

Windows 10 The Latest :

# Linuxがほぼそのまま動くようになった「WSL2」のネットワーク機能

<https://www.atmarkit.co.jp/ait/articles/1909/09/news020.html> [この記事はPDF出力に対応していません]

2020年春に提供予定のWindows 10の機能アップデート「20H1」に実装予定のLinux環境「WSL 2」の開発が着々と進んでいる。懸案だったネットワーク機能が大幅に改良され、localhostへの対応が行われた。その変更点について解説する。

2019年09月09日 05時00分 更新

[塩田紳二, 著]

インデックス

[連載目次](#)

Windows 10上でLinux環境が実行できる「Windows Subsystem for Linux (WSL)」は、Linuxサーバを管理しているシステム管理者はもちろんのこと、Linuxの豊富なコマンドを使って文書処理を効率よく実行したいと思うユーザーにおいても、便利な機能として注目を集めている。

以下の記事で紹介したように、すでに現行のWSL（以下、WSL 1）に比べて、よりLinuxとの互換性を高めたWSL 2の開発が進んでおり、Windows Insider Previewで提供されているプレビュー版で公開されている。

- Windows 10 The Latest 「[完全なLinuxがWindows 10上で稼働する？ 『WSL 2』とは](#)」
- Windows 10 The Latest 「[『WSL 2』へのバージョンアップでLinux互換環境はどう変わるのか？](#)」

2019年8月1日に公開されたビルド18950では、WSL 2のネットワーク機能に改良が行われた。これは、以前から指摘されていた問題に対する改良であり、ユーザーの利便性、Linuxとの互換性を高める上で重要なポイントといえる。そこで本稿では、実際にビルド18950で動作確認した結果を踏まえつつ、その変更点を紹介しよう。

## WSL 2での「localhost」の問題点とは？

WSL 1は、Windowsカーネル経由でネットワークを利用していたため、WSL 1は、ホストとなるWindows 10と同じIPアドレスを持ち、同じTCP/IPスタックを利用していた。このため、Linux側（WSL 1側）、Win32側のアプリケーションは相互にlocalhostを介して通信することができた。

これに対して、WSL 2は独立したTCP/IPスタック（LinuxカーネルのTCP/IPスタック）を持つため、ホストのWindows 10と同じIPアドレスを持つわけにはいなくなっている（IPアドレスが衝突するため）。

Win32側もWSL 2側もHyper-Vの仮想スイッチ（仮想ネットワーク）を介して、それぞれが独立したネットワークノードとして動作するため、それぞれがIPアドレスを持つことになる。そのためLocalhostでは、Win32、WSL 2側のそれぞれの内部で完結してしまう。

個別のIPアドレスを持つだけならあまり問題はないが、Windows 10のHyper-Vの仮想ネットワークは、IPアドレスの割り当てをWindows 10の起動時に行い、割り当てアドレスは毎回変わってしまう。

このことからWin32側では、WSL 2側のアドレスを毎回チェックする必要があり、スクリプトなどでWSL 2側のネットワークアプリケーションへの接続を記述する、といったことがかなり面倒になる。もちろん、手動URLなどを入力するにしても、IPアドレスを調べる必要がある。

Windows Serverなどでは、DHCPサーバを動作させることができるため、Hyper-Vの内部ネットワークに対しても一定の割り当てができ、DNSサーバを使って名前でのアクセスが可能になる。しかしWindows 10には、ユーザーが自由に制御できるDHCPサーバもDNSサーバも付属していないため、システム側の自動割り当てを使うしかなく、再起動のたびに違うIPアドレス割り当てられてしまう。

ビルド18950では、WSL 2側へのTCP/IPでのアクセスに「localhost」が利用できるようになった。これにより、HTTPサーバのようなTCP/IPを利用するLinux側のサービスアプリケーションに対して、Win32側からは常にlocalhostをアドレスに使ったアクセスが可能になる。

また、WSL 2側へのアクセスをスクリプトなどに記述する場合などにも、IPアドレスを調べる必要がなく、固定したURLの表記などで済ませることが可能になった。

## Hyper-Vのネットワークを確認しておく

---

WSL 2のlocalhostアクセスの仕組みなどを説明する前に、Hyper-Vのネットワークについて確認しておく。

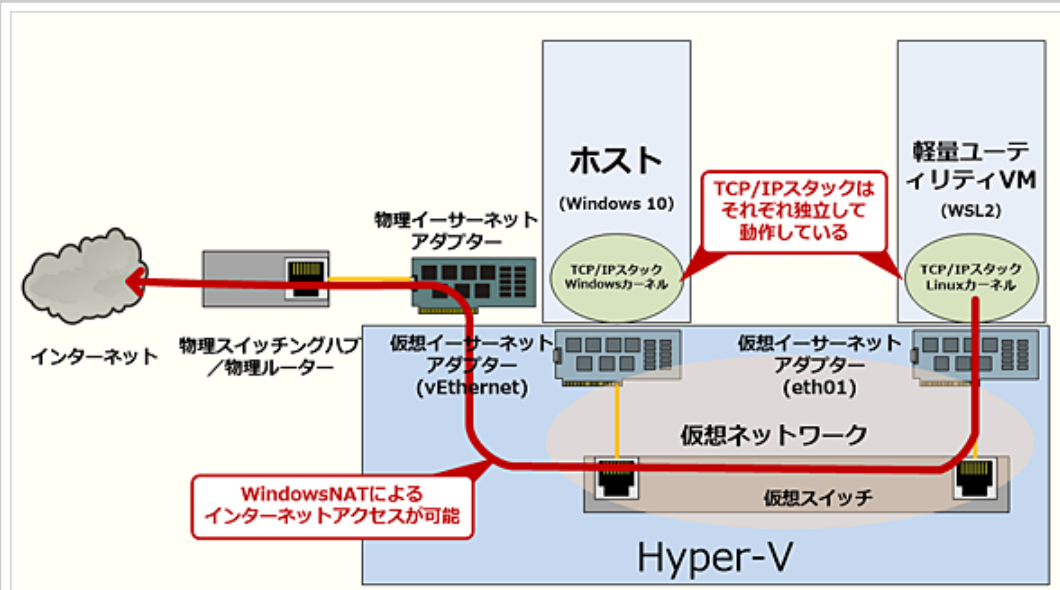
WSL 2が動作している軽量ユーティリティー仮想マシンは、Hyper-Vが管理する仮想マシン環境で動作する。Hyper-Vは、Windows 10やWindows Server上で動作するが、このとき物理ハードウェアの上で動いているWindows OSをホストWindows（ホストOSやプライマリーパーティションと呼ぶこともある）、仮想マシン内で動作しているOSをゲストOS（またはセカンダリーパーティション。こちらはWindows以外のOSのこともある）と呼んで区別する。本記事でもこの表記を使う。なお、WSL 2はゲストOS側に当たり、WSL 1/2関連のドキュメントでは、ホストWindowsを「Win32側」呼ぶことがある。

