Import Windows Server to Amazon EC2 with PowerShell

February 2017

This paper has been archived.

For the latest technical content about this subject, see the AWS Whitepapers & Guides page:
http://aws.amazon.com/whitepapers

Notices

This document is provided for informational purposes only. It represents AWS's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS's products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Contents

Abstract

This whitepaper is for Microsoft Windows IT professionals who want to learn how to use Amazon Web Services (AWS) VM Import/Export to import custom Windows Server images into Amazon Elastic Compute Cloud (Amazon EC2). PowerShell code is provided to demonstrate one way you could automate the task of importing images and launching instances, but there are many other DevOps automation techniques that could come into play in a well thought-out cloud migration process.

Introduction

Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. Amazon EC2 reduces the time required to obtain and boot new server instances. It changes the economics of computing by allowing you to pay only for capacity that you actually use.

You have full administrator access to each EC2 instance, and you can interact with your instances just as you do with your on-premises servers. You can stop your instance and retain the data on your boot partition, then restart the same instance using PowerShell or a browser interface.

Amazon EC2 Dedicated Instances

Dedicated Instances are Amazon EC2 instances that run in a virtual private cloud (VPC) on hardware that's dedicated to a single customer. Your Dedicated Instances are physically isolated at the host hardware level from instances that belong to other AWS accounts. However, Dedicated Instances may share hardware with other instances from the same AWS account that are not Dedicated Instances. Dedicated Instances allow you to bring your own licenses for Windows Server. For more information, see http://aws.amazon.com/dedicated-instances.

Amazon EC2 Dedicated Hosts

An Amazon EC2 Dedicated Host is a physical server with Amazon EC2 instance capacity fully dedicated to your use. Dedicated Hosts can help you address compliance requirements and reduce costs by allowing you to use your existing server-bound software licenses.

Dedicated Hosts allow you to allocate a physical server and then launch one or more Amazon EC2 instances of a given type on it. You can target and reuse specific physical servers and be within the terms of your existing software licenses.

In addition to allowing you to Bring Your Own License (BYOL) to the cloud to reduce costs, Amazon EC2 Dedicated Hosts can help you to meet stringent

compliance and regulatory requirements, some of which require control and visibility over instance placement at the physical host level. In these environments, detailed auditing of changes is also crucial. You can use the AWS Config service to record all changes to your Dedicated Hosts and instances.

Dedicated Hosts allow you to use your existing per-socket, per-core, or per-virtual machine (VM) software licenses, including Microsoft Windows Server and Microsoft SQL Server. Learn more at https://aws.amazon.com/ec2/dedicated-hosts/.

AWS Server Migration Service

AWS Server Migration Service (AWS SMS) is an agentless service that makes it easier and faster for you to migrate thousands of on-premises workloads to AWS. AWS SMS allows you to automate, schedule, and track incremental replications of live server volumes, making it easier for you to coordinate large-scale server migrations.

Each server volume replicated is saved as a new Amazon Machine Image (AMI), which can be launched as an EC2 instance in the AWS Cloud. AWS SMS currently supports VMware virtual machines, and support for other physical servers and hypervisors is coming soon.

AWS SMS supports migrating Windows Server 2003, 2008, 2012, and 2016, and Windows 7, 8, and 10.

VM Import/Export

VM Import/Export enables you to easily import virtual machine images from your existing environment to Amazon EC2 instances and export them back to your on-premises environment. This allows you to use your existing virtual machines that you have built to meet your IT security, configuration management, and compliance requirements by bringing those virtual machines into Amazon EC2 as ready-to-use instances. VM Import/Export is available at no additional charge beyond standard usage charges for Amazon EC2 and Amazon Simple Storage Service (Amazon S3).

You can use PowerShell to import a Hyper-V or VMware image. VM Import will convert your virtual machine (VM) into an Amazon EC2 AMI, which you can use to run Amazon EC2 instances.

