FortiProxy - Cloud Deployment Guide (Microsoft Azure)

Version 1.2.8

FORTINET DOCUMENT LIBRARY
https://docs.fortinet.com

FORTINET VIDEO GUIDE
https://video.fortinet.com

FORTINET BLOG
https://blog.fortinet.com

CUSTOMER SERVICE & SUPPORT
https://support.fortinet.com

FORTINET TRAINING & CERTIFICATION PROGRAM
https://www.fortinet.com/support-and-training/training.html

NSE INSTITUTE
https://training.fortinet.com

FORTIGUARD CENTER
https://fortiguard.com/

END USER LICENSE AGREEMENT
https://www.fortinet.com/doc/legal/EULA.pdf

FEEDBACK
Email: techdoc@fortinet.com

November 2, 2020
FortiProxy 1.2.8 Cloud Deployment Guide (Microsoft Azure)
45-128-674631-20201102

TABLE OF CONTENTS

Change Log

Date	Change Description
November 2, 2020	Initial release

Overview

Microsoft Azure is a cloud computing service created by Microsoft for building, testing, deploying, and managing applications and services through a global network of Microsoft-managed data centers. The following sections explain how to create a virtual FortiProxy instance in Microsoft Azure.

Deploying the FortiProxy VM on the Microsoft Azure cloud platform

You can deploy the FortiProxy VM using a PowerShell script. For an example script, refer to Script on page 8. Customize the script so that it fits your needs.

You need a thorough knowledge of PowerShell and various Azure services and features to use this deployment method.

1. Download the VHD image file.

a. Download the FortiProxy image from https://support.fortinet.com. For example:
`FPX_AZURE-v100-buildXXXX-FORTINET.out.hyperv.zip`
`XXXX` is the build number.

b. Unzip it and locate the VHD image file. You will find the `fortios.vhd` file in the Virtual Hard Disks directory.

2. Create a virtual machine.

a. Install Azure PowerShell using PowerShellGet (available at https://docs.microsoft.com/en-us/powershell/azure/azurerm/install-azurerm-ps?view=azurermps-5.7.0).

b. Enter values for the following variables in the script:

- `$loginusername`—The user name for logging in to the Microsoft Azure cloud platform.
- `$loginPassword`—The password for logging in to the Microsoft Azure cloud platform.
- `$subscriptionid`—Your subscription ID for the Microsoft Azure cloud platform.
- `$location`—See https://azure.microsoft.com/en-us/global-infrastructure/regions/.
- `$resourcegroup`—The resource group name is lowercase and between 3 and 24 characters long. If the resource group does not exist, the script creates it.
- `$adminUsername`—The user name for logging in to your Microsoft Azure VM.
- `$adminPassword`—The password for logging in to your Microsoft Azure VM.
- `$sourceVhd`—The path to the `fortios.vhd` file that you downloaded in step 1.
- `$customdataFile`—The path to a text file containing your initial VM configuration. For example:

```
config system global
   set hostname fpx1
   set admin-time 120
end
```

- `$vmname`—The name of your new VM.
  NOTE: The VM name cannot exceed 64 characters in length or contain the following characters:
  `~!@#$%^&*()=+_[]{}\|;:'",<>/?

c. Save and run the script.

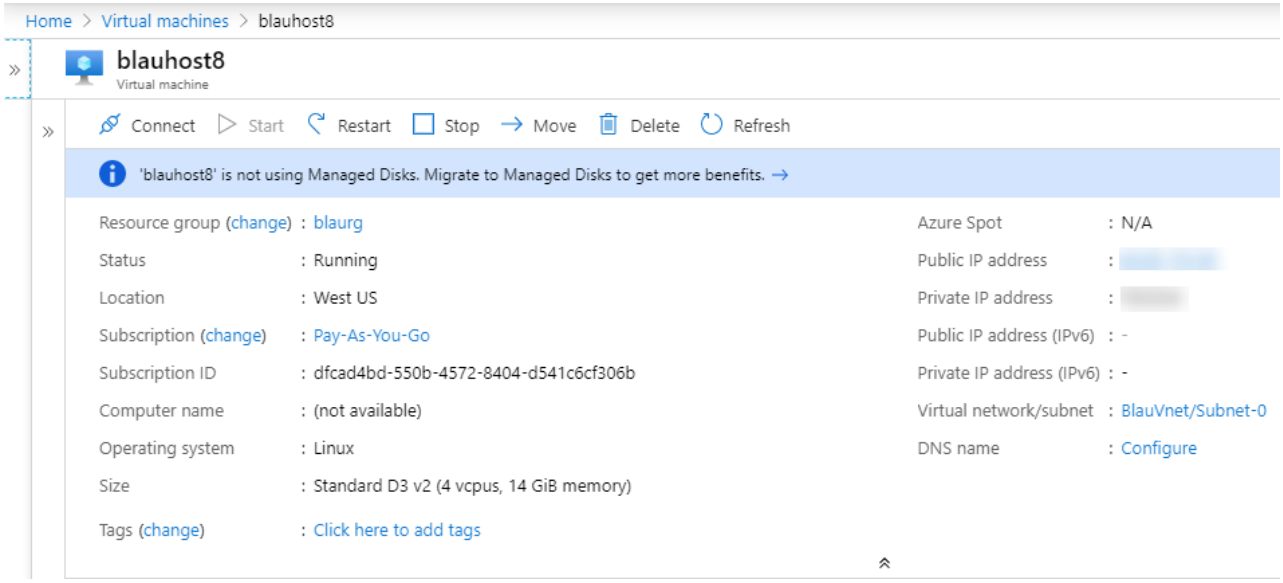
d. Enter `1`, `2`, `3`, or `4` for the number of network interfaces that you want the script to create.

e. Enter `y` to use the variable values that you defined in the script. If you enter `n`, the script prompts you for each value.
The script creates a resource group if one does not already exist. The resource group contains the virtual machine, storage account, one to four network interfaces, a public IP address, and a virtual network.

3. Access the FortiProxy VM.

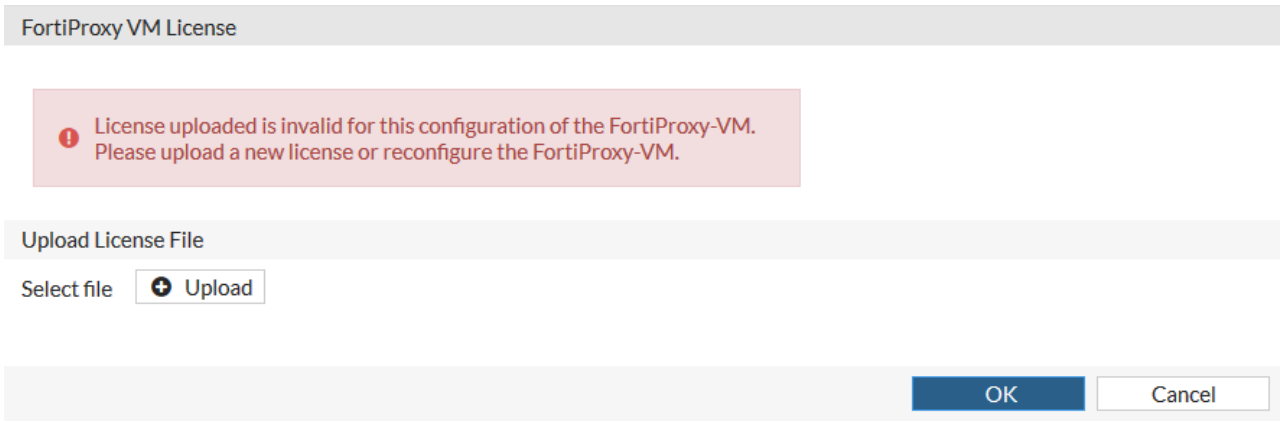
a. Go to *Home* > *Virtual machines*.

b. Select the name of your virtual machine to see the configuration details.



c. To access the FortiProxy VM, use the public IP address in a browser: `http://<public_IP_address>`. Use the user name and password that you specified when creating the VM. You can also use SSH (`ssh <user_name>@<public_IP_address>`) or HTTPS to access the public IP address of your FortiProxy VM.

d. Select *Upload* to upload your FortiProxy VM license file and then select *OK*.



e. After the FortiProxy VM reboots, you can log in.