各仮想マシンに対して「仮想ネットワークアダプター（いわゆる仮想NIC、Network Interface Card）」が提供され、仮想ネットワークに接続する。この仮想ネットワークをHyper-Vでは「仮想スイッチ」と呼ぶ。

このスイッチは、いわゆるスイッチングハブのことだ。仮想スイッチには3つのタイプがある。物理的なネットワークアダプターを利用する「外部ネットワーク」、純粋な仮想ネットワークとなる「内部ネットワーク」「プライベートネットワーク」の3つである。

外部ネットワークは、物理的なネットワークカードを仮想化し、ホスト側を含めて、仮想ネットワークアダプターを介して、物理的なネットワークに接続する。これに対して、内部ネットワークとプライベートネットワークは、ホストWindowsが接続しているネットワークとは別の物理的な存在のない仮想的なネットワークとして作られる。内部とプライベートの違いは、ホストWindowsが参加しているかどうかの違いである。プライベートネットワークは仮想マシンだけが接続するネットワークだ。

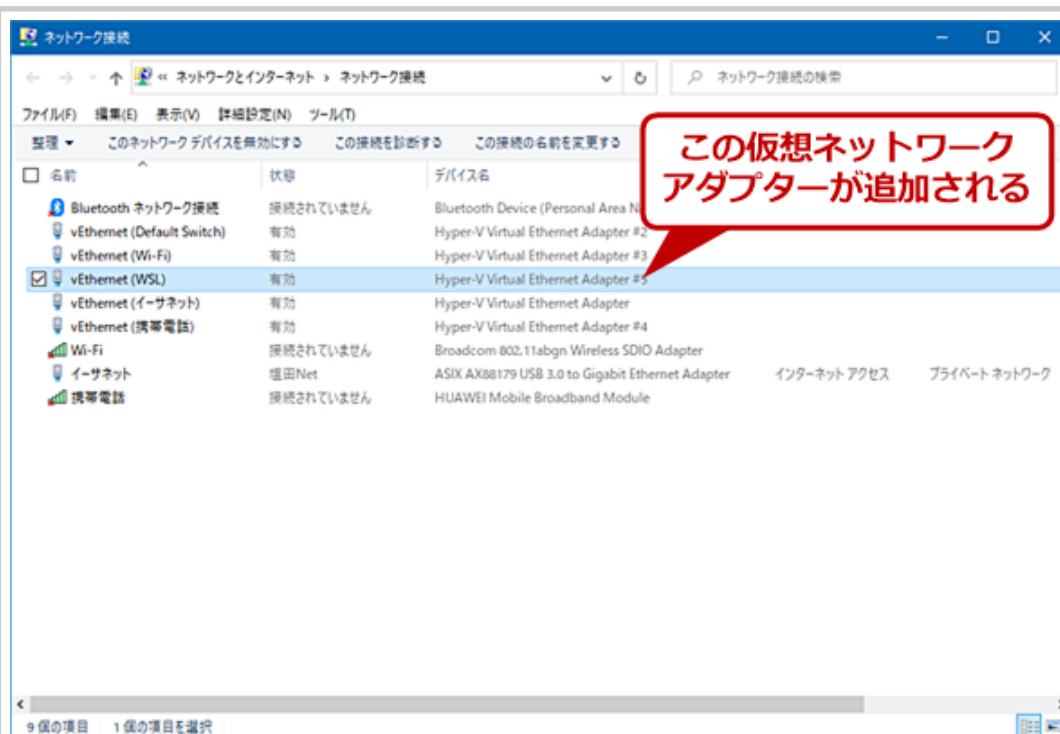
このうちWSL 2が使っているのは、「内部ネットワーク」だ。内部ネットワークには、ホストWindowsも参加する。



### Hyper-Vの内部ネットワーク

Hyper-Vの仮想スイッチの1つ「内部ネットワーク」では、仮想的なネットワークに仮想マシンとホストWindowsが共に接続した状態を作る。また、Windows NAT (WinNAT) 機能を設定することで、仮想マシン側からホストWindowsのインターネット接続を利用することも可能だ。

WSL 2を起動すると、[コントロールパネル] - [ネットワークと共有センター] - [アダプター設定の変更] に、「vEthernet(WSL)」という「Hyper-V Virtual Ethernet Adapter」が表示されるようになる。これがWSL 2用の内部ネットワークに接続するホストWindows側のネットワークアダプターになる。



### WSL 2を起動したときのWin32側ネットワークアダプターの一覧

WSL 2を起動するとWin32側（ホストWindows側）に「vEthernet(WSL)」という仮想ネットワークアダプターが表示されるようになる。

Hyper-Vの仮想ネットワークや仮想スイッチ自体は、ホストWindowsのネットワーク関連の機能では見ることができない。そもそもWindows 10には、Windows VistaやWindows 7にあったネットワーク（ネット

