関連付けることができます。追加の EIP をそのインスタンスに関連付ける場合、そのインスタンスに関連付けられた追加の EIP ごとに按分計算ベースで課金されます。追加の EIP は Amazon VPC でのみ使用できます。Elastic IP アドレスを効率的に使用できるように、これらの IP アドレスが実行中のインスタンスに関連付けられていない場合、または停止したインスタンスやアタッチされていないネットワークインターフェイスに関連付けられている場合、時間単位の小額の料金が課金されます。 [詳細はこちら](#)

Elastic IP アドレスの設定

パブリック IPv4 アドレスプール

パブリック IP アドレスプールは、パブリック IP アドレスのプール。顧客が所有してアカウントに持ち込むプール、または顧客が所有して引き継ぎアドレスするプールから割り当てられます。...

- ☒ Amazon の IPv4 アドレスプール
- ☐ AWS アカウントに持ち込むパブリック IPv4 アドレス(プールが見つからなかったため)
- ☐ 顧客所有の IPv4 アドレスのプール(顧客所有のプールが見つからないため、オプションは無効です) [詳細はこちら](#)

Global static IP addresses

AWS Global Accelerator can provide global static IP addresses that are announced worldwide using anycast from AWS edge locations. This can help improve the availability and latency for your user traffic by using the Amazon global network. [詳細はこちら](#)

Create accelerator [🔗](#)

キャンセル **割り当て**

16. 15 で割当てられた静的パブリック IPv4 アドレスに対して、「アクション」から「Elastic IP アドレスの関連付け」をクリックし、13 で作成したEC2 インスタンスに関連付ける。数秒後には工事が完了し、この IPv4 アドレス宛の通信が EC2 インスタンスにルーティングされるようになる。

インスタンス

Q i-00c05e4c098693796 X [🔗](#)

プライベート IP アドレス

Elastic IP アドレスを関連付けるプライベート IP アドレスです。

Q プライベート IP アドレスを選択します

再関連付け

Elastic IP アドレスがすでにリソースに関連付けられている場合に、そのアドレスを別のリソースに再度関連付けられるかどうかを指定します。

☐ この Elastic IP アドレスの再関連付けを許可する

キャンセル **関連付ける**



17. 16 で割当てられた Elastic IP アドレス宛に ping を打ってみて、応答があるかどうか確認する。

```
C:\WINDOWS\system32\cmd.exe
C:\>ping 54.250.11.23

54.250.11.23 に ping を送信しています 32 バイトのデータ:
54.250.11.23 からの応答: バイト数 = 32 時間 = 6ms TTL=45
54.250.11.23 からの応答: バイト数 = 32 時間 = 6ms TTL=45
54.250.11.23 からの応答: バイト数 = 32 時間 = 6ms TTL=45
54.250.11.23 からの応答: バイト数 = 32 時間 = 6ms TTL=45

54.250.11.23 の ping 統計:
    パケット数: 送信 = 4, 受信 = 4, 損失 = 0 (0% の損失)、
    ラウンドトリップの概算時間 (ミリ秒):
        最小 = 6ms、最大 = 6ms、平均 = 6ms

C:\>
```

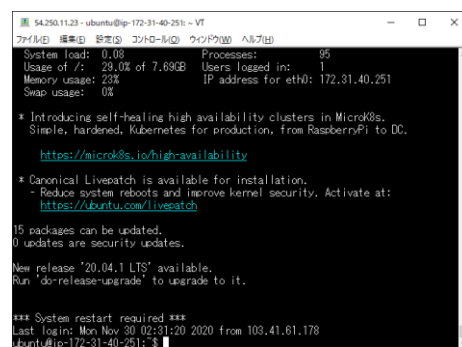
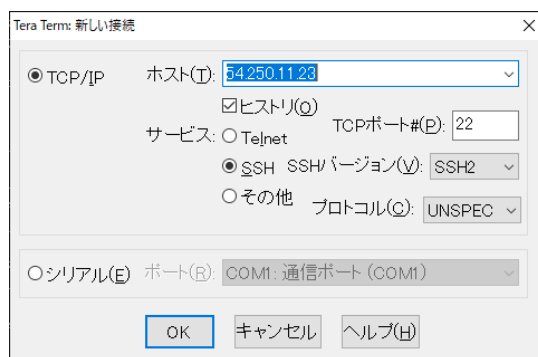
※ IP アドレスは、サンプルである。
実際の構築中の IP アドレスに置換すること。

18. 任意の好きな SSH クライアント (例: Tera Term) で、この IP アドレス宛に SSH 接続を行なう。

SSH のユーザー ID: ubuntu

SSH の認証方式は、秘密鍵とし、先ほど作成したキーペアの pem ファイルを指定する。

ubuntu というユーザーでログインできるはずである。(初回サーバー起動後は、ログインできるまで数分かかることがある。)



3-3. 中継ゲートウェイプログラムのビルドの実行

以下のコマンドを順に実行する。

あるコマンドを実行し、エラーが出ていないかどうか確認してから、次を実行すること。

```
sudo apt-get -y update && sudo apt-get -y install build-essential libreadline-dev
```

うまくいけば、gcc 等のコンパイラがインストールされる。

```
git clone https://thintele:MikakaNet2020@github.com/IPA-CyberLab/IPA-DN-ThinApps-Private.git --recursive -b beta7preview11
```

改行は削除して、1 行に続けて入力すること。うまくいけば、中継ゲートウェイのソースコードがダウンロードされる。

```
cd ~/IPA-DN-ThinApps-Private/src/ ; make -j 8
```

ダウンロードした中継ゲートウェイのソースコードをビルドする。

```
sudo ~/IPA-DN-ThinApps-Private/src/bin/thingate
```

ビルドが完了し、バイナリプログラム「thingate」が正しく動作することを確認する。以下のように表示される。

ThinGate service program

Copyright (c) NTT East Special Affairs Bureau. Copyright (c) IPA CyberLab of Industrial Cyber Security Center.

All Rights Reserved.

thingate command usage:

thingate start - Start the ThinGate service.

thingate stop - Stop the ThinGate service if the service has been already started.



3-4. 中継ゲートウェイプログラムの起動と終了

ビルドした中継ゲートウェイプログラム (UNIX デーモン) を、以下のようにして起動する。

```
sudo ~/IPA-DN-ThinApps-Private/src/bin/thingate start
```

起動に成功すると、

The ThinGate service has been started.