AWS Tools for Windows PowerShell

The AWS Tools for Windows PowerShell are a set of PowerShell cmdlets that are built on top of the functionality exposed by the AWS SDK for .NET. AWS Tools for Windows PowerShell enable you to script operations on your AWS resources from the PowerShell command line. Although the cmdlets are implemented using the service clients and methods from the SDK, the cmdlets provide an idiomatic PowerShell experience for specifying parameters and handling results. For example, the cmdlets for Tools for Windows PowerShell support PowerShell pipelining—that is, you can pipeline PowerShell objects both into and out of the cmdlets. Learn more at https://aws.amazon.com/documentation/powershell/.

AWS Config

AWS Config is a fully managed service that provides you an inventory of your AWS resources, as well as configuration history, and configuration change notifications to enable security and governance. Config Rules enable you to automatically check the configuration of your AWS resources. You can discover existing and deleted AWS resources, determine your overall compliance against rules, and dive into configuration details of a resource at any point in time. These capabilities enable compliance auditing, security analysis, resource change tracking, and troubleshooting. This enables you to manage your Windows Server licenses on Dedicated Hosts as required by Microsoft.

Licensing Considerations

Organizations that own Microsoft software licenses and Software Assurance have the option of bringing their own licenses (BYOL) to the cloud under the terms of Microsoft's License Mobility program (included with Software Assurance). In many cases, software license costs can dominate the cost of the computing, storage, and networking infrastructure in the cloud, so BYOL can be very beneficial. However, you must evaluate BYOL carefully.

For Windows Server and SQL Server, AWS also offers License Included (LI) as an option. It's called License Included because the software is pre-installed in the AMI and the complete software licenses are included when you launch an Amazon EC2 instance with those AMIs, even Client Access Licenses (CALs). You pay as you go for the Windows Server and SQL Server licenses, either hourly while the instance is running or with a 1- or 3-year Reserved Instance. Reserved Instances offer substantial discounts.

The LI model is convenient and flexible, but if you move a licensed on-premises workload to the cloud with LI instances then you would essentially be paying for dual software licenses. Even though that sounds expensive, it still might make sense to do in some cases, particularly if you plan to consolidate some of your workloads, or re-platform some application servers, or discontinue purchasing Software Assurance. So you need to consider your options, including BYOL, carefully.

However, don't assume that BYOL is always more economical. It's advisable to create a simple spreadsheet to make a balanced comparison of BYOL vs. LI. With BYOL, if you haven't bought the licenses yet, you need to know your Microsoft reseller bulk license discount. You also need to include the cost of Software Assurance (even if it's already a sunk cost, consider whether you plan to renew it), and the cost of EC2 Dedicated Hosts and Instances. Don't forget to include the correct number of licenses for all the cores on the instances you plan to use for Windows Server and SQL Server. With LI, you need to consider whether you are purchasing Reserved Instances, which offer substantial discounts.

> Tip: When using the AWS Simple Monthly Calculator to determine instance costs without licenses, select Amazon Linux even though you'll be importing your own Windows Server image. This avoids the license cost that the calculator automatically assumes for Windows Server.

Also, there are considerable advantages with LI:

- The licenses are fully managed by AWS, so you don't need to worry about auditing.
- You can forego the cost of Software Assurance for those licenses.

- You don't need to buy CALs.
- Each LI for Windows Server includes two Remote Desktop CALs.
- LI reduces your costs if you decide to consolidate workloads later.
- LI reduce your costs when you stop the instances.
- LI reduces your costs if you don't need the full capacity of a Dedicated Host.
- You retain the freedom to re-platform your workload.

Preparing for the Walkthroughs

Overview

The remainder of this paper walks you through several activities with Windows PowerShell. You can adapt and reuse these code snippets in your own AWS account to automate the following tasks:

- Import a Windows Server virtual machine to Amazon EC2.
- Launch and terminate a Dedicated Instance using your custom AMI.
- Configure Microsoft Key Management Services (KMS) to apply user-supplied licensing.
- Allocate a Dedicated Host and launch an instance in the host using your custom AMI, and then terminate the instance and the Dedicated Host.

Important: If you choose to follow along with the remaining sections in this paper, you will be creating resources in your AWS account, which will incur billing charges.

Prerequisites

These walkthroughs assume that you have previously exported a Windows Server image (for example, from VMware as an Open Virtualization Archive or OVA file) and stored it in an Amazon S3 bucket in your account. VM Import/Export also supports Microsoft Hyper-V, but an OVA is referenced here as an example.

You'll need to have the AWS Tools for Windows PowerShell and grant security rights for PowerShell to access your AWS account. The easiest way to do that is to launch a Windows Server instance in Amazon EC2 with an AWS Identity and Access Management (IAM) role.

You'll also need an Amazon Virtual Private Cloud VPC, a subnet, a security group, and a key-pair in the Region where you import the image. You certainly can create those in PowerShell, but it's generally more reliable to create as much of your infrastructure as possible using AWS CloudFormation. The reason is that you need to consider how to roll back your stack in case any errors occur while building it. AWS CloudFormation provides a simple mechanism to automatically roll back so that you won't be left paying the bill for an incomplete stack after an error occurs. To roll back in PowerShell, you would need to trap potential errors at the point where each resource is created in your script and then write the code to remove or deallocate every other resource that the script had successfully created up to that point. That would get very tedious in regular PowerShell but could be more easily handled with PowerShell Desired State Configuration (DSC).

To comply with your Windows Server license terms and implement BYOL, you'll need to a have a Microsoft KMS instance running in your VPC. The walkthrough shows you how to configure the BYOL instance for Microsoft KMS, though you can proceed with this walkthrough without having Microsoft KMS running.

Finally, these walkthroughs assume that your own workstation is running Windows Server 2016, though these steps should work with other versions with minor modifications.

Walkthrough: Import Your Custom Image

1. On the **Windows Start** menu, choose **Windows PowerShell ISE**.
2. In the **Windows PowerShell ISE**, press **Ctrl+R** to show the Script Pane (or on the **View** menu, choose **Show Script Pane**).
3. The AWS Tools for PowerShell allow you to specify the AWS Region separately in most cmdlets, but it's simpler to set the default Region for your whole session. For example, run the following commands in PowerShell to set "us-west-2" as the default Region. You'll be using the

"lab_region" variable again later in this walkthrough, so make sure you set it here to your preferred Region.

```
$lab_region = "us-west-2"
Set-DefaultAWSRegion $lab_region
```

4. To use the VM import service role in your own AWS account, create an IAM policy document to grant access for the Amazon EC2 Import API (vmie.amazonaws.com). You must name the role "vmimport". (Note: you could create this policy in the AWS Management Console, but this example shows how to do it with a document in PowerShell.)

```
$importPolicyDocument=@"
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"",
         "Effect":"Allow",
         "Principal":{
            "Service":"vmie.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "StringEquals":{
               "sts:ExternalId":"vmimport"
            }
         }
      }
   ]
}
"@
New-IAMRole -RoleName vmimport -AssumeRolePolicyDocument
$importPolicyDocument
```

5. Associate a policy with the "vmimport" role so that VM Import/Export can access the VM image in your S3 bucket and create an AMI in Amazon EC2. If you'd like to create your own restrictive policy for security reasons, see this page for guidance: http://docs.aws.amazon.com/vm-import/latest/userguide/import-vm-image.html. AWS also provides a couple of managed (built-in) policies

that make it convenient to grant access to the VM import service role to Amazon S3 and Amazon EC2. (Note: This code consists of two commands that are wrapped to fit the document.)

```
Register-IAMRolePolicy -RoleName vmimport -PolicyArn
arn:aws:iam::aws:policy/AmazonS3FullAccess
Register-IAMRolePolicy -RoleName vmimport -PolicyArn
arn:aws:iam::aws:policy/AmazonEC2FullAccess
```

6. Create a userBucket object to define the location of your image file and an ImageDiskContainer parameter, both of which are passed to the Import-EC2Image cmdlet. However, before running these commands, replace <UniqueBucketName> with the name of the bucket where you stored the OVA file. If you are importing Hyper-V, change the Format property to “VHD”.

```
$userBucket = New-Object Amazon.EC2.Model.UserBucket
$userBucket.S3Bucket = "<UniqueBucketName>"
$userBucket.S3Key = $file
$windowsContainer = New-Object Amazon.EC2.Model.ImageDiskContainer
$windowsContainer.Format = "OVA"
$windowsContainer.UserBucket = $userBucket
```

7. Now create an object for the remaining parameters for the import task. Set the "Platform” parameter to match the imported operating system type. The “LicenseType” parameter controls how the instance that is imported is configured for licensing. Set it to BYOL.

```
$params=@{
    "ClientToken"="MyCustomWindows_" + (Get-Date)
    "Description"="My custom Windows image"
    "Platform"="Windows"
    "LicenseType"="BYOL"
}
```

8. Now you’re ready to start the import task. When you run this command, the import process will take about 45 minutes, but you can proceed with the remaining steps in this paper if you’re willing to temporarily use

other AMI IDs. This command is all one line, but wrapped here to fit the page.

```
Import-EC2Image -DiskContainer $windowsContainer @params –region
$lab_region
```

9. You can check the progress of the import task with the following command, which will show the Progress property and the Status property. The Progress property reports the current percentage complete status for the import task. The Status property indicates the migration phase.

```
Get-EC2ImportImageTask -region $lab_region
```

Walkthrough: Launch a Dedicated Instance

1. While waiting for your own image to be imported, you can follow the rest of the walkthroughs using an AWS AMI. All the steps will work the same regardless of the AMI, except that you'll need to provide a key-pair to access an AWS AMI. When you launch an instance from your own imported AMI, you don't need to provide a key-pair if you already have an Administrator password. The command below obtains the AMI ID of the latest version of the AWS AMI for Windows Server 2016 ("base" means without SQL Server). The my_ami variable will be used later, so make sure you set it here. If you run this step after your import process is complete, you can use that AMI ID instead.

```
$my_ami = (Get-EC2ImageByName "Windows_2016_Base").ImageId
```

2. Configure two variables for use when launching the instance. Setting the instance type to "dedicated" means that you want a Dedicated Instance. With the exception of the t2 instance type, most instance types can be used for Dedicated Instances.

```
$tenancy_type = "dedicated"
$instance_type = "m4.large"
```

3. This step configures variables to store the networking parameters you'll use when you launch a new instance. Enter the Classless Inter-Domain Routing (CIDR) address of a subnet you've created in your VPC where you want to launch the new instance. If you don't provide a private IP address during launch, one will be assigned automatically within the subnet. However, you may want to script it for various reasons.

   The New-Ec2Instance cmdlet will use this private_IP address, and you will log into the instance in the next walkthrough to configure Microsoft KMS. If your workstation is not an EC2 instance in a *public* subnet in the same VPC where you are launching this instance in a *private* subnet, then you will need to do one of the following: (a) launch the instance in a public subnet; (b) use Remote Desktop Protocol (RDP) to allow remote connections into another instance in its associated public subnet; or (c) set up a Remote Desktop Gateway in its public subnet (see Remote Desktop Gateway on the AWS Cloud: Quick Start Reference Deployment http://docs.aws.amazon.com/quickstart/latest/rd-gateway/welcome.html).

```
$private_IP = "10.50.3.10"
$Subnet = "10.50.3.0/24"
$SubnetObj = Get-EC2Subnet -Filter @{Name="cidr"; Values=$Subnet}
```

4. Configure a variable to store the security group parameter you will use when you launch the new instance. Later in this walkthrough, you'll login to the instance through Remote Desktop to set up KMS for BYOL, so make sure the security group allows inbound RDP access from the Internet.

```
$SecurityGroup = "MySecurityGroup"
$SGObj = Get-EC2SecurityGroup -Filter @{Name="tag-value";
Values=$SecurityGroup}
```

5. Create a variable for the key-pair name parameter you will use to decrypt the administrator password for the new instance. Don't include the file extension .PEM. If you are launching an imported image on which you know the administrator password, you don't need to provide a key-pair.

```
$key_pair = "<key-pair-name>"
```

6. Now you're ready to launch your Dedicated Instance. Many other optional parameters can be configured with this cmdlet to customize the instance. However, the following is the minimum you need to launch an instance with BYOL.

```
$my_instance = New-EC2Instance
    ` -ImageId $my_ami
    ` -Tenancy $tenancy_type
    ` -InstanceType $instance_type
    ` -SubnetId $SubnetObj.SubnetID
    ` -PrivateIpAddress $private_IP
    ` -securityGroupId $SGObj.GroupID
    ` -KeyName $key_pair
```

7. It's a good idea to create a name tag for the new instance. The last two lines are a single cmdlet, wrapped here to fit the page.

```
$Tag = New-Object amazon.EC2.Model.Tag
$Tag.Key = 'Name'
$Tag.Value = "Server2016-Imported"
New-ec2Tag -ResourceID
$my_instance.runninginstance[0].instanceID -Tag $Tag
```


Walkthrough: Configure Microsoft KMS for BYOL

To comply with Microsoft licensing requirements for EC2 Dedicated Instances using the BYOL model, you must either supply a Windows license key for the instance, or configure it to use Microsoft KMS on a server that you manage.

In this task you will configure the Dedicated Instance to use a manually specified Microsoft KMS. You will connect to the new instance using Windows Remote Desktop Connection. If you used an AWS AMI to launch this instance, you need to decrypt the password using the lab key-pair in order to connect. If

you launched this instance using your imported image, you already know the local administrator account and password.

1. Log in to the AWS Management Console and go to the EC2 Dashboard.
2. Select only the instance you just launched with PowerShell.
3. Choose **Connect**.
4. In the **Connect To Your Instance** dialog box, choose **Get Password**. You might need to retry this a couple of times to give the instance a few minutes to initialize.
5. For **Key Pair Path**, choose **Choose File** (the button is named **Browse** in some browsers).
6. Browse to the .pem file on your local machine for the key-pair you specified when launching the instance, and choose **Open**.
7. Choose **Decrypt Password**.
8. Copy the decrypted password to your clipboard buffer.
9. Run **Remote Desktop Connection**.
10. In the **Computer** box enter the IP address of the Dedicated Instance you launched and choose **Connect**.
11. When prompted for credentials, log in as Administrator and paste the decrypted password from your clipboard buffer.
12. On the Remote Desktop Connection warning dialog box, choose **Yes** to ignore the verification warning.
13. In the Remote Desktop Connection session for the Server2016-Imported instance, when the desktop appears, choose **No** in the **Networks** dialog box to disable discovery (this is a Windows Server 2016 feature that is not available in earlier versions).
14. In the Remote Desktop Connection session for the Server2016-Imported instance, launch Windows PowerShell and run the following command to display the current configuration settings of the Microsoft KMS client.

```
slmgr.vbs /dlv
```

15. Enter the following commands to update the active Microsoft KMS configuration and confirm the change. Replace the IP address with a functioning KMS server that you have installed in your VPC. This command won't immediately fail if you don't have a running KMS instance at the given IP address.

```
slmgr.vbs /skms 10.50.3.100
slmgr.vbs /dlv
```

16. Close the Remote Desktop Connection to the Dedicated Instance and return to your workstation instance where you launched the instance. Terminate the Dedicated Instance. This cmdlet should be entered as a single line.

```
Remove-EC2Instance -InstanceId
$my_instance.Instances[0].InstanceId -Force
```

Walkthrough: Allocate a Dedicated Host and Launch an Instance

In this task you will launch and terminate an instance in a Dedicated Host.

1. Create variables for the Availability Zone and quantity parameters. Edit the $AZ variable appropriately before running this command.

```
$AZ = 'us-west-2a'
$Qty = 1
$AutoPlace = 'On'
```

2. Request a Dedicated Host. This reuses the $instance_type variable you created earlier, which was m4.large. Note that Dedicated Hosts are not available for all instance types.

```
new-EC2hosts
    ` -InstanceType $instance_type
    ` -AvailabilityZone $AZ
    ` -quantity $Qty
```

```
    ` -AutoPlacement $AutoPlace
```

3. Query the properties of your Dedicated Host. This command may initially return no data. Wait a moment and retry it. This command returns the number of physical CPU cores and sockets, the total number of virtual CPUs, and the type of instance supported on your Dedicated Host.

```
(get-EC2hosts).HostProperties
```

4. List the instances running on your Dedicated Host. This shows that initially there are no instances running in the host.

```
(get-EC2hosts).Instances
```

5. Specify the tenancