三重大学教育学部附属教育実践総合センター紀要　2015,　第 35 号，41 － 46 頁

# Hyper-V WMI クラスを用いた仮想マシン群の構築

丁　亜希*・山守　一徳*

集中型のネットワーク関連演習環境として、我々は PowerShell コマンドレットを用いて、演習に必要な台数分の Linux 仮想マシンを自動的に作成し再設定を行う方法をこれまでに考案した。この方法では、Hyper-V 管理ホストから仮想マシンへ、共用仮想ハードディスクを使って、仮想マシンの再設定に必要なデータを引き渡し、仮想マシン側で再設定を行うことになっている。そのため、実運用時に必要としない共用仮想ハードディスクを作成しなければならない。

そこで、本研究では、Hyper-V の統合サービスを利用して、管理ホスト側で直接に仮想マシンの再設定を行う構築方法を考案し、CentOS 7 を仮想マシンとした演習環境の構築実験を行った。これにより、仮想マシンの作成と再設定の作業は以前の方法より効率が良いことが分かった。

**キーワード**：仮想化、Hyper-V、PowerShell、WMI クラス

## 1．はじめに

ネットワーク関連演習の授業環境として、我々は演習受講者数に合わせて必要な台数分の仮想マシンを自動的に作成および再設定を行う方法を考案した[1)]。これは Hyper-V 仮想化システムを用いて、1 台の Windows 8.1 コンピュータに CentOS 6.4 が稼働する仮想マシン群を 1 つの命令で作成し、集中型の演習用仮想環境を作り出すものであった。昨年度、本学部の「計算機ネットワーク」授業で、通信ソフトウェアのプログラミング作業などを含む 20 名程度の小規模授業の演習用仮想環境としての有用性を実証してきた。

この方法では、まず、テンプレートと呼ばれる 1 台の仮想マシンを作成し、授業や演習内容に合わせて各種の環境設定を行う。そして、管理ホスト（Hyper-V 仮想化システムが稼働する Windows マシン）上で、予め用意した PowerShell スクリプトを実行させ、テンプレートの仮想ハードディスクをコピーすることによって複数台の新しい仮想マシン（以降、これらの仮想マシンをゲストとも呼ぶことにする）を作成する。コピーによって作られたので、新しいゲストの IP アドレスなどのネットワーク環境設定はテンプレートのそれと同じようになっている。そのままでは使えないから、各ゲストを一度起動してネットワーク環境の再設定を行わなければならない。そのため、管理ホストから、ゲストの再設定に必要な情報、IP アドレスやホスト名などをゲストに知らせる必要がある。そこで、1 つの共用仮想ハードディスクを設置し、それを介して管理ホストとゲスト間のデータ引き渡しを実現した。

*三重大学教育学部

しかしながら、その共用仮想ハードディスクは、管理ホストとゲストが共用できるが、クラスター（Windows 8.1 はクライアント Hyper-V なので使えない）を構成しない環境では、同時に使用することができない。交代でしか使えないから、マウントとアンマウント操作が多く、効率が悪い。元々の演習用環境ではそのような共用仮想ハードディスクが必要としない場合でも、データ引き渡しの目的だけで設置しなければならない。データ引き渡し後、ゲスト側で一度しか実行されない設定用プログラムを実行させ、設定を行う。テンプレートとなるマシンでも、再起動したらこの設定用プログラムが実行されるので、再起動してはならない。そのため、一旦テンプレートが作成されたら、通常の仮想マシンとして使えなくなるという欠点があった。

そこで、本研究では、Hyper-V の統合サービスを利用して、管理ホスト側で直接、ゲストのネットワーク環境の再設定を行う新しい構築方法を考案する。データ引き渡し用の共用仮想ハードディスクを使う必要がなく、ゲスト側の設定用プログラムも要らない。この新しい方法では Hyper-V 統合サービスを利用するので、実験対象となる仮想マシンには、2014 年 7 月に新たに配布され、統合サービスを含む Hyper-V へのサポート状況[2)] がより進歩した CentOS 7 がインストールされたものとする。

## 2．Hyper-V 統合サービスと CentOS 7

Hyper-V 統合サービスとは、仮想マシンを管理したり、性能を強化したりするためのいくつかのユーティリティのことである。これらのユーティリティは、管理ホスト側とゲスト側との両方で同時に有効化される時のみ、機能することになる。

管理ホスト側では、Hyper-V 環境が有効化されたら、統合サービスは自動的に使えるようになっている。

ゲスト側の CentOS 7 では、デフォルトで統合サービスの一部しか有効化されてない。統合サービスの有効化状況はコマンドレット Get-VMIntegrationService で確認することができる。PrimaryStatusDescription 欄に「コンタクトなし」が表示されると、Enabled 欄が True になっていてもゲスト側での対応プログラムが稼働されてないことを示す。CentOS 7 の初期状態では、管理ホスト側からゲストのネットワーク設定を行うために必要なキー値ペア交換（KVP: Key Value Pair Exchange）サービスが有効化されてないことがわかる。それを有効化するには、ゲスト側で、下記のようにパッケージ hypervkvpd をインストールして起動する必要がある。

```
yum -y install hypervkvpd
systemctl start hypervkvpd
```

キー値ペア交換サービスが正しく有効化されたら、管理ホスト側で Get-VMNetworkAdapter または Hyper-V マネージャを実行すれば、ゲスト側の IP アドレスが見えるようになる。この状態になれば、後述の方法で管理ホスト側からゲストのネットワーク設定を行うことが可能になる。

一方、以前の配布バージョンと比べ、CentOS 7 の一つの大きな変更は、NetworkManager という動的ネットワーク管理システムが主なる管理方式になり、従来のネットワーク管理方式（ifcfg-ethx 型設定ファイルとスクリプトの使用）が補助的な位置に入れ替えられたことである[3)]。通常、全てのネットワーク設定は NetworkManager を介して行うことになるが、一旦従来の方式で設定し直したら、デバイスが NetworkManager の制御下から外れることになる。こうなると、nmcli など NetworkManager 命令は正常に機能しなくなる。一般的な用途ならば NetworkManager を使わずに CentOS 7 を運用しても構わないが、授業や演習に利用する場合は問題になっている。授業や演習では、できるだけ学生に新しい技術を勉強させたいからである。このような場合は、NetworkManager の制御から外されないように方策を考えなければならない。

C 言語で書かれた hypervkvpd のソースプログラムを読んで、ゲスト側でのネットワーク設定作業が分かった。その仕組みを図 1 に示す。hypervkvpd が netlink ソケット経由で管理ホストからネットワーク構成データを受け取り、それらの情報を/var/lib/hyperv/ifcfg-eth0 ファイルに書き込む。その後、hypervkvpd が ifcfg-eth0 ファイルを引数として bash のスクリプト hv_set_ifconfig を

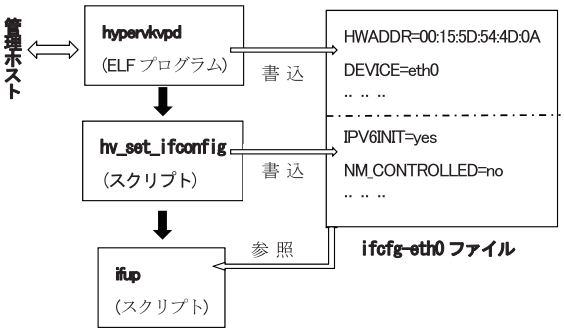


**図 1. ゲスト側でのネットワーク設定作業の流れ**

起動する。hv_set_ifconfig が従来方式の ifup スクリプトを使ってネットワークの設定操作を行う。そのため、デフォルトとしてキー値ペア交換サービスによるゲストのネットワーク設定では、このようにネットワーク・デバイスが NetworkManager の制御から外されることになる。

hv_set_ifconfig の動作をさらに詳しく調べてみると、起動時引数の ifcfg-ethx ファイルには既に hypervkvpd によって IP アドレスやデフォルト・ルータなどの情報が書き込まれている。hv_set_ifconfig がさらにこの ifcfg-ethx ファイルに

```
echo "IPV6INIT=yes" >> $1
echo "NM_CONTROLLED=no" >> $1
```