と表示される。

中継ゲートウェイプログラムは、UNIX デーモンとして常時稼働する。

一度デーモンとして起動すると、通常、終了する必要はないが、何らかの理由で終了する場合は以下のようにする。

```
sudo ~/IPA-DN-ThinApps-Private/src/bin/thingate stop
```

終了に成功すると、

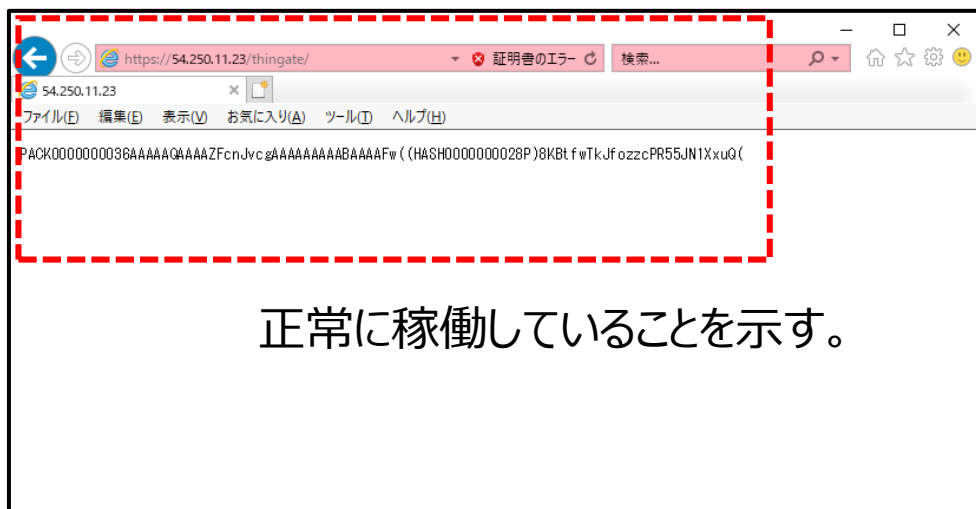
ThinGate service has been stopped.

と表示される。



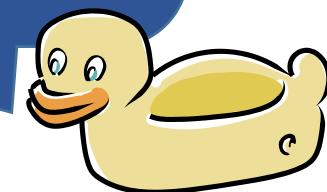
3-5. 中継ゲートウェイシステム起動後の稼働チェック

- 中継ゲートウェイシステムが起動した後、正しく稼働しているかどうかは、以下の方法でチェックできる。
 - インターネット上の任意の端末 (作業用マシンでもよい) で Web ブラウザを起動する。
 - Web ブラウザで、<https://a.b.c.d/thingate/> という URL にアクセスする。
(a.b.c.d の部分は、中継ゲートウェイシステムの IP アドレスに置換する。)
 - SSL 証明書エラーが発生するが、無視してページを表示する。
 - “PACK” という文字で始まる 1 行の複雑な文字列が表示された場合は、中継ゲートウェイシステムは正しく稼働している。

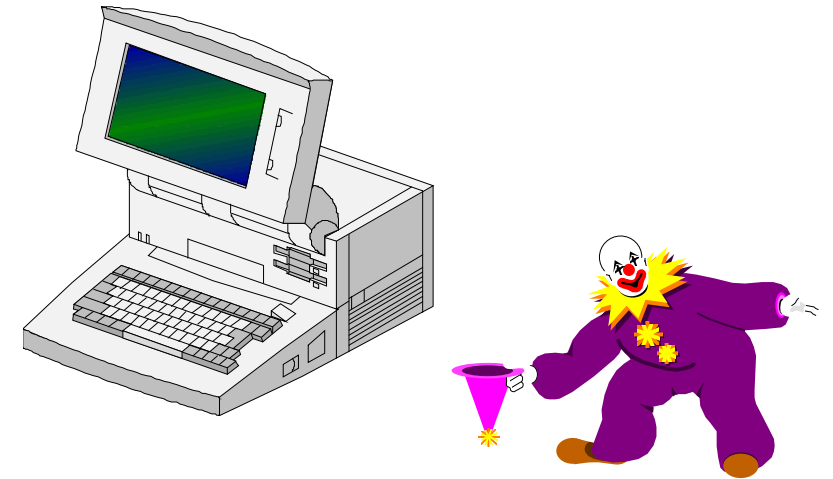


うまくブラウザで接続できない場合は、以下の点を確認しよう。

- クラウド (AWS) 等の VM 基盤でネットワーク側にファイアウォールがある場合は、TCP 443 (https) を開放しているか。
- OS のファイアウォールを同様に開放しているか。



4. アプリケーションをビルドしよう



4-1. アプリケーション・インストーラ構築の流れ

1. 作業用マシンへの Visual Studio 2019 のインストール
↓
2. ソースコード一式のダウンロード
↓
3. ソースコードのビルド



4-2. 作業用マシンへの Visual Studio 2019 のインストール

- 作業用マシンに、Microsoft 社の Visual Studio 2019 をダウンロードしてインストールする。
<https://visualstudio.microsoft.com/ja/downloads/>



- 個人または小規模企業での利用
- 大企業であっても、オープンソースへの貢献を目的としてシン・テレワークシステムをビルドし検証する場合

は Community エディションをダウンロードすればよい。

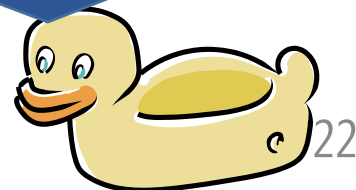
上記に該当せず、商用目的のみで利用する場合は、Professional エディションの体験版をダウンロードし、体験期間終了後に購入すること。