ワークフルマップ)を表示させる機能がなく、ネットワークに接続しているNIC(ネットワークアダプター)のみが見える状態だ。

また、内部ネットワークに関しては、Windows NAT (WinNAT)を設定することで、仮想マシンからのインターネットアクセスが可能になる。

なお、Hyper-Vの仮想ネットワークでは、デフォルトで「[プライベートアドレス](#)」が割り当てられる。プライベートアドレスとは、インターネット側には参加できないものの、社内LANのような組織内部のネットワークで自由に使用できる、特別なIPアドレスである。複数の仮想マシンが個別の仮想スイッチを使う可能性があるため、Windows 10のHyper-Vでは、IPv4アドレスとして「172.16.0.0~172.31.255.255」までのプライベートアドレスを使い、これを複数に分割して利用している。

## localhostとは

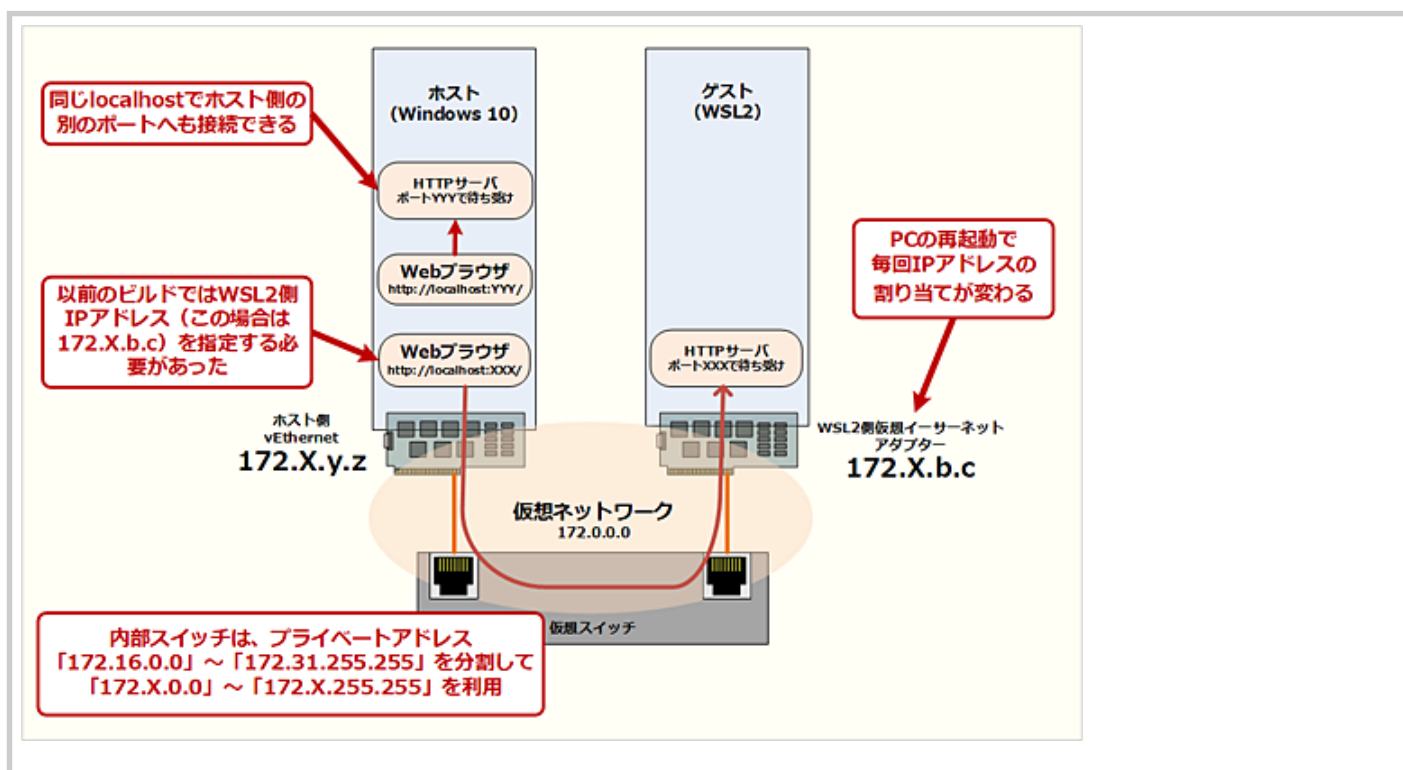
今回の改良の核心である「localhost」についても、簡単に確認しておく。

localhostとは、自分自身のIPアドレスを表す特殊な名前(ホスト名)で、「ループバックアドレス」ともいう。

IPv4では、単独のアドレスではなく、127.0.0.0という特殊なネットワーク全てがlocalhostで、慣例的に「127.0.0.1」を自分自身のIPアドレスの数値表現として使う(ただし「127.0.0.0」に属するIPアドレスは全て自分自身を表す)。基本的にどんなシステムでも、localhostという名前は、DNSサーバなどの外部の名前解決プロトコルを使わずに「127.0.0.1」に解決できる。IPv6では、「::1」(下位1ビットのみが「1」で後は「0」のアドレス)が単独でlocalhostとして指定されており、IPv4のようにlocalhostはネットワークになっていない。

通常のIPアドレスやドメイン名を指定できるのであれば、localhostは指定可能だ。例えば、Webサーバが動作しているマシンで、Webブラウザから「http://localhost/」とすれば、同じマシンで動作しているWebサーバにアクセスできる。

WSL 2が稼働しているとき、特定のポート番号で待ち受け(リッスン)している、HTTPサーバなどのTCP/IPアプリケーションに対しては、Win32側(ホストWindows側)からは、localhostを接続先として指定できる。「http://localhost:<ポート番号>/」でWSL 2側のHTTPサーバにアクセスが可能になる。