などの命令によってその他のネットワーク設定関連情報を加え、/etc/sysconfig/network-scripts にコピーしてから、ifup を起動する*。

従って、問題を解決するために、

```
echo "NM_CONTROLLED=no" >> $1
```

を

```
echo "NM_CONTROLLED=yes" >> $1
```

に変更すれば、設定対象のデバイスが NetworkManager の制御下から外れないことになるであろう。

hypervkvpd の設計では、このようにシステム管理者が hypervkvpd を再コンパイルせずに、スクリプトだけを編集して、設定方法を選択し調整することができるようになっている。因みに、IPv6 を使わない場合、同ファイル中の IPV6INIT=yes を no に変更すればよい。

## 3．Hyper-V WMI クラスとゲスト設定方法

WMI（Windows Management Instrumentation）クラスは、いろいろなウィンドウズシステム管理に使えるクラスで、PowerShell などのスクリプトにも取り込むことができ、幅広く使われている。Hyper-V WMI クラ

*ifup は ifcfg-ethx 中の GATEWAY=addr の記述に対して正しくデフォルト・ルータの設定ができなかった。sed -i -e 's/GATEWAY=/GATEWAY0=/' $1 をスクリプトに追加したら問題が解決した。

ス[4] はその構成の一部として、仮想化システムの管理に使われている。

管理ホスト側からゲストのネットワーク設定を行うためには、Hyper-V レプリケーション機能を利用する場合、Set-VMNetworkAdapterFailoverConfiguration コマンドレット、または Msvm_ReplicationService クラスの SetFailoverNetworkAdapterSettings メソッドを使ってよい。授業環境としての仮想マシン群では、レプリカを作らないので、Msvm_VirtualSystemManagementService クラスの SetGuestNetworkAdapterConfiguration メソッドを使うことにする。管理ホスト側で、コマンドレット Get-WmiObject を使ってインスタンスを次のように生成する。

```
$VSMS = Get-WmiObject
    -Namespace root¥virtualization¥v2
    -Class Msvm_VirtualSystemManagementService
```

次に $VSMS.SetGuestNetworkAdapterConfiguration を呼び出すのに必要な二つの引数を用意する。

一番目の引数は、ネットワーク設定対象となるゲストの指定である。CIM_ComputerSystem クラスのインスタンスへの参照でなければならない。

ここで、CIM_ComputerSystem クラスを継承した Msvm_ComputerSystem クラスを利用して設定対象のゲスト情報を取得する。

```
$CS = Get-WmiObject -Class Msvm_ComputerSystem
  -Namespace root¥virtualization¥v2
  -Filter "ElementName=ゲスト名"
```

全システムに管理ホストを含む複数のコンピュータが存在しているので、フィルタでプロパティ ElementName が設定対象ゲストであるオブジェクトだけを抽出する。

二番目の引数は、IP アドレスなどの設定内容を指定する。Msvm_GuestNetworkAdapterConfiguration クラスのインスタンス（ネットワーク・アダプタのオブジェクト）のテキスト表現でなければならない。

```
$CONFS = Get-WmiObject
  -Namespace root¥virtualization¥v2
  -Class Msvm_GuestNetworkAdapterConfiguration
```

これで、$CONFS に全システム中でのすべてのネットワーク・アダプタのオブジェクトが入ってくる。当然ながら、その中には、設定対象ゲスト以外のアダプタも含まれる。設定対象となるアダプタを探し出さねばならない。簡単のために、対象ゲストには 1 つだけのネットワーク・アダプタがしか存在しないとする。

```
foreach ( $CONF in $CONFS ) {
  if ( $CONF.InstanceID.Contains($CS.name) ) {
    break
  }
}
```