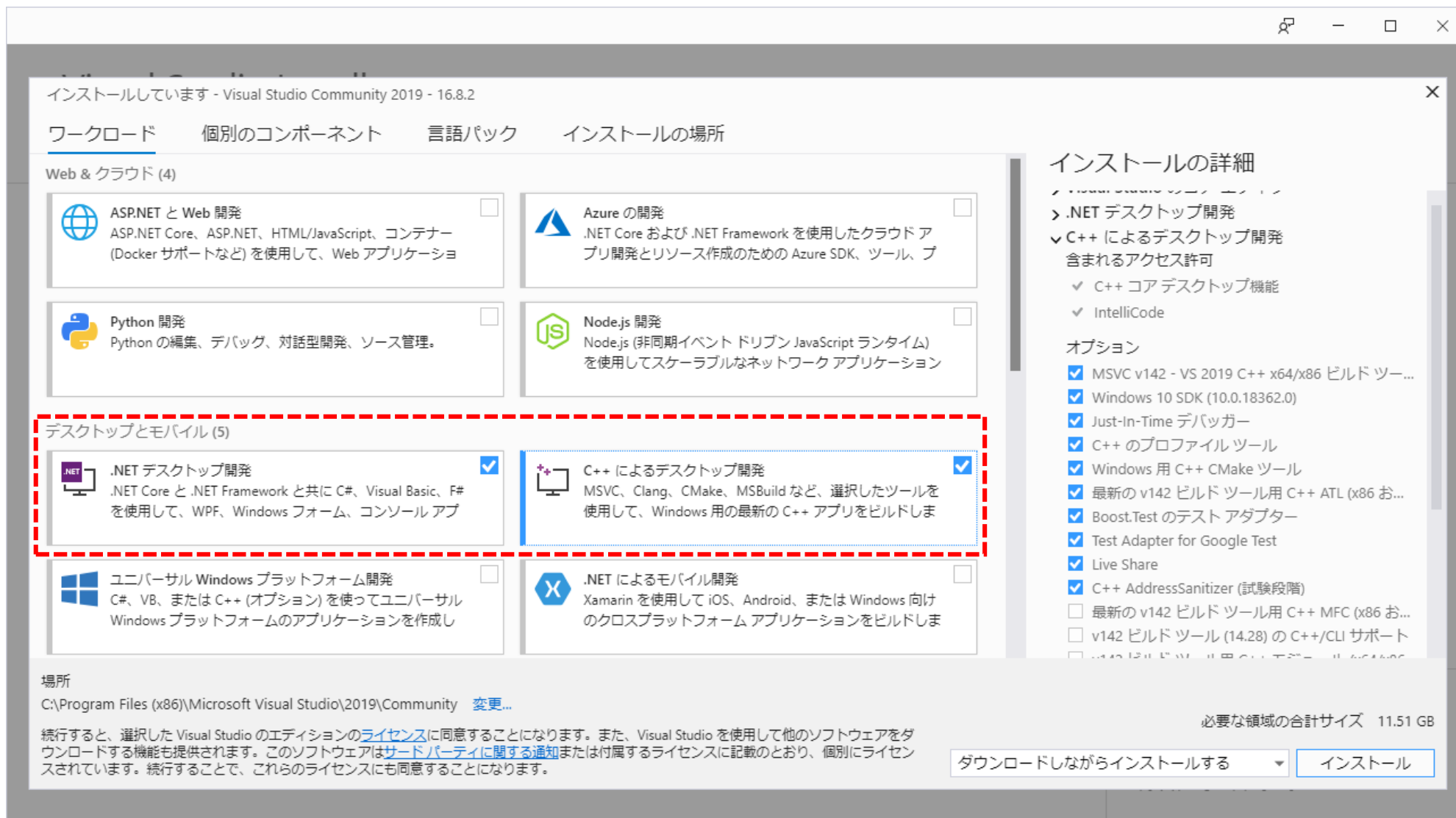
VS2019 には、複数のエディションがある。
Community はほぼフル機能が利用できる。

- 小規模企業 (年売上 100 万ドル以下) での利用
または
- オープンソースソフトウェア開発への貢献
の目的であれば、Community が利用可能である。

シン・テレワークシステムのソースコードはオープンソース
であるので、評価・テストする目的であれば、
Community エディションで良いと考えられる。

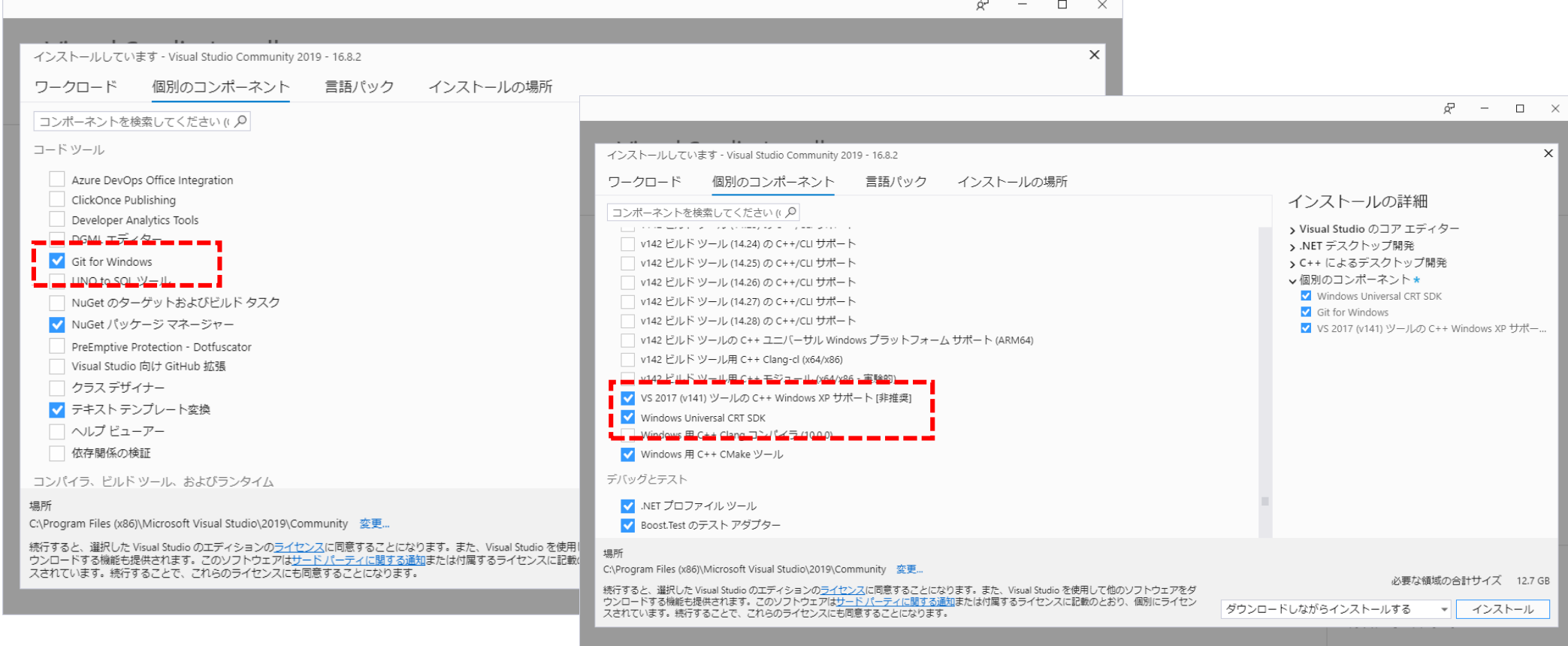
一方、年売上 100 万ドル以上の企業で、かつ、オープンソースへの貢献を意図していない利用の場合 (単に社内システムを構築する等) は Professional エディションが必要であると考えられる。





- Visual Studio 2019 のインストーラでは、最初に、「.NET デスクトップ開発」と「C++ によるデスクトップ開発」の 2 つをチェックする。
- 次に、「個別のコンポーネント」画面に移動する。



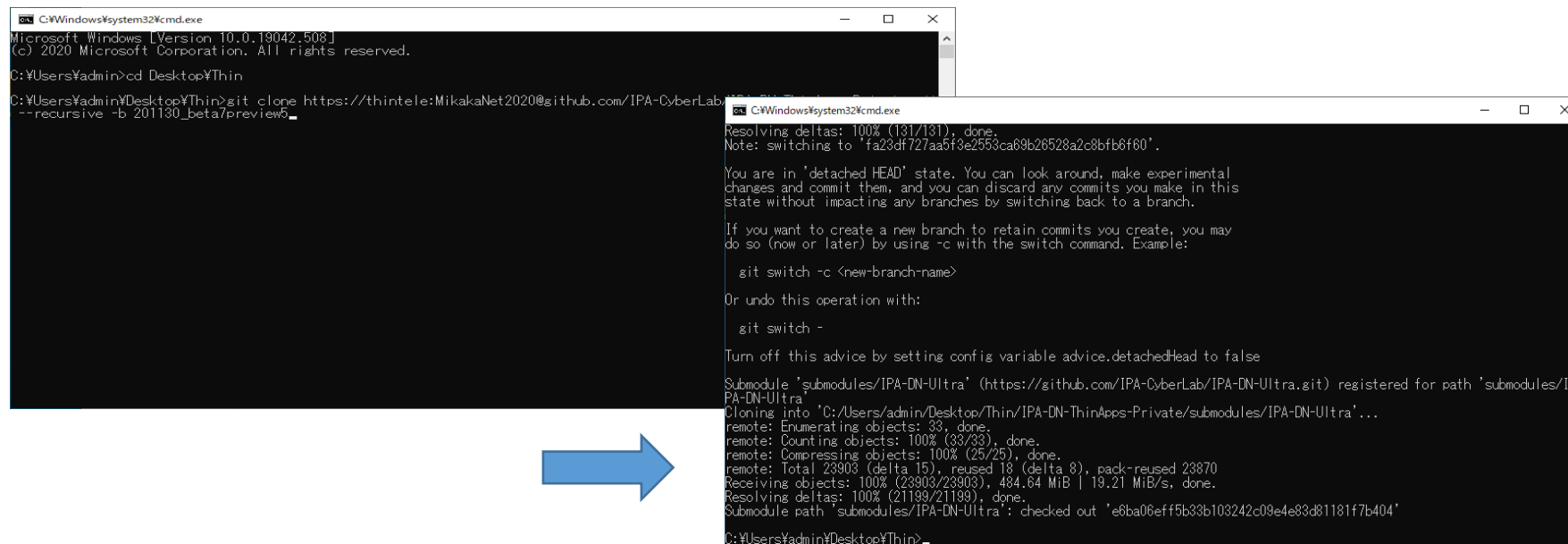
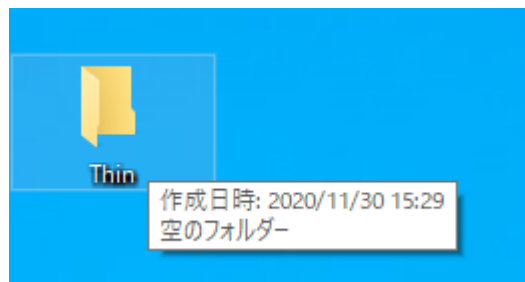


- 「個別のコンポーネント」として、上図のとおり、
 - Git for Windows
 - VS 2017 (v141) ツールの C++ Windows XP サポート (非推奨)
 - Windows Universal CRT SDK
 の3つを忘れずに手動でチェックし、インストールを開始する。
- Visual Studio 2019 のインストールが完了したら、VS2019 が自動で起動することがあるが、一度終了する。



4-3. シン・テレワークシステムのソースコードのダウンロード

1. デスクトップ等、適当な場所に、構築作業用の新しいフォルダを作成する。たとえば、デスクトップに “Thin” というフォルダを作成する。



2. コマンドプロンプト (Windows + R → “cmd” を実行) を起動し、1 で作成したフォルダに CD (カレントディレクトリの移動) する。

例: `cd C:\Users\<ユーザー名>\Desktop\Thin`

3. Git を用いてソースコード一式をダウンロードする。(下記の入力例の改行はなくし、1 行で入力すること。)

`git clone https://thintele:MikakaNet2020@github.com/IPA-CyberLab/IPA-DN-ThinApps-Private.git --recursive -b beta7preview11`