Script

The following script uses Windows PowerShell 5.1 with AzureRM module 5.7 to deploy the FortiProxy VM on the Microsoft Azure cloud platform. The script creates a resource group, virtual machine, one to four network interfaces, a public IP address, and a virtual network. You can use an existing resource group if you prefer. You can change the script to fit your needs.

fpx_deploy_example.ps1:

```
Param(
       [Parameter(Mandatory)][int]$numberOfNics = 1,
       [int]$addSshKey = 0
)

# Make it interactive for the number of NIC
'parameters accepted,'
'[int]numberOfNics = number of nics of the instance (between 1 and 4)'
'[int]addSshKey = 1 for yes, 0 for no. If yes, your key must be provided in the
corresponding line.'
'[string]vmsize = examples: standard_d3_v2'
'`n'
echo "numberOfNics : $numberOfNics"
if ($numberOfNics -gt 4 -or $numberOfNics -lt 1) {
       'This script only supports NIC number from 1 to 4.'
}

################################# User-Defined Variables Begins ##################
# Azure account information
# Pre-define your account information:
$loginusername = "xx@xxxx" # "<your Azure user name>"
$loginPassword = "xxx" #"<your Azure password"
$SecureloginPassword = ConvertTo-SecureString $loginPassword -AsPlainText -Force
$subscriptionid = "xxx" #<your subcription ID>"

# Azure Resource Group Setting
>$location = "" # Azure location: for example, West Europe, North Europe, West US,
West US 2, East US,  East US 2, Japan West
$resourcegroup= ""      #your resource group name
$storageAccountName = $resourcegroup + "storage"  # define your storage account
name

# Azure VM Login Credentiald
$adminUsername = ""    #"your VM user name"
$adminPassword = ""    #"your VM password, length (12+)
$securePassword = ConvertTo-SecureString $adminPassword -AsPlainText -Force

# FortiProxy VHD File Location
```

```
$sourceVhd = "path\to\your\vhd\file" # your VHD file path"

# Azure Cloud Initial Configuration File
$customdataFile= "path\to\your\config\data\file"  #"your path to configuration
file"

# this part is for the VM variables
$vmsize = "standard_d3_v2" # for example, "standard_d3_v2"
# your FortiProxy VM name cannot exceed 64 characters in length or contain the
following characters: ` ~ ! @ # $ % ^ & * ( ) = + _ [ ] { } | ; : ' " , < > / ?.
$vmname = 'xxx'
#$keyData=''

##################### Optional user-defined variables ##############################
# Your VM storage profile
$containerName = "vhds"
$targetOsVhd = "FortiProxyOs.vhd"
$targetDataVhd = "FortiProxyData.vhd"

# Specify your own VNet environment variables
$vnetName = "YourVnet"  # Specify Your VNet Name
$vnetAddrPrefix = '10.0.0.0/16'  #Specify your VNet address prefix
$subnetName0 = 'Subnet-0'
$subnetName1 = 'Subnet-1'
$subnetName2 = 'Subnet-2'
$subnetName3 = 'Subnet-3'
$subnetAddrPrefix0 = '10.0.0.0/24'
$subnetAddrPrefix1 = '10.0.1.0/24'
$subnetAddrPrefix2 = '10.0.2.0/24'
$subnetAddrPrefix3 = '10.0.3.0/24'
$nicName0 = 'nic-eth0'
$nicName1 = 'nic-eth1'
$nicName2 = 'nic-eth2'
$nicName3 = 'nic-eth3'
$publicIpName = 'pip-0'
$addrPrefixList = $subnetAddrPrefix0, $subnetAddrPrefix1, $subnetAddrPrefix2,
$subnetAddrPrefix3
$addrPrefixList = $addrPrefixList[0..($numberOfNics-1)]

############################## User-defined variables end ############################


function getExistingVnet($name, $resourceGrp) {
  while ($vnet = Get-AzureRmVirtualNetwork -ResourceGroupName $resourceGrp -Name
$name -ErrorAction ignore) {
    Write-Host "Virtual Network $name exists already. The VNet's address space is:
"
       Write-Host $vnet.addressSpaceText
       if ((Read-Host 'Use the existing vnet (y/n)?') -eq 'y') {
```

```
          break
        } else {
          $name = Read-Host 'Specify your Virtual Network Name'
        }
    }
    $vnet = if ($vnet) {$vnet} else {$null}
    return $vnet, $name
}

function getExistingSubnet($subnetName, $addrPrefix, $addrPrefixList, $vnet,
$vnetAddrPrefix) {
    $subnet = $null
    for ($i=0; $i -lt $vnet.Subnets.Count; $i++) {
         if ($vnet.Subnets[$i].Name -eq $subnetName) {
           Write-Host "Subnet $subnetName exists already. Its address space is:  "
$vnet.Subnets[$i].AddressPrefix
           if ((Read-Host 'Use the existing subnet (y/n)?') -eq 'y') {
                 $subnet = $vnet.Subnets[$i]
                 break
                 } else {
                 do {
                         $subnetName = Read-Host 'Specify your subnet Name'
                         } while ($subnetName.length -eq 0)
                 $i = -1
                 continue
           }
        }
        if ($vnet.Subnets[$i].AddressPrefix -eq $addrPrefix) {
          Write-Host "Address Prefix $addrPrefix has been allocated to subnet named:  "
$vnet.Subnets[$i].Name
          if ((Read-Host 'Use the existing subnet (y/n)?') -eq 'y') {
                 $subnet = $vnet.Subnets[$i]
                 $subnetName = $vnet.Subnets[$i].Name
                 break
                 } else {
                 do {
                   $addrPrefix = Read-Host "Duplicated address $addrPrefix, Specify
your address prefix within addr space of $vnetAddrPrefix"
                 } while ($addrPrefixList.IndexOf($addrPrefix) -ge 0)
                 $addrPrefixList[$i] = $addrPrefix
                 $i = -1
                 continue
           }
        }
    }
    return $subnet, $subnetName, $addrPrefix
}

function checkNicName($resourcegrp, $name) {
```

```
        $nic = Get-AzureRmNetworkInterface -ResourceGroupName $resourcegrp -Name $name -
ErrorAction Ignore
        while ($nic) {
                Write-Host "NIC: $name exists already."
                if (-not $nic.VirtualMachine -and (Read-Host "It has not been assigned
to any VM, Re-use $name? (y/n)") -eq 'y') {
                        Remove-AzureRmNetworkInterface -ResourceGroupName
$resourcegroup -Name $name -Force
                        break
                } else {
                        do {
                                $name = Read-Host "NIC: specify another name"
                        } while ($name.Length -eq 0)
                        $nic = Get-AzureRmNetworkInterface -ResourceGroupName
$resourcegrp -Name $name -ErrorAction Ignore
                }
        }
        return $name
}

function checkExistingPublicIp($name, $resGrp) {
        while ($pip = Get-AzureRmPublicIpAddress -ResourceGroupName $resGrp -Name
$name -ErrorAction Ignore) {
                if ($pip.IpConfiguration.id) {
                        do {
                                $name = Read-Host "Public IP Name: $name has been
assigned already. Specify another Name"
                        } while ($name.length -eq 0)
                } else {
                        break
                }
        }
        $pip = if ($pip) {$pip} else {$null}
        return $pip, $name
}

function getNicName($resourcegrp, $name) {
        $nic = Get-AzureRmNetworkInterface -ResourceGroupName $resourcegrp -Name
$name -ErrorAction Ignore
        while ($nic) {
                if ($nic.VirtualMachine) {
                        $nicVm = $nic.VirtualMachine.Id.Split('/')[-1]
                        Write-Host "NIC: $name has been assigned to VM $nicVm already."
                } elseif ((Read-Host "NIC $name exists and has not been assigned to any
VM, Re-use $name? (y/n)") -eq 'y') {
                        break
                }
                do {
                        $name = Read-Host "NIC: specify another name"
```

```
                } while ($name.Length -eq 0)
                $nic = Get-AzureRmNetworkInterface -ResourceGroupName $resourcegrp
-Name $name -ErrorAction Ignore
        }
        $nic = if($nic) {$nic} else {$null}
        return $nic, $name
}

function getExistingPublicIp($name, $resGrp, $nic) {
        if ($nic -and $nic.IpConfigurations) {
                if ($nic.IpConfigurations[0].PublicIpAddress) {
                        $name = $nic.IpConfigurations[0].PublicIpAddress.Id.Split('/')
[-1]
                        $pip = Get-AzureRmPublicIpAddress -ResourceGroupName $resGrp
-Name $name
                }
        } else {
                while ($pip = Get-AzureRmPublicIpAddress -ResourceGroupName $resGrp
-Name $name -ErrorAction Ignore) {
                        if ($pip.IpConfiguration.id) {
                                do {
                                        $name = Read-Host "Public IP Name: $name has
been assigned already. Specify another Name"
                                } while ($name.length -eq 0)
                        } else {
                                break
                        }
                }
        }
        $pip = if ($pip) {$pip} else {$null}
        return $pip, $name
}

function checkExistingVm($name, $resGrp) {
        while ($vm = Get-AzureRmVm -ResourceGroupName $resGrp -Name $name -ErrorAction
Ignore) {
                do {
                        $name = Read-Host "VM name: $name exists already, specify
another name"
                } while ($name.Length -eq 0)
        }
        return $name
}

function getExistingVm($name, $resGrp, $context, $containerName) {
                $vm = Get-AzureRmVm -ResourceGroupName $resGrp -Name $name -ErrorAction
Ignore
                $blobs = $null
                $blobs = Get-AzureStorageBlob -Context $context -Container
$containerName -ErrorAction Ignore
```

```
                while ($vm) {
                        if((Read-Host "VM machine name $name exists already. Overwrite
the existing vm (y/n)") -eq 'y') {
                                Remove-AzureRmvm -ResourceGroupName $resGrp -Name $name
-Force
                                delVmDisks $vm $containerName $context $blobs
                                break
                        } else {
                                do {
                                        $name = Read-Host "VM name: $name exists already,
specify another name"
                                } while ($name.Length -eq 0)
                                $vm = Get-AzureRmVm -ResourceGroupName $resGrp -Name
$name
-ErrorAction Ignore
                        }
                }
                return $name
}

function getExistStorageAccount($name, $resGrpObj, $location) {
        $storageAccount = Get-AzureRmStorageAccount -ResourceGroupName
$resGrpObj.ResourceGroupName -name $name -ErrorAction Ignore
        while ($storageAccount) {
                if ($resGrpObj.location -ne $storageAccount.Location) {
                        do {
                                $name = Read-Host "Storage account name: $name exists
in a location other than $location, specify another name."
                        } while ($name.Length -eq 0)
                        $storageAccount = Get-AzureRmStorageAccount -ResourceGroupName
$resGrpObj.ResourceGroupName -name $name -ErrorAction Ignore
                } else {
                        break
                }
        }
        if (-not $storageAccount) {$storageAccount = $null}
        return $storageAccount, $name
}

function delVmDisks($vm, $containerName, $context, $blobs) {
        if ($blobs) {
                delBlobOfDisk $vm.StorageProfile.OsDisk.Name, $blobs $containerName
$context
                foreach ($dataDisk in $vm.StorageProfile.DataDisks) {
                        delBlobOfDisk $dataDisk.Name $blobs $containerName $context
                }
        }
}
```

```
function delBlobOfDisk($diskName, $blobs, $containerName, $context) {
       if ($diskName) {
              foreach ($blob in $blobs) {
                     if ($blob.ICloudBlob.Metadata['MicrosoftAzureCompute_DiskName']
-eq $diskName) {
                                   Remove-AzureStorageBlob -Container $containerName
-context $context -Blob $blob.Name -Force
                                   break
                     }
              }
       }
}


if (-Not ((Read-Host 'Specify whether to use the pre-defined values in script
(y/n)') -eq 'y')) {
       $loginusername = if($inputVal = Read-Host "Your Azure user (enter to use
predefined value - $loginusername)") {$inputVal} else {$loginusername}
       $SecureloginPassword = if(($inputVal = Read-Host "Your Azure password (enter to
use predefined value)" -AsSecureString).Length) {$inputVal} else
{$SecureloginPassword}
       $subscriptionid = if($inputVal = Read-Host "Your subcription id (enter to use
predefined value - $subscriptionid)") {$inputVal} else {$subscriptionid }
       $location = if($inputVal = Read-Host "Azure location (enter to use predefined
value - $location)") {$inputVal} else {$location }
       $resourcegroup = if($inputVal = Read-Host "Your storage account name (enter to
use predefined value - $resourcegroup)") {$inputVal} else {$resourcegroup}
       $storageAccountName= if($inputVal = Read-Host "Your resource group name (enter
to use predefined value - $storageAccountName)") {$inputVal} else
{$storageAccountName}
       $adminUsername = if($inputVal = Read-Host "Your vm username (enter to use
predefined value - $adminUsername)") {$inputVal} else {$adminUsername }
       $securePassword = if(($inputVal = Read-Host "Your vm password (enter to use
predefined value)" -AsSecureString).Length) {$inputVal} else {$securePassword}
       $sourceVhd = if($inputVal = Read-Host "local path of the VHD file (enter to use
predefined value - $sourceVhd)") {$inputVal} else {$sourceVhd }
       $customdataFile= if($inputVal = Read-Host "path of custom data file (enter to
use predefined value - $customdataFile)") {$inputVal} else {$customdataFile}
       $vmsize = if($inputVal = Read-Host "Specify the VM size (enter to use predefined
value - $vmsize)") {$inputVal} else {$vmsize}
       $vmname = if($inputVal = Read-Host "Specify the name of VM  (enter to use
predefined value - $vmname)") {$inputVal} else {$vmsize}
}

$logincred = New-Object System.Management.Automation.PSCredential ($loginusername,
$SecureloginPassword)
Login-AzureRmAccount -Credential $logincred
```

```
Select-AzureRmSubscription -Subscription $subscriptionid
$resourceGrpObj = Get-AzureRmResourceGroup -Name $resourcegroup -Location $location
-ErrorAction SilentlyContinue -ErrorVariable resGrpNotFound
if ($resGrpNotFound) {
       New-AzureRmResourceGroup -Name $resourcegroup -Location $location
       'New resource group ' + $resourcegroup + ' is created.'
} Else {
       'Existing resource group ' + $resourcegroup + ' is found.'
       $storageAccount, $storageAccountName = getExistStorageAccount
$storageAccountName $resourceGrpObj $location
       $ctx = if ($storageAccount) {$storageAccount.Context } else { $null}
       $container = if ($ctx) {Get-AzureStorageContainer -Name $containerName -Context
$ctx -ErrorAction Ignore} else {$null}
       $vmname = checkExistingVm $vmname $resourcegroup
       #$vmname = getExistingVm $vmname $resourcegroup $ctx $containerName

       $vnet, $vnetName = getExistingVnet $vnetName $resourcegroup
       $subnet0, $subnetName0, $subnetAddrPrefix0 = getExistingSubnet $subnetName0
$subnetAddrPrefix0 $addrPrefixList $vnet $vnetAddrPrefix
       $nicName0 = checkNicName $resourcegroup $nicName0
       $publicIp, $publicIpName = checkExistingPublicIp $publicIpName $resourcegroup
$nic0

       if ($numberOfNics -gt 1) {
              $subnet1, $subnetName1, $subnetAddrPrefix1 = getExistingSubnet
$subnetName1 $subnetAddrPrefix1 $addrPrefixList $vnet $vnetAddrPrefix
              $nicName1 = checkNicName $resourcegroup $nicName1
       }
       if ($numberOfNics -gt 2) {
              $subnet2, $subnetName2, $subnetAddrPrefix2 = getExistingSubnet
$subnetName2 $subnetAddrPrefix2 $addrPrefixList $vnet $vnetAddrPrefix
              $nicName2 = checkNicName $resourcegroup $nicName2
       }
       if ($numberOfNics -gt 3) {
              $subnet3, $subnetName3, $subnetAddrPrefix3 = getExistingSubnet
$subnetName3 $subnetAddrPrefix3 $addrPrefixList $vnet $vnetAddrPrefix
              $nicName3 = checkNicName $resourcegroup $nicName3
       }
}

Write-Output "$(Get-Date -f $timeStampFormat) - "

# this part is for providing the customData
$customdatacontent = Get-Content $customdataFile -Raw

# Getting the subscription and storage account information
if (!$storageAccount) {
              $storageAccount = New-AzureRmStorageAccount -ResourceGroupName
$resourcegroup -Name $storageAccountName -Location $location -SkuName Standard_LRS
```

```
-Kind Storage -EnableEncryptionService Blob
}
if(-not $storageAccount) {
       throw "Unable to find storage account '$storageAccountName'. Cannot continue."
}
if (!$ctx) { $ctx = $storageAccount.Context}
#$storageAccountName = $storageAccount.StorageAccountName
Write-Output "$(Get-Date -f $timeStampFormat) - StorageAccountName is :
$storageAccountName"

if(-Not $container) {
       New-AzureStorageContainer -Name $containerName -Context $ctx -Permission blob
}

$destinationVhd =
"https://$storageAccountName.blob.core.windows.net/vhds/fortios.vhd"
$sourceImageUri =
"https://$storageAccountName.blob.core.windows.net/vhds/fortios.vhd"
while (Get-AzureStorageBlob -Context $ctx -Blob $targetOsVhd -Container
$containerName -ErrorAction Ignore) {
       do {
                $targetOsVhd = Read-Host "Target VM OS VHD $targetOsVhd exists already,
specify an other name"
       } while($targetOsVhd.Length -eq 0)
}
while (Get-AzureStorageBlob -Context $ctx -Blob $targetDataVhd -Container
$containerName -ErrorAction Ignore) {
       do {
                $targetDataVhd = Read-Host "Target VM Data VHD $targetDataVhd exists
already, specify an other name"
       } while($targetDataVhd.Length -eq 0)
}
$vhdUri = "https://$storageAccountName.blob.core.windows.net/vhds/$targetOsVhd"
Add-AzureRmVhd -LocalFilePath $sourceVhd -Destination $destinationVhd -
ResourceGroupName $resourcegroup -OverWrite -NumberOfUploaderThreads 5
$dataDiskUri =
"https://$storageAccountName.blob.core.windows.net/vhds/$targetDataVhd"


# some reserved script variables
$resourceGroupName = $storageAccount.ResourceGroupName

Write-Output "$(Get-Date -f $timeStampFormat) - Resource Group Name is :
$resourceGroupName"


Write-Output "$(Get-Date -f $timeStampFormat) - Creating Virtual Network"
if (!$publicIp) {
       $publicIp = New-AzureRmPublicIpAddress -Name $publicIpName -ResourceGroupName
```

```
$ResourceGroupName -Location $location -AllocationMethod Dynamic -force
}

if (!$vnet) {
	$vnet = New-AzureRmVirtualNetwork -ResourceGroupName $resourceGroupName
-Location $location -Name $vnetName -AddressPrefix $vnetAddrPrefix
}

Write-Output "$(Get-Date -f $timeStampFormat) - Adding Subnets to Virtual Network"

if (!$subnet0) {
	$vnet = $vnet | Add-AzureRmVirtualNetworkSubnetConfig -Name $subnetName0 -
AddressPrefix $subnetAddrPrefix0 | Set-AzureRmVirtualNetwork
}

$nic0 = New-AzureRmNetworkInterface -ResourceGroupName $resourceGroupName -Location
$location -Name $nicName0 -SubnetId $vnet.Subnets[0].id -PublicIpAddressId
$publicIp.Id -force
if ($numberOfNics -gt 1) {
	if (!$subnet1) {
		$vnet = $vnet | Add-AzureRmVirtualNetworkSubnetConfig -Name $subnetName1
-AddressPrefix $subnetAddrPrefix1 | Set-AzureRmVirtualNetwork
	}
	$nic1 = New-AzureRmNetworkInterface -ResourceGroupName $resourceGroupName -
Location $location -Name $nicName1 -SubnetId $vnet.Subnets[1].id -force
} elseif ($numberOfNics -gt 2) {
	if (!$subnet2) {
		$vnet = $vnet | Add-AzureRmVirtualNetworkSubnetConfig -Name $subnetName2
-AddressPrefix $subnetAddrPrefix2 | Set-AzureRmVirtualNetwork
	}
	$nic2 = New-AzureRmNetworkInterface -ResourceGroupName $resourceGroupName -
Location $location -Name $nicName2 -SubnetId $vnet.Subnets[2].id -force
} elseif ($numberOfNics -gt 3) {
	if (!$subnet3) {
		$vnet = $vnet | Add-AzureRmVirtualNetworkSubnetConfig -Name $subnetName3
-AddressPrefix $subnetAddrPrefix3 | Set-AzureRmVirtualNetwork
	}
	$nic3 = New-AzureRmNetworkInterface -ResourceGroupName $resourceGroupName -
Location $location -Name $nicName3 -SubnetId $vnet.Subnets[3].id -force
}


Write-Output "$(Get-Date -f $timeStampFormat) - Configuring VM"
$vmConf = New-AzureRmVMConfig -VMName $vmname -VMSize $vmsize

$cred = New-Object System.Management.Automation.PSCredential ($adminUsername,
$securePassword)
$vmConf = Set-AzureRmVMOperatingSystem -Linux -VM $vmConf -ComputerName $vmName -
Credential $cred -CustomData $customdatacontent
```

```
#$vmConf = Set-AzureRmVMOperatingSystem -Linux -VM $vmConf -ComputerName $vmName -
Credential $cred

$vmConf = Add-AzureRmVMNetworkInterface -VM $vmConf -Id $nic0.Id -Primary
if ($numberOfNics -gt 1) {
       $vmConf = Add-AzureRmVMNetworkInterface -VM $vmConf -Id $nic1.Id
} elseif ($numberOfNics -gt 2) {
       $vmConf = Add-AzureRmVMNetworkInterface -VM $vmConf -Id $nic2.Id
} elseif ($numberOfNics -gt 3) {
       $vmConf = Add-AzureRmVMNetworkInterface -VM $vmConf -Id $nic3.Id
}


$vmDiskName = $vmname + '_OsDisk' #Specify your VM Disk Name
Write-Output "$(Get-Date -f $timeStampFormat) - Config the OS Disk name
$vmDiskName..."
$vmConf = Set-AzureRmVMOSDisk -VM $vmConf -Name $vmDiskName -SourceImageUri
$destinationVhd -VhdUri $vhdUri -CreateOption fromImage -Linux


Write-Output "$(Get-Date -f $timeStampFormat) - Creating FPX VM: $vmname"
$result = New-AzureRmVM -ResourceGroupName $resourceGroupName -Location $location -
VM $vmConf

Write-Output "$(Get-Date -f $timeStampFormat) - Finish Running Script"
```