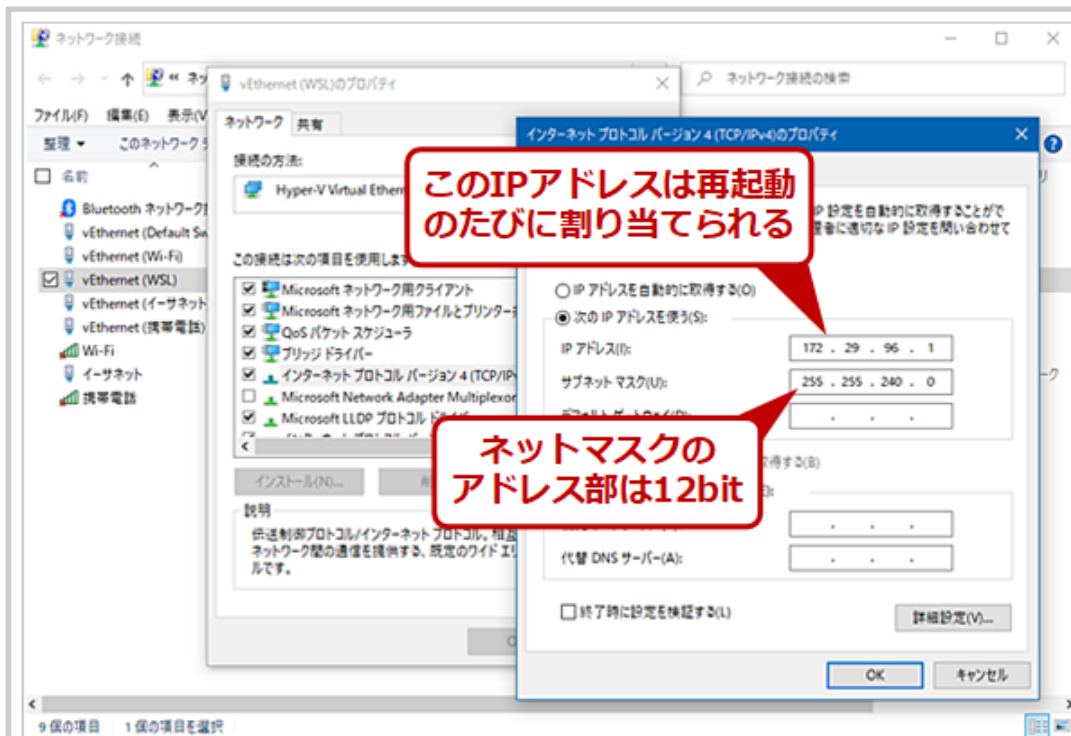


### WSL 2とWin32のネットワーク接続

Windows Insider Previewのビルド18950からは、WSL 2側のTCP/IPアプリケーションへのアクセス時のアドレス指定にlocalhostが使えるようになった。

## WSL 2のネットワークの実際

では、具体的にビルド18950のWSL 2のネットワークを見てみることにしよう。WSL 2を起動すると、ホストWindowsのネットワークアダプターに「vEthernet(WSL)」が現れる。あるマシンでは、そのアドレス割り当ては、下画面のように、IPv4アドレスが「172.29.96.1」、そのネットマスクが「255.255.240.0」となっていた。

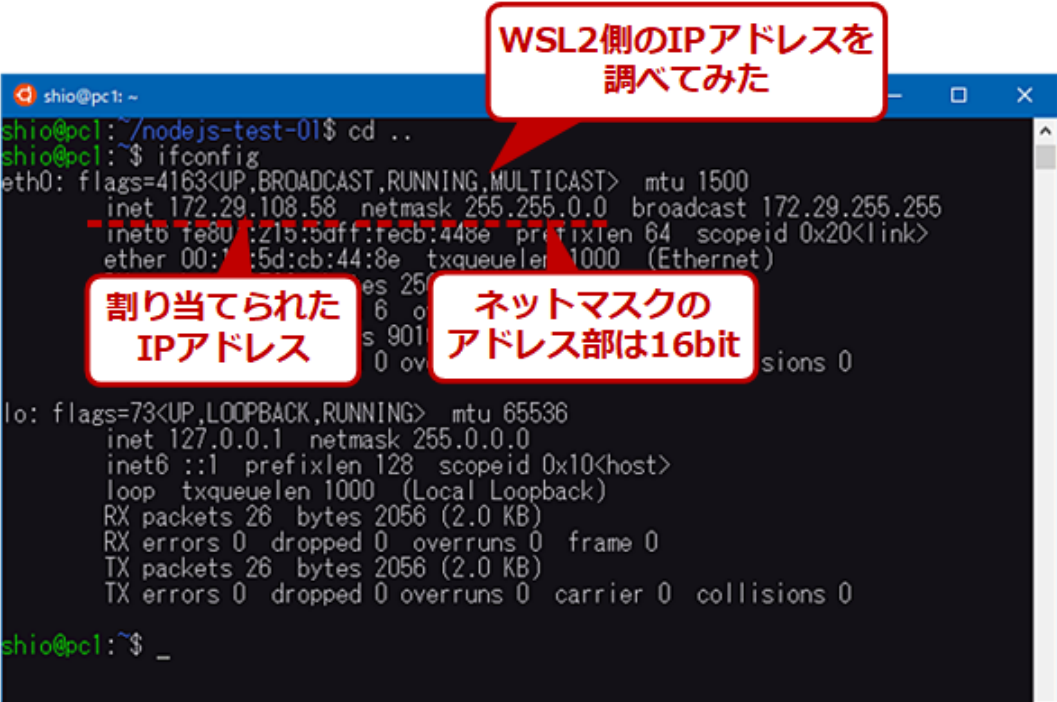


### Win32側の仮想ネットワークアダプターのIPアドレス

「vEthernet(WSL)」のIPアドレス割り当ては再起動するごとに変わる。今回は「172.29.96.1」が割り当てられている。

このときWSL 2側のネットワークインタフェース「eth0」では、それぞれ「172.29.108.58」／「255.255.0.0」という設定になっていた。両方でネットマスク（サブネットワークのIPアドレス範囲）が違っているのが気になるが、どちらも同じネットワーク（「172.29.96.0」／「255.255.240.0」すなわち「172.29.96.0～172.29.111.255」）に属している。





WSL2側のIPアドレスを調べてみた

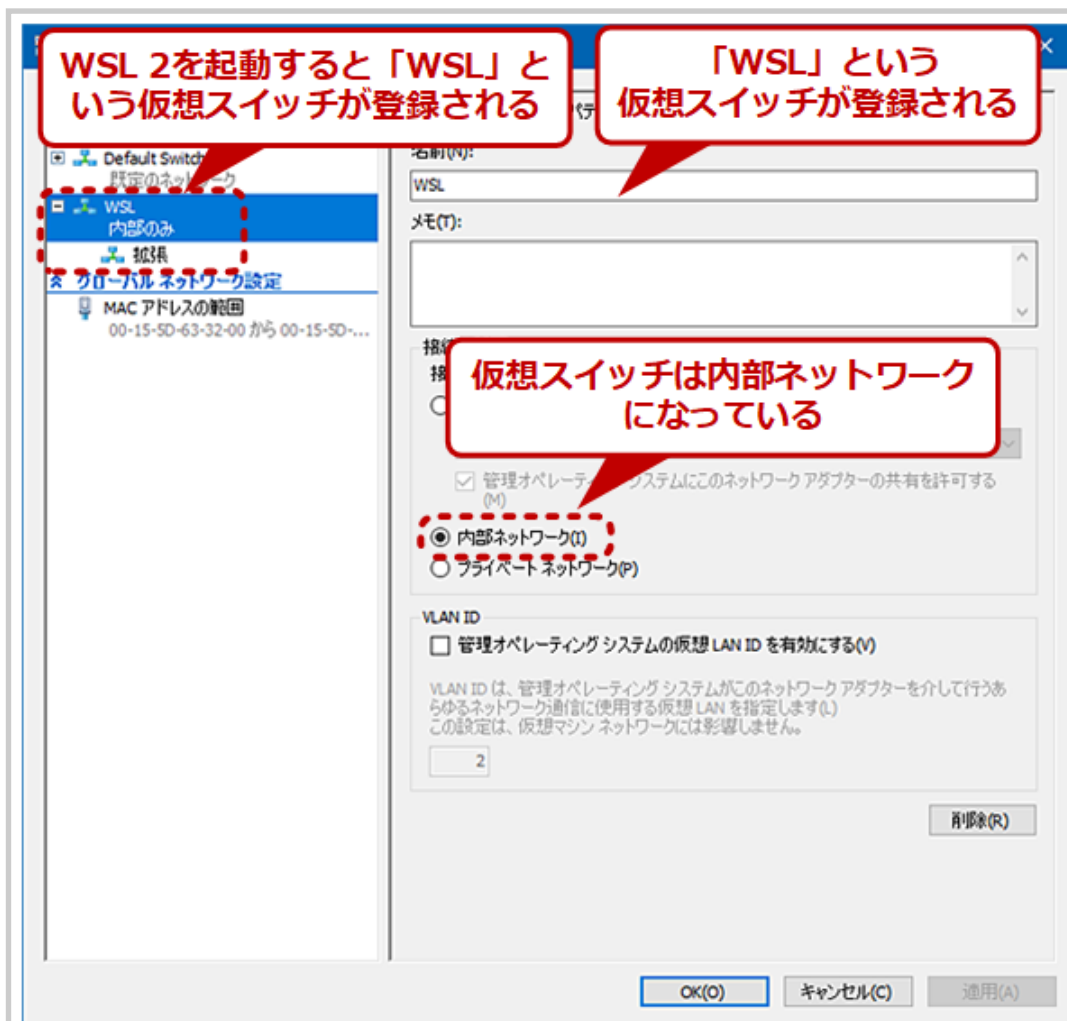
割り当てられたIPアドレス

ネットマスクのアドレス部は16bit

```
shio@pci: ~  
shio@pci: ~/nodejs-test-01$ cd ..  
shio@pci: ~$ ifconfig  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
inet 172.29.108.58 netmask 255.255.0.0 broadcast 172.29.255.255  
inet6 fe80::215:5dff:feeb:448e prefixlen 64 scopeid 0x20<link>  
ether 00:0c:29:5d:cb:44:8e txqueuelen 1000 (Ethernet)  
RX packets 26 bytes 2056 (2.0 KB)  
RX errors 0 dropped 0 overruns 0 frame 0  
TX packets 26 bytes 2056 (2.0 KB)  
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
inet 127.0.0.1 netmask 255.0.0.0  
inet6 ::1 prefixlen 128 scopeid 0x10<host>  
loop txqueuelen 1000 (Local Loopback)  
RX packets 26 bytes 2056 (2.0 KB)  
RX errors 0 dropped 0 overruns 0 frame 0  
TX packets 26 bytes 2056 (2.0 KB)  
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
shio@pci: ~$ _
```

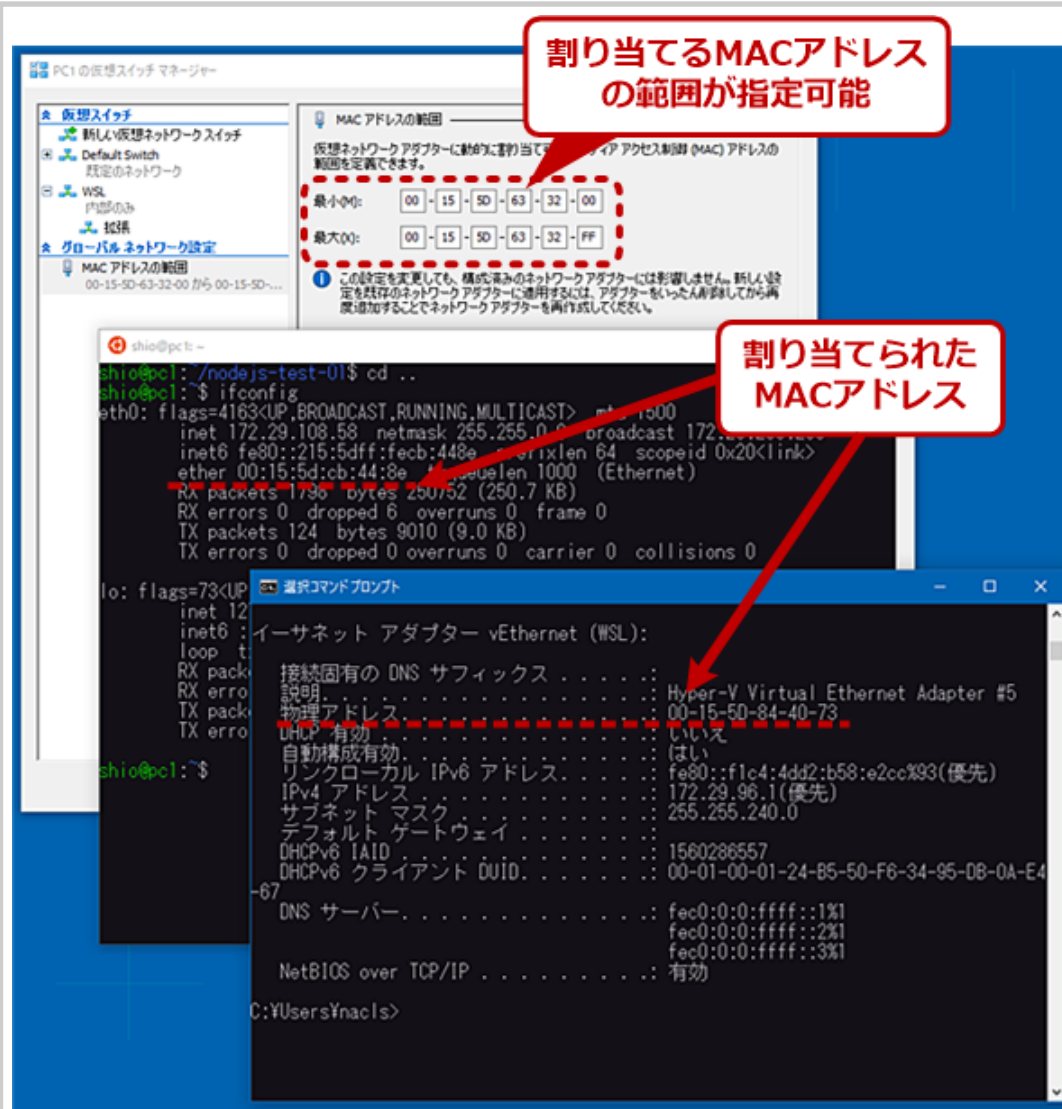
**WSL 2側のネットワークインタフェースのIPアドレス**  
WSL 2側は、IPアドレスに「172.29.108.58」が割り当てられているが、ネットマスクの値が「255.255.0.0」と「vEthernet(WSL)」側と違っていた。

また、WSL 2が起動しているとき、Hyper-Vの仮想スイッチマネージャーには、「WSL」という名前の仮想スイッチが表示されるようになる。ただし、WSL 2の仮想ネットワークアダプターや「vEthernet(WSL)」が使う仮想MACアドレスは、Hyper-Vの管理範囲外のものが使われていた。



#### WSL 2用のHyper-V仮想スイッチ「WSL」

WSL 2を起動すると、Hyper-Vの仮想スイッチマネージャーにWSLという名の仮想スイッチが登録される。



#### Win32側およびWSL 2側の仮想ネットワークアダプターのMACアドレス

Hyper-Vでは、仮想ネットワークアダプターに割り当てるMACアドレスの範囲を定義できる。だが、「vEthernet(WSL)」やWSL 2の「eth0」に割り当てられるMACアドレスはその範囲外のものが使われている。

## localhostによるWSL 2側へのアクセス方法

では、実際にlocalhostを使って、Win32からWSL 2側のTCP/IPポートへアクセスしてみよう。WSL 2側でポートの待ち受け（リッスン）をするサービスには、「[Node.js](#)」というJavaScript実行プラットフォームを利用した。Node.js上で以下のような簡単なプログラムを起動すると、ポート3000番で待ち受ける簡易なHTTPサーバが出来上がる。これに対してWebブラウザからアクセスすると、指定された文字列を返す。

```
var http = require('http');
var port = 3000;
http.createServer(function (req, result) {
  console.log('Connect:' + result.connection.remoteAddress);
  result.writeHead(200, {'Content-Type': 'text/plain'});
  var now = new Date();
  result.end('This is WSL2. ' + now.toISOString() + '\n');
}).listen(port, '0.0.0.0');
console.log('Server running at http://localhost:' + port + '/');
```

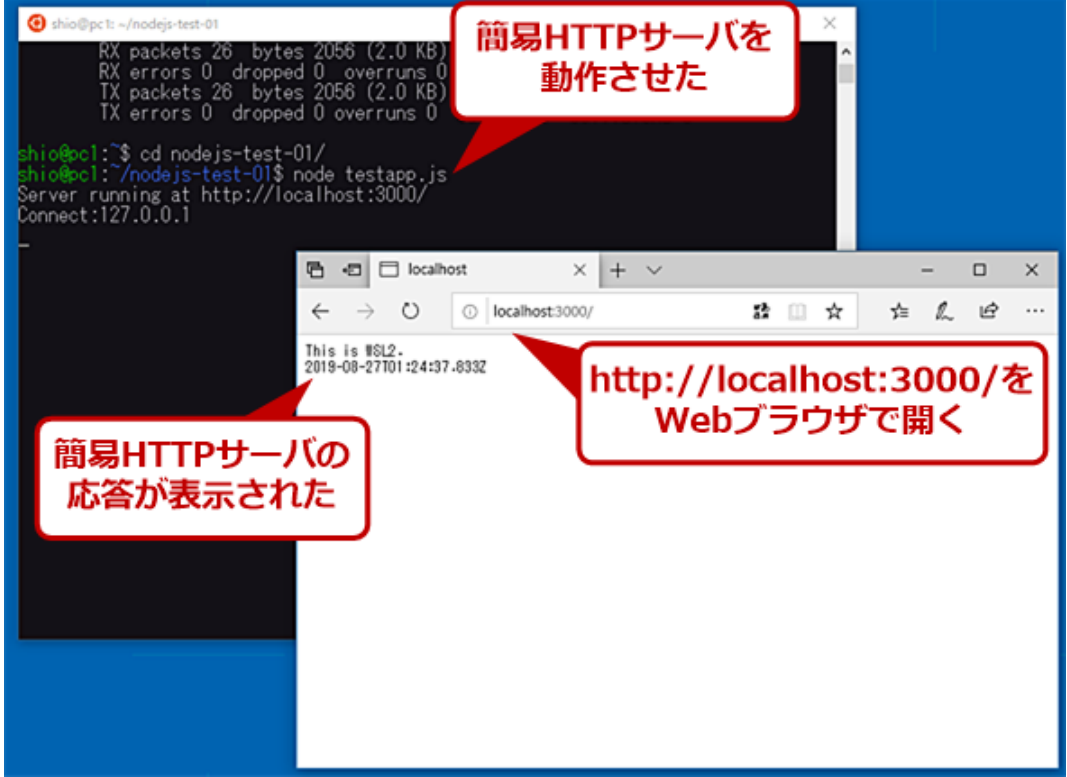
待ち受けポート番号は  
3000

「This is WSL2」という  
文字列と日付を応答する

Node.jsによる超簡易HTTPサーバ



これを動作させて、Win32側のWebブラウザから「http://localhost:3000/」にアクセスしてみたところ、WSL 2側はWin32とは異なるIPアドレスなのにもかかわらず、正しくアクセスできた。



The screenshot shows a terminal window on the left and a web browser on the right. The terminal window displays the following text:

```
shio@pci: ~/nodejs-test-01
RX packets 26 bytes 2056 (2.0 KB)
RX errors 0 dropped 0 overruns 0
TX packets 26 bytes 2056 (2.0 KB)
TX errors 0 dropped 0 overruns 0

shio@pci:~$ cd nodejs-test-01/
shio@pci:~/nodejs-test-01$ node testapp.js
Server running at http://localhost:3000/
Connect:127.0.0.1
```

The web browser window shows the URL 'localhost:3000/' and the response 'This is WSL2. 2019-08-27T01:24:37.833Z'.

Three red callout boxes with white text are overlaid on the image:

- Top right: 簡易HTTPサーバを動作させた (Started the simple HTTP server)
- Bottom left: 簡易HTTPサーバの応答が表示された (The response of the simple HTTP server was displayed)
- Bottom right: http://localhost:3000/をWebブラウザで開く (Open http://localhost:3000/ in the web browser)

**Win32からWSL 2側のHTTPサーバにlocalhostでアクセスできる**  
WSL 2側でNode.jsを使って超簡易HTTPサーバを動作させ、Win32側からWebブラウザを使いlocalhostでアクセスしてみた。具体的には、「http://localhost:3000/」をURLとして開くと、超簡易HTTPサーバの応答が表示された。

一方、逆のWSL 2からWin32側へのアクセスは、localhostでは行うことができず、「vEthernet(WSL)」ネットワークアダプターに割り当てられたIPアドレスを指定する必要がある。Win32側でIISによるHTTPサーバ（Webサーバ）がポート80番でアクセスを待ち受けている状態で、これをWSL 2の「curl」というHTTPクライアント経由でアクセスしてみると、localhostではアクセスができなかった。「vEthernet(WSL)のIPアドレス」もしくは「ホストWindowsの物理ネットワークアダプターのIPアドレス」ではアクセスが可能だった。

**WSL 2側からcurlでWin32側のvEthernetアダプターのアドレスにアクセス**

```
shio@pc1: ~$ curl -s 'http://172.29.96.1:80/' | grep '<H1>'
<H1>Home Page of PC1</H1>
shio@pc1: ~$ curl 'http://localhost:80/'
curl: (7) Failed to connect to localhost port 80: Connection refused
```

**localhostではWin32側のサービスにアクセスできない**

**Win32側IISのホームページにある<H1>が返ってきている**

Home Page of PC1

Welcome Bienvenue Tervetuloa

WSL 2からWin32側にはlocalhostでアクセスできない  
Win32側で動作しているIISのHTTPサーバに対し、WSL 2側からアクセスするには、「vEthernet(WSL)」または物理ネットワークアダプターのIPアドレスを指定する必要がある、localhostではWin32側のサービスにはアクセスできなかった。

では、WSL 2側で待ち受けようとしているポート番号がすでにWin32側で使われていたらどうなるのかもテストしてみよう。

Win32側にはすでにIISのHTTPサーバが動作し、ポート番号80を利用中である。この上で、前述のNode.jsによるHTTPサーバの待ち受けポート番号を同じ80番にして起動してみた。結果、これはエラーになった。つまり、WSL 2側としては、Win32側ですでに利用中のポートは、使用中となっていることを示す。

**ポート番号80はIISが利用しているため、WSL2側のポート80で待ち受けしようとするとエラーになる**

```
shio@pc1: ~/nodejs-test-01$ node testapp80.js
Server running at http://localhost:80/
events.js:183
    throw er; // Unhandled 'error' event
    ^

Error: listen EACCES 0.0.0.0:80
    at Object._errnoException (util.js:1022:11)
    at exceptionWithHostPort (util.js:1044:20)
    at Server.setupListenHandle [as _listen2] (net.js:1350:19)
    at listenInCluster (net.js:1408:12)
    at doListen (net.js:1517:7)
    at _combinedTickCallback (internal/process/next_tick.js:141:11)
    at process._tickCallback (internal/process/next_tick.js:180:9)
    at Function.Module.runMain (module.js:695:11)
    at startup (bootstrap_node.js:188:16)
    at bootstrap_node.js:609:3
shio@pc1: ~/nodejs-test-01$ _
```

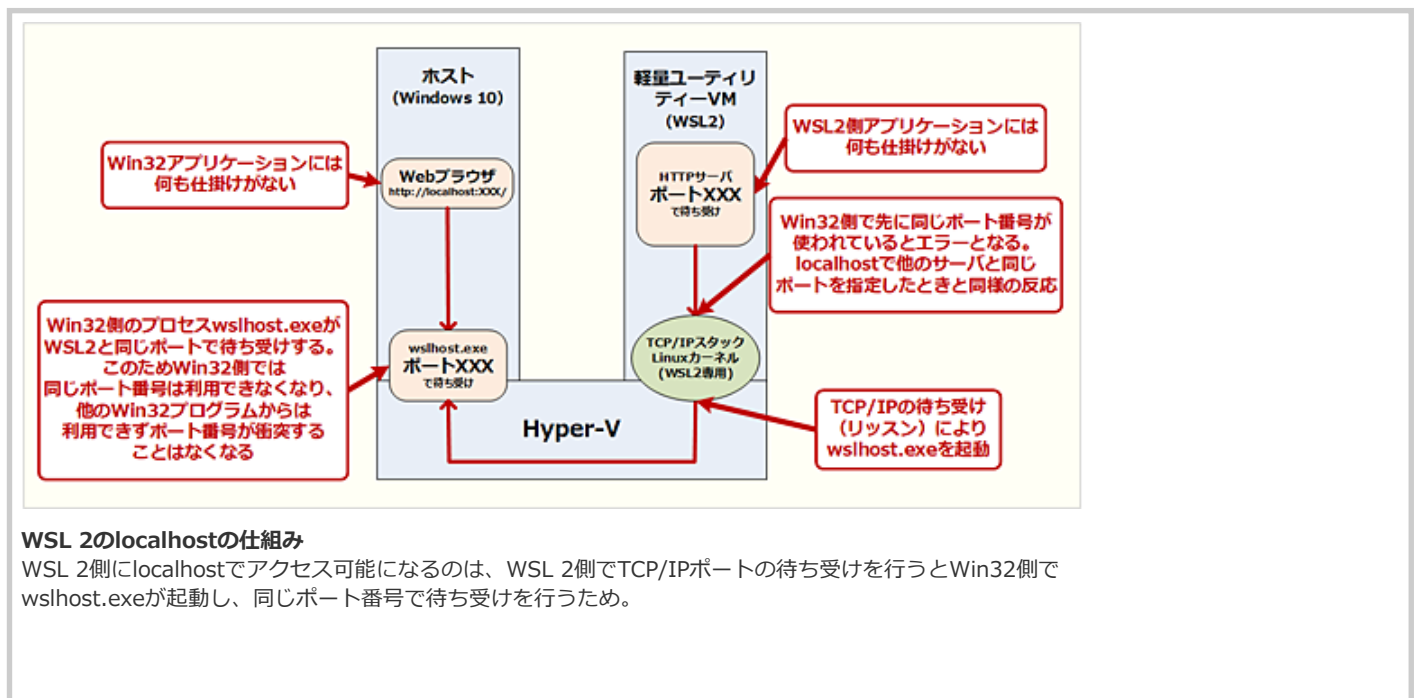
**Win32側で使用中のポート番号では、WSL 2側で待ち受け不可**

Win32側ではポート番号80（HTTPのWell Knownポート）をIISが利用している。この状態でWSL 2側のポート80で待ち受けしようするとエラーになる。

## WSL 2のlocalhostアドレスの仕組み

WSL 2が動作中に、[Win32側でnetstatコマンド](#)を使って、待ち受けポートを表示させると、wslhost.exe というプログラムが、WSL 2側の待ち受けポートと同じ番号でWin32側の待ち受けを行っている。つまり、WSL 2/Hyper-V側で、WSL 2での待ち受けを検出すると、同じポートでWin32側の待ち受けを行うwslhost.exeを起動しているようだ。

このようにすることで、該当のポート番号は、win32側で使用中となり、win32側ソフトウェアが同じポート番号を利用するのを防ぐことができる。また、wslhost.exe起動時に、すでにWin32側でポート番号が利用中であれば、エラーを返して、WSL 2で待ち受けしようとしていたソフトウェアにポートが利用中であることを示すエラーを返す。



WSL 2のネットワークにおけるlocalhostはこのような仕組みになっているのだと思われる。逆にWSL 2側でも待ち受けプロセスを起動できるなら、WSL 2上でも、localhostでWin32側サービスを利用可能になるはずだが、まだ、そこまでは実装されていないようだ。とはいえ、対称性を考えると、いずれ実装されることになるのではないかなと思われる。

□

WSL 2のlocalhost対応は、おそらく、Visual StudioのVS Code Serverを稼働させるためにも必要な機能だったため、搭載が優先して行われたと思われる。

VS Code Serverは、エディタであるVisual Studio Codeからネットワーク接続を介して、他のプラットフォーム上でコード生成などの開発作業を行うものだ。これを使うことで、Windows OS上でVS Codeを動作させ、WSLなどLinux側でコード生成やデバッグなどが可能になる。WSL 2上のVS Code Serverにlocalhostでアクセスできるようになれば、設定などを簡略化でき、ユーザーも指定が容易になるだろう。もし、localhostが利用できなければ、起動時にWSL 2側のIPアドレスを調べる仕組みが必要になるからだ。

しかし、これにより、WSL 2側にある豊富なインターネット用サービスソフトウェアがWin32から簡単にアクセスできるようになる。このときにWSL 2では、Dockerなどのコンテナが利用できるというのもメリット

トの1つだ。

20H1は、2020年春の予定であるため、これからまだ改良にかける時間はある。さらに改良が進むものと思われる。

インデックス ●●●

「[Windows 10 The Latest](#)」

Copyright© Digital Advantage Corp. All Rights Reserved.