4-4. 中継ゲートウェイの IP アドレス等の指定

src¥bin ディレクトリを開く。

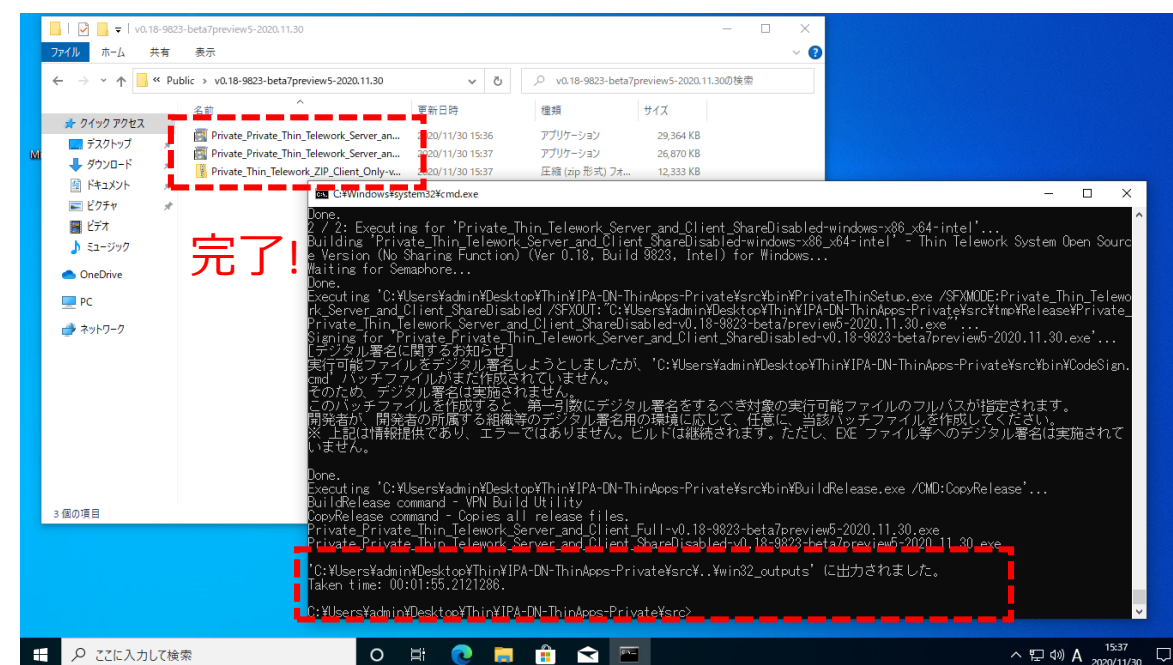
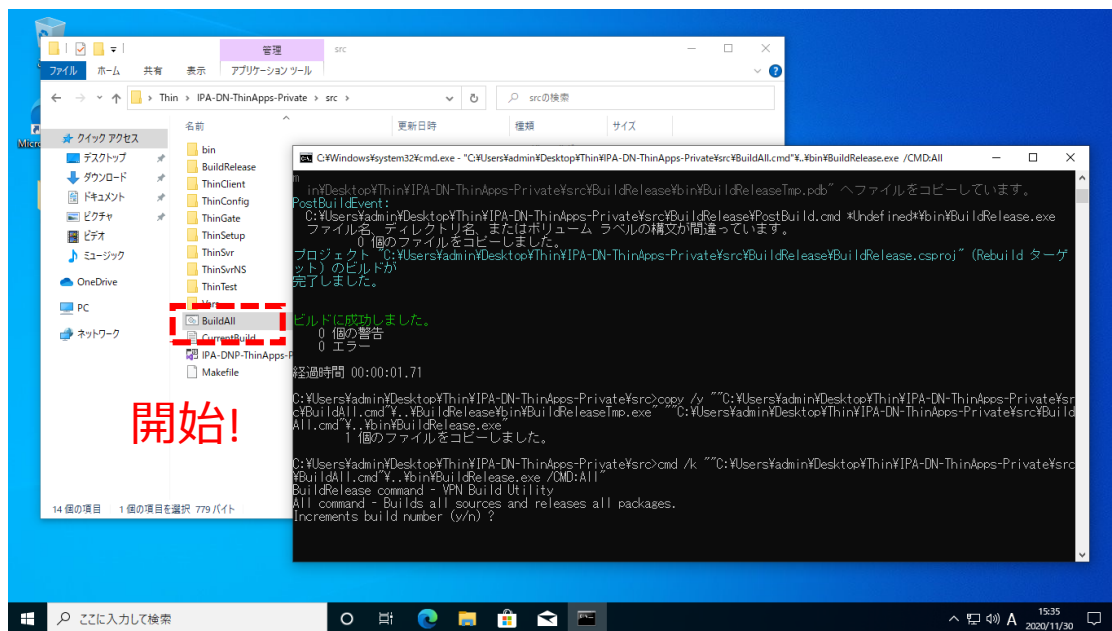
EntryPoint.dat をメモ帳などのテキストエディタで開く。

URL およびシステム名を書換える。

1. 上図のように、ソースコード集の“src¥bin¥EntryPoint.dat”をメモ帳等のテキストエディタで開くと、下の方に `https://` で始まる URL がある。これを、「3-5. 中継ゲートウェイシステム起動後の稼働チェック」の URL と同じ文字列に置き換える。
2. “SYSTEM:” で始まる行がある。ここに、サンプルとして “MiKaKa Corporation Private Gateway 001” という文字列が書かれている。これを、自組織のシステムを意味する適当な文字列 (英数字のみ) に置換する。(置換しなくても動作するが、アプリケーションの起動画面に表示されるので、できれば置換したほうがよい。)
※ “SYSTEM:” の文字は削除しないこと。
3. EntryPoint.dat ファイルを保存して、テキストエディタを閉じる。



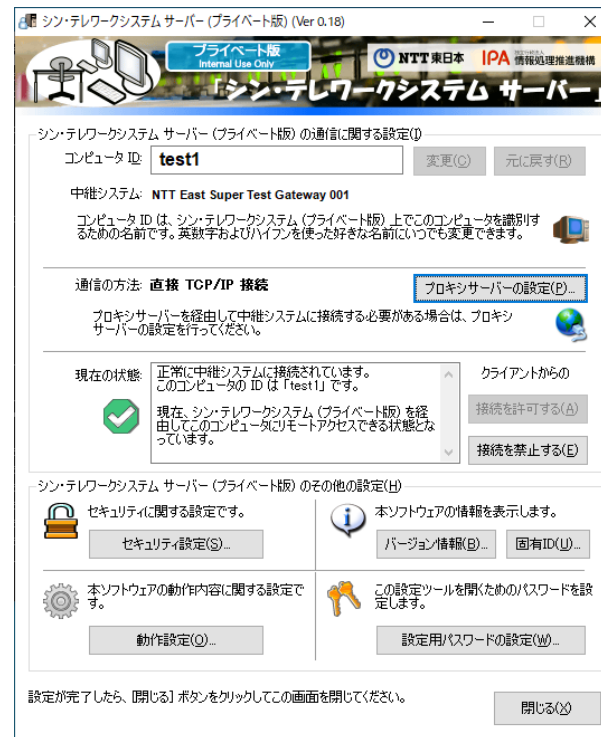
4-5. ビルドの実施



- ビルドは、src¥BuildAll.cmd (バッチファイル) をダブルクリックすることで開始される。
- ビルドが開始されると、“Increments build number (y/n) ?” と表示される。
ここでは、“y” と入力し、Enter キーを押す。
- ビルドが完了すると、win32_outputs¥Public¥<識別子>¥ にインストーラー式が生成される。



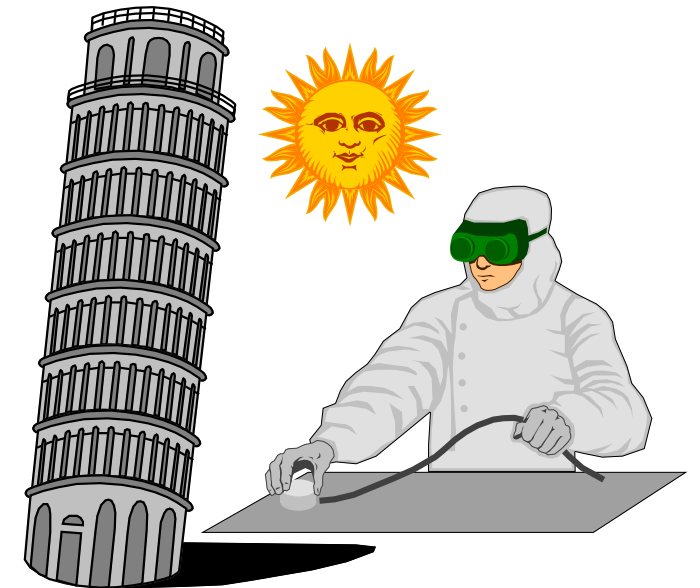
4-6. 生成されたインストーラの動作の確認



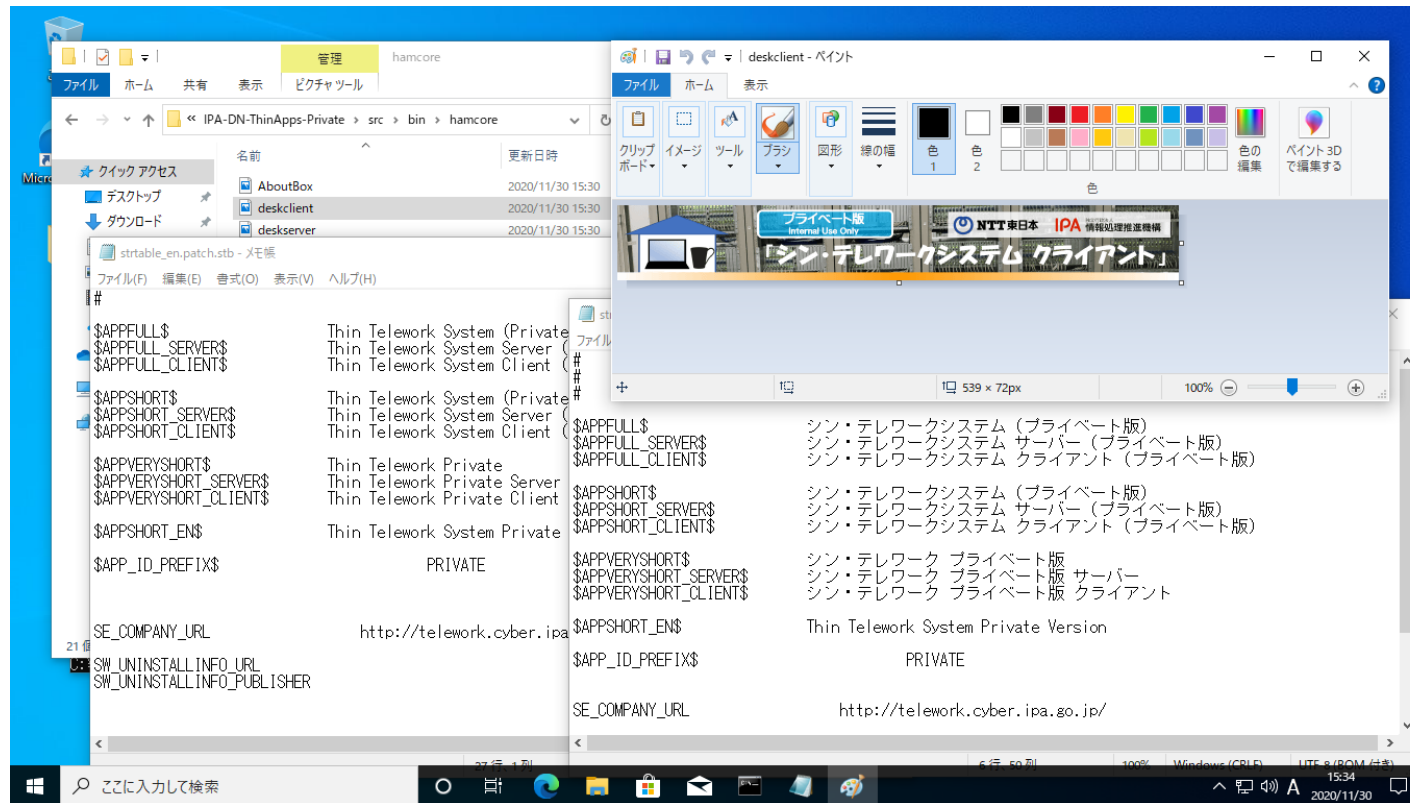
- ビルドされたインストーラを、任意の端末にインストールし、構築し稼働している中継ゲートウェイを経由してテレワーク通信が可能となっていることを確認する。



5. 高度な構築・運用手法



5-1. アプリケーションの UI の高度なカスタマイズ方法



- src¥bin¥hamcore¥strtable_ja.patch.stb および strtable_en.patch.stb ファイルをテキストエディタで編集することにより、ソフトウェアのブランド名などを変更することができる。
- また、src¥bin¥hamcore¥ ディレクトリの BMP ファイルを編集することにより、ソフトウェア中の画像を変更することができる。



5-2. アプリケーションの ID (AppId) の変更によるユニーク化

- 今後、本格的に、プライベートバージョンを構築し展開する場合は、アプリケーションの ID は、ソースコードから、他 (※) と重複のないように変更されるべきである。
そうしないと、複数版を共存インストールすることができない。
※ ソースコードをそのまま用いて他の組織がビルドしたバージョン等。
- ソースコード中には、デフォルトのアプリケーション ID として “Private” が設定されている。以下の 5 つのファイル中の文字列 “Private” を 8 文字以内の英数字の他のユニークな文字列 (例: Mikaka) に置換すること。
 - src¥Vars¥Vars.cs
 - src¥Vars¥Vars.h
 - src¥Vars¥Vars.props
 - src¥bin¥hamcore¥strtable_ja.patch.stb
 - src¥bin¥hamcore¥strtable_en.patch.stb
- src¥Vars¥Vars.h ファイルには
「#define DS_RPC_PORT 9825」
という定数がある。これは、シン・テレワークシステム サーバーのバックグラウンドプロセスと、サーバー設定ツールとの間の設定用通信に利用される localhost 内 TCP ポートである。9826 以降の、他の組織とかぶりにくい、適当な整数に変更することを推奨する。※「#define DS_URDP_PORT 3459」は、変更しないこと。



5-3. ゲートウェイに登録済みのサーバー一覧の管理

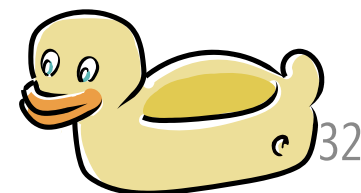
- 中継ゲートウェイに登録されているサーバーの一覧は、以下のデータベースファイルに保存されている。
- データベースはテキスト形式であり、cat コマンドで、内容を表示することができる。

```
sudo cat ~/IPA-DN-ThinApps-Private/src/bin/MachineDatabase.config
```

```
54.250.11.23 - ubuntu@ip-172-31-40-251: ~/IPA-DN-ThinApps-Private VT
ファイル(F) 編集(E) 設定(S) コントロール(C) ウィンドウ(W) ヘルプ(H)
# -----
# You may edit this file when the server program is not running.
# In prior to edit this file manually by your text editor,
# shutdown the server background service.
# Otherwise, all changes will be lost.
#
declare root
[
  uint Revision 5

  declare Machines
  [
    declare MSID-DESK-EA4063CBEA5FF26DE4BCF9B35551A0CFD5B3913F
    [
      string CertHash EA4063CBEA5FF26DE4BCF9B35551A0CFD5B3913F
      uint64 CreateDate 1606686022950
      string CreateHost 103x41x
      string CreateIp 103.41
      uint64 FirstClientDate 0
      string HostSecret2 1D72D5148FBF2CF64347401ADBBC151E0CAA703B
      uint64 LastClientDate 0
      string LastIp 103.41
      uint64 LastServerDate 1606699739152
      string Msid MSID-DESK-EA4063CBEA5FF26DE4BCF9B35551A0CFD5B3913F
      uint NumClient 0
      uint NumServer 11
      string Pcid test1
      uint64 ServerMask64 146
      uint64 UpdateDate 1606686022950
      string WolMacList $
    ]
  ]
]
```

- thingate デモンプロセスを停止中は、データベース (config ファイルの内容) をテキストエディタで書換えることができる。この方法により、コンピュータ ID を強制的に変更したり、登録を削除したりすることも可能である。
- Date 系の変数は、1970/01/01 09:00:00 (GMT) を 0 としてミリ秒単位で増加する 64 bit 整数である。(JST ではないので注意)
- その他の変数の意味は、
<https://github.com/IPA-CyberLab/IPA-DN-Ultra/blob/7de9d485817536826a2dab4f7edf9c98bd41d2b7/src/Cedar/WtGate.h#L179>
の WG_MACHINE のコメントを参照すること。



5-4. ゲートウェイに現在接続しているサーバーセッション一覧の表示

- 中継ゲートウェイに現在接続しているサーバーセッション一覧は、中継ゲートウェイ上から以下のコマンドを実行することで表示することができる。

```
curl -k https://127.0.0.1/thinstat/
```

実行例

```
54.250.11.23 - ubuntu@ip-172-31-40-251: ~/IPA-DN-ThinApps-Privat VT
ファイル(F) 編集(E) 設定(S) コントロール(C) ウィンドウ(W) ヘルプ(H)
ubuntu@ip-172-31-40-251:~/IPA-DN-ThinApps-Private/src/bin$ curl -k https://127.0.0.1/thinstat/
declare root
[
  {
    uint NumSession 1
    declare 2DEE0B25E999503D520DD8572F3FACE17CF627F7
    [
      uint64 EstablishedDateTime 1606699739189
      string Hostname 103x41x...
      string IpAddress 103.41...
      string Msid MSID-DESK-EA4063CBEA5FF26DE4BCF9B35551A0CFD5B3913F
      uint NumClients 0
      uint64 ServerMask64 146
      string SessionId 2DEE0B25E999503D520DD8572F3FACE17CF627F7
    ]
  }
]
ubuntu@ip-172-31-40-251:~/IPA-DN-ThinApps-Private/src/bin$
```

意味

- EstablishedDateTime: 接続確立日時。
(数値の意味は 5-3 を参照のこと)
 - Hostname: 接続元ホスト FQDN
 - IpAddress: 接続元ホスト IP アドレス
 - Msid: 固有 ID (MSID-DESK-固有ID)
 - NumClients: 現在接続中のクライアント数
 - ServerMask64: サーバー情報を示すビットマスク
 - SessionId: セッション固有の乱数 ID
- ServerMask64 のビットフラグの一覧:

<https://github.com/IPA-CyberLab/IPA-DN-Ultra/blob/7de9d485817536826a2dab4f7edf9c98bd41d2b7/src/Cedar/Desk.h#L167>



5-5. ゲートウェイの Linux カーネルの推奨設定

```
# Kernel
kernel.panic=15
kernel.panic_on_oops=1
vm.overcommit_memory=2
vm.overcommit_ratio=90
vm.panic_on_oom=0
vm.oom_kill_allocating_task=1
kernel.sysrq=0
kernel.core_uses_pid=1

# Network
net.core.rmem_default=16777216
net.core.rmem_max=16777216
net.core.wmem_default=16777216
net.core.wmem_max=16777216

# for busy servers
net.core.somaxconn=1024
net.ipv4.tcp_rfc1337=1
net.ipv4.tcp_tw_reuse=1
net.ipv4.tcp_max_syn_backlog=1024
net.ipv4.tcp_fin_timeout=4
net.ipv4.tcp_rmem=4096 87380 5955584
net.ipv4.tcp_wmem=4096 65536 5955584
net.ipv4.tcp_timestamps=0

# Semaphones
kernel.sem=65536 655360 64 655360
kernel.msgmni=65536

# Files
fs.inotify.max_queued_events=1048576
fs.inotify.max_user_instances=1048576
fs.inotify.max_user_watches=1048576
fs.file-max=1048575
fs.aio-max-nr=1048575
kernel.msgmax=1048575
kernel.msgmnb=1048575
kernel.threads-max=1048575
net.core.netdev_max_backlog=182757
vm.max_map_count=262144
```

- /etc/sysctl.conf を編集することにより、Linux カーネルのパラメータをチューニングすることができる。
- パラメータのチューニングには、色々な党派があるが、IPA で現在運用している「シン・テレワークシステム」のゲートウェイは、左のようなパラメータを指定している。
- 大量の FD (ファイル・ディスクリプタ) やソケットを使用した処理をエラーなく行なうために、左記を参考にして、十分なスケーラビリティ設定を行なうことがのぞましい。
- また、カーネルの limits 変数について、
nofile, nproc, sigpending はいずれも大きな値 (例: 1048575)、
memlock は unlimited と指定することが望ましい。
- これらのチューニングを行なわない場合、パフォーマンスが劣化したり、大量の接続が同時にあった場合に予期せぬ動作が発生したりすることがある。



5-6. 中継ゲートウェイのより詳細な設定

- 中継ゲートウェイの bin ディレクトリにある “ThinGate.ini” ファイルを編集することで、中継ゲートウェイの動作を変更可能である。(いずれもデーモン再起動が必要)

```
// 動作 TCP ポート  
ListenPort 443
```



中継ゲートウェイの待受け TCP ポート (Listen するポート) を変更することができる。

```
// ゲートウェイ証明書  
ServerCert @ThinGate.cer  
ServerKey @ThinGate.key
```

```
// ゲートウェイ動作モード  
StandaloneMode 1
```

```
// DNS 逆引きをしない  
NoLookupDnsHostname 0
```



1 にすると、DNS 逆引きを無効化できる。接続を受付ける際のパフォーマンスが向上する。

```
// DoS 攻撃防止を無効  
DisableDoSProtection 0
```



1 にすると、組み込みの DoS 攻撃対策機能 (同じ IP アドレスから大量の接続要求がきた場合に無視する) を OFF にすることができる。

```
// プライベート IP LAN 内での登録  
AllowPrivateIp 1
```

```
// 通信タイムアウト設定 (msecs)  
TunnelTimeout 30000  
TunnelKeepAlive 10000  
TunnelUseAggressiveTimeout 1
```

```
// OTP (ワンタイムパスワード) 送付用 SMTP サーバーの指定  
// SMTP における認証や SSL には対応しておりません。¥  
// SMTP サーバー側の設定を適用に実施し、  
// このホストからの SMTP 接続に対して、ユーザー認証・SSL なしで送付できる設定としてください。  
SmtpServerHostname 192.168.3.50  
SmtpServerPort 25  
SmtpOtpFrom telework@example.org
```



OTP (ワンタイムパスワード) 機能を利用する場合の、OTP をメール送信するために必要な SMTP サーバーの設定である。OTP を利用する場合は、必ず設定をすること。



5-7. 独自のプライベート CA (認証局) の証明書の利用

- シン・テレワークシステムでは、クライアントとサーバーは、接続先の中継ゲートウェイが真正であるかどうかを、X.509 デジタル証明書 (RSA 形式) により検証する。
- シン・テレワークシステムのソースコードには、デフォルトで、サンプル証明書が組み込まれている。
- デフォルトのサンプル証明書でも動作するが、信頼できない回線でパケットが書換えられ、中継ゲートウェイの偽物が立てられる可能性を排除するため、独自の CA および CA で発行される証明書による検証を行なうことが推奨される。

ゲートウェイ側

- src¥bin¥ThinGate.cer
に、ゲートウェイ用の X.509 証明書 (つまり、公開鍵) を置くこと。
- src¥bin¥ThinGate.key
に、ゲートウェイ用の X.509 証明書に対応する秘密鍵を置くこと。

アプリケーション側

- src¥bin¥EntryPoint.dat の先頭部分の
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----
で囲まれている部分を、CA の X.509 ルート証明書 (つまり、公開鍵) に置換すること。

使用できる証明書の形式等

- X.509 v3 証明書であること。
- RSA 2048 ～ 4096 bit が利用されており、ダイジェスト形式が SHA-2 形式であること。
- 証明書の用途については、制限はない。
- ゲートウェイ用の証明書の有効期限がチェックされる。CA の証明書の有効期限は検証しない。
- CRL 機能は利用できない。



5-8. IP アドレスの代わりに DNS FQDN を指定する方法 等

- アプリケーション側では、「4-4. 中継ゲートウェイの IP アドレス等の指定」で EntryPoint.dat に中継ゲートウェイの IP アドレスを指定したが、DNS ホスト名 (FQDN) が利用可能であれば、DNS 名を指定することも可能である。

例: <https://abc.example.org/thingate/>

- これにより、中継ゲートウェイが可変 IP アドレスであっても、Dynamic DNS による運用が可能となる。
- 長期的に利用される可能性があるサービスを構築するときは、IP アドレス直打ちではなく、DNS 名による運用を推奨する。

- 中継ゲートウェイを 443 (https) 以外のポートで動作させる設定 (5-6) などを行っている場合、アプリケーション側では、以下のような URL でポート番号を指定することが可能である。

例: <https://abc.example.org:1234/thingate/>

ただし、サーバーまたはクライアントの利用環境で HTTP, HTTPS しか疎通しないようなファイアウォールやプロキシサーバーが想定されるので、中継ゲートウェイは、できる限り HTTPS (443) ポートで運用することを推奨する。



5-9. 中継ゲートウェイのプロセスの自動起動方法

- 「3-4. 中継ゲートウェイプログラムの起動と終了」で説明した thingate プロセスは、一度起動すると、終了するまで継続して稼働する。しかしながら、Linux を一度シャットダウンした場合、何らかの方法で、次回起動時に自動起動するようにしておく必要がある。
- Linux の起動時に自動的に thingate プロセスを起動する方法は、OS のディストリビューションによって様々であるが、一般には以下の方法が利用される。
 - 簡易: /etc/rc.local ファイル (シェルスクリプト) を作成し、これに記載しておく方法。
 - 本格: UNIX 用デーモンスクリプトを記述し、"/etc/init.d/デーモン名 start" や "chkconfig デーモン名 start" 等で起動するようにして、デーモンを OS 起動時に自動起動するように設定する方法。
- 上記 2. は OS によって異なるため、以下では、1 の方法を解説する。

1. テキストエディタで /etc/rc.local を開く (まだなければ作成する)

```
sudo nano /etc/rc.local
```

2. 以下のような内容を記入 (必要に応じて追記) し、保存する。

```
#!/bin/bash  
/home/ubuntu/IPA-DN-ThinApps-Private/src/bin/thingate start
```

3. 実行権限を付与する。

```
sudo chmod 755 /etc/rc.local
```

4. reboot コマンドで Linux を再起動し、再起動後、自動的にデーモンが起動するかどうかテストする。

5-10. シン・テレワークシステムのソースコードを分析する

- 第 4 章では、単純にソースコードをビルドしたが、せっかくインストールした Visual Studio 2019 を用いると、ソースコードを快適に表示して読解・分析することができる。「src¥IPA-DNP-ThinApps-Private.sln」ファイルをダブルクリックして起動するとよい。
- ある処理がどのように実装されているのか知りたい場合は、ソースコードを検索するとよい。