各アダプター・オブジェクトを順次に調べ、プロパティ InstanceID に設定対象ゲスト名$CS.name が含まれば、そのアダプタを設定対象とし、調べ作業を終了させる。$CONF には下記のような設定項目があり、それらに設定の必要に応じて新しいデータを書き込む。

```
$CONF.DefaultGateways   デフォルト・ルータの指定
$CONF.DHCPEnabled   動的 IP アドレス取得かの指定
$CONF.DNSServers    DNS サーバの指定
$CONF.IPAddresses   IP アドレスの指定
$CONF.ProtocolIFType   IPv4 か IPv6 かなどの指定
$CONF.Subnets       サブネット・マスクの指定
```

以上で、二番目引数に必要なオブジェクトを作った。要求される書式のテキスト表現にするには、クラス ManagementBaseObject から継承したメソッド GetText を使う。$CONF.GetText（1）はマネージオブジェクト形式を返してくる。

最後に、SetGuestNetworkAdapterConfiguration メソッドを呼び出し、ゲストの hypervkvpd に設定項目を渡し、ネットワーク設定を行うことができる。

## 4．仮想マシン群の構築

文献 1 と同じように、仮想マシン群を構築するための PowerShell スクリプト（付録参照）を作成する。1 台のテンプレート仮想マシンを予め準備し、このスクリプトを使って必要な台数の仮想マシンを自動的に複製する。スクリプトの外部仕様がほぼ同じであるが、内部的には WMI クラスを用いて管理ホストから複製されたゲストのネットワーク再設定を行う。そのため、引数リストにサブネット・マスク、デフォルト・ルータおよび DNS サーバを追加する。共用仮想ハードディスクを使わないから、そのための引数を削除する。引数リストを以下にまとめて記す。

```
-fromVM          テンプレートの指定、必須
-startIPaddress  仮想サーバの開始 IP アドレス、必須
-SubnetMask      サブネット・マスクの指定
-DefaultGateway  デフォルト・ルータの指定
-DNSServers      DNS サーバの指定
-toPath    複製先フォルダの指定
-prefix    複製される仮想マシン名の前半の指定
-suffix    複製される仮想マシン名の後半の指定
-number    複製台数の指定
```

スクリプトは、前処理部と繰返し部との 2 つの部分から構成される。前処理部は、仮想マシン複製前の初期処理や共通データを準備する。繰返し部は、1 台 1 台の仮想マシンを複製して再設定を行う。

前処理部では、変数$hdd にテンプレートの仮想ハードディスクを、$switchname に仮想スイッチを、$mac に現在既存仮想マシンに使われる MAC アドレスの最大値を、$ip に開始 IP アドレスの数値データ（10 進数）をそれぞれ格納する。アドレスを数値データに変換して変数に保持するのは、後での＋1 演算がしやすくなるためである。

繰返し部では、仮想マシン生成部とネットワーク設定部との 2 つの部分からなる。仮想マシン生成部では、まず、仮想マシンのホスト名、仮想ハードディスクの複製先、MAC アドレス、IP アドレスを決定する。次に、テンプレートの仮想ハードディスクをコピーして仮想マシンを作成してから、主記憶、ネットワーク・アダプタおよび MAC アドレスを設定する。ネットワーク設定部では、新しく生成された仮想マシンを起動して、前述のキー値ペア交換サービスによってネットワーク設定を行う。但し、ここでは、まだ二つの問題が存在している。

1 つ目は、SetGuestNetworkAdapterConfiguration メソッドの実行はゲスト側で有効なネットワーク・アダプタの存在が必要である。一方、新たに作られたゲストが起動されてからアダプタ有効化になるまで一定の時間がかかる。従って、スクリプトはこの時間を待たなければならない。そのため、Get-VMNetworkAdapter コマンドレットでゲストのアダプタ情報を取得し、その IPaddresses 情報を調べ、アダプタが有効化されていない場合は、5 秒間待つ。

```
$IPs = Get-VMNetworkAdapter $new
while([string]::IsNullOrEmpty($IPs.IPaddresses))
{ sleep 5 }
```

CentOS 7 では、新しいアダプタが有効化になると少なくとも IPv6 のアドレスが設定されている。

もう 1 つの問題は、キー値ペア交換サービスによるゲストのネットワーク設定では、残念ながら、ホスト名の設定がサポートされてない。それは、管理ホストからゲストへのネットワーク設定がそもそもフェイルオーバーのための技術であり、フェイルオーバーの場合、ホスト名変更が必要ないからであろう。しかし、そのままでは、複製された仮想マシンのホスト名はすべてテンプレートと同じホスト名になり、それぞれのゲストに一度ログインして手作業でホスト名を再設定しなければならないので不便である。1 つ解決方法として、前述の hv_set_ifconfig スクリプトを利用することが考えられる。例えば、テンプレートの hv_set_ifconfig の最後行に、次のように 4 行を追加する。IP アドレスからホスト名を派生して、ホスト名を設定することになる。

```
foo=`grep IPADDR $1 | cut -d"." -f4`
HN="vm"$foo".edu.mie-u.ac.jp"
hostname $HN
echo $HN > /etc/hostname
```

## 5．実験と考察

CPU が i7-3930K、主記憶容量が 32 GB、OS が Windows Server 2012 R2 である管理ホストで、仮想マシン群の自動生成に関する 3 つの実験を行った。実験 1 では、スクリプトは正しく動作できるかを確認する。実験 2 と実験 3 では、より効率的に仮想マシン群を生成するために、スクリプトの並行化を試す。文献 1 の方法と違って、交代でしか使えない共用仮想ハードディスクを利用しないから、並行化が可能になるのである。

### 実験 1　スクリプトの動作確認

テンプレートとなる仮想マシン CentOS7 を Hyper-V マネージャで予め用意しておく。一度ログインして授業や演習内容に合わせて各種の設定を行う。後で複製される複数の仮想マシンの内容を一致させるためにテンプレートを一旦シャットダウンする。

PowerShell を管理者として起動して、次のコマンドレットを実行し、5 台の仮想マシンを自動的に生成する。

```
./Make-VM.ps1 -Number 5 -FromVM centos7
    -StartIPaddress 133.67.84.21 -Suffix 21
```

図 2 に、PowerShell スクリプトを使って 5 台の仮想マシンを生成する実験を示す。Get-VMNetworkAdapter コマンドレットを実行して、5 台の仮想マシン名が vm21 から vm25、MAC アドレスは 00155D545103～07、IP アドレスは 133.67.84.21～25 になっていることが確認できた。また、生成された仮想マシンにログインして、hostname や nmcli d show eth0 を実行すると、ホスト名の設定が正しく設定されたこと、ネットワーク・デバイスが Network Manager の管理下であることが確認できた。

Get-VM コマンドレットで仮想マシンの一覧（図 2）を表示すると、Uptime 欄を見ればわかるように、新しく生成された仮想マシンの起動間隔は約 70 秒である。それは、1 台のマシンの生成に必要な時間を示している。詳しく調べるために、スクリプトにタイムスタンプを記録することによって、仮想マシン生成の各段階にかかる時間を計測した。5 台の仮想マシンを生成する場合は、平均で 1 台生成にあたり、仮想ハードディスクコピーに

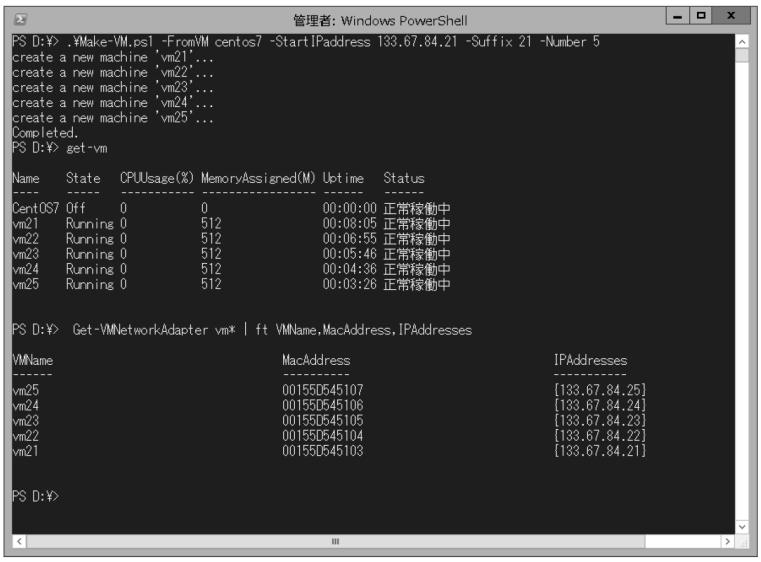


**図 2. 仮想マシン自動生成スクリプトの動作確認**

は約 4 秒、仮想マシン生成には約 27 秒、ネットワーク設定には約 39 秒、それぞれかかったことが分かった。テンプレートマシンの仮想ハードディスクの大きさは約 3.2 GB であって、Measure-Command で計測すると、実験に使われるマシンでは、それのコピーに 47 秒かかることが分かる。しかし、実際にスクリプトで生成した場合 4 秒しか要らない。これは、Windows のスーパーフェッチ機能により、コピー内容がキャッシュ済であると解釈してよいであろう。

文献 1 では、仮想ハードディスクの大きさが 5 GB で、1 台の仮想マシンの生成には約 150 秒かかる、と報告されていた。ディスク大きさの違いがあっても、今回の方法では、仮想マシン生成時間がかなり短縮できたと言えよう。

### 実験 2　ネットワーク設定部だけの並行化

ネットワーク設定部にかかった時間の 39 秒は、殆ど仮想マシンの起動およびネットワーク・アダプタ有効化なるまでの待ち時間である。従って、仮想マシン生成部が終了したら、ネットワーク設定部をバックグラウンド・プロセスとして実行させ、待っている間に次の仮想マシン生成部を実行し始めると、スクリプトの実行全時間が短縮すると考えられる。

ネットワーク設定部を切り出し、別名のスクリプト例えば bp.ps1 とおく。それをバックグラウンド・プロセスとして実行させるには、いろいろな方法があるが、ここで、

```
$args="-File ./bp.ps1", "引数 1", "引数 2",…
Start-process powershell
        -ArgumentList $args -NoNewWindow
```

を元のスクリプトに記述するようにする。

実験 1 と同じように、タイムスタンプを記録することによって、5 台の仮想マシン生成の開始から最後のバックグラウンド・プロセスの終了までの経過時間は 267 秒であったことが分かった。この場合は、1 台仮想マシンの平均生成時間は 53 秒、つまり約 24%短縮されたことになる。

### 実験 3　仮想マシン生成部からの並行化

実験 3 では、並行化の範囲をさらに拡大して、仮想マシン生成からネットワーク設定までの並行化を試みた。同じく 5 台の仮想マシン生成の開始から最後のバックグラウンド・プロセスの終了までの経過時間は 290 秒になった。全体の時間は、実験 1 より 60 秒短縮されたが、実験 2 より逆に 23 秒長くなっていた。タイムスタンプによれば、5 つのバックグラウンド・プロセスがほぼ同時に仮想ハードディスクのコピーを始め、コピーに要する時間が 40 秒〜42 秒であって、長くなったことが分かった。これは、一つのハードディスクに対する複数操作においては、並行処理が逆効果になるという一般的な認識に一致した結果であった。

## 6．おわりに

通常のコンピュータ端末室では、一度構築した授業環境は比較的に長い期間に使い続けることが多い。しかし、ネットワーク関連授業と演習では、できるだけ新しい技術を取り込む必要があり、毎年異なる授業や演習内容に合わせた新しい環境が要求される。また、演習中の誤操作によるシステムが稼働不能になったりすることもあって、頻繁に再構築する必要がある。そのため、より手軽に使え、自動化性能の高い構築方法が期待されている。

本研究で実現した仮想マシン群の自動生成方法では、仮想マシン生成に必要な時間を大きく短縮しただけでなく、仮想マシン側でネットワーク等の再設定用プログラムが不要なので、テンプレートの準備作業もかなり軽減されている。

これからの授業や演習等教育実践の中で、開発された PowerShell スクリプトの便利さが十分に発揮されると期待している。

## 参考文献

(1) 丁亜希，山守一徳：ネットワーク演習ための仮想サーバ環境構築，三重大学教育学部附属教育実践総合セン

ター紀要，第 34 号，13-18，2014 年 3 月.
(2) マイクロソフト社：CentOS サポート情報，http://technet.microsoft.com/en-US/library/dn531026.aspx，2014 年 12 月参照.
(3) レッドハット社：Networking Guide, https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Networking_Guide/index.html，2014 年 12 月参照.
(4) マイクロソフト社：Hyper-V WMI classes, http://msdn.microsoft.com/en-us/library/hh 850078(v=vs.85).aspx，2014 年 12 月参照.

付録 PoweShell スクリプト Make-VM.ps1 without error check

```
#引数リスト
param(
    [Parameter(Mandatory=$True)]
    [string]$FromVM,
    [Parameter(Mandatory=$True)]
    [string]$StartIPaddress,
    [string]$SubnetMask="255.255.255.0",
    [string]$DefaultGateway="133.67.84.254",
    [string]$DNSServers="133.67.88.5",
    [string]$ToPath = (Get-VMHost).VirtualHardDiskPath,
    [string]$Prefix = "vm",
    [int]$Suffix = 1,
    [int]$Number = 1
)
#前処理部
$ErrorActionPreference = "stop"
$hdd = (Get-VMHardDiskDrive "$FromVM" `
      -ControllerType IDE -ControllerNumber 0).Path
$switchname = (Get-VMNetworkAdapter "$FromVM").SwitchName
$mac = (Get-VMNetworkAdapter "$FromVM").MacAddress
foreach ($aa in (Get-VM | Get-VMNetworkAdapter).MacAddress)
       { if( $aa -gt $mac ){ $mac = $aa } }
$ip = ([IPAddress]$StartIPaddress).Address
#繰返し部
for ( $ii=0; $ii -Lt $Number; ++$ii){
  #仮想マシン生成部
  $new = "$Prefix" + ("{0:00}" -f ($Suffix+$ii))
  $newpath = "$ToPath¥$new"
  if(!(Test-Path "$newpath" )){mkdir "$newpath"|Out-null}
  $newhdd = "$newpath¥$new.vhdx"
  $mac = "{0:x12}" -f ( 1 + "0x$mac" )
  $IPaddress = ([IPAddress]$ip).IPAddressToString
  $ip = $ip + 16777216
  write "create a new machine '$new'..."
  Copy-Item "$hdd" "$newhdd" -Force
  New-VM "$new" -Path "$ToPath" -VHDPath "$newhdd"|Out-null
  Set-VMMemory "$new" -DynamicMemoryEnabled $true`
    -MinimumBytes 512 MB -StartupBytes 512 MB
  Remove-VMNetworkAdapter -VMName "$new"
  Add-VMNetworkAdapter -VMName "$new" `
                       -SwitchName "$switchname"
  Set-VMNetworkAdapter -VMName "$new" `
                       -StaticMacAddress $mac
  #ネットワーク設定部
  Start-vm $new
  Get-VMIntegrationService $new|Enable-VMIntegrationService
  $CS = Get-WmiObject -Namespace root¥virtualization¥v2`
  -Class Msvm_ComputerSystem -Filter "ElementName='$new'"
  $CONFS = Get-WmiObject -Namespace root¥virtualization¥v2`
           -Class Msvm_GuestNetworkAdapterConfiguration
  foreach ( $CONF in $CONFS ) {
     if($CONF.InstanceID.Contains($CS.name)){break}}
  $CONF.DefaultGateways = @("$DefaultGateway")
  $CONF.DHCPEnabled = $false
  $CONF.DNSServers = @("$DNSServers")
  $CONF.IPAddresses = @("$IPaddress")
  $CONF.ProtocolIFType = 4096
  $CONF.Subnets = @("$SubnetMask")
  $VSMS = Get-WmiObject -Namespace root¥virtualization¥v2`
           -Class Msvm_VirtualSystemManagementService
  $IPs = Get-VMNetworkAdapter $new
  while( [string]::IsNullOrEmpty($IPs.IPaddresses))
       {sleep 5}
  $VSMS.SetGuestNetworkAdapterConfiguration(`
           $CS.path,$CONF.GetText(1)) | Out-null
}
write "Completed."
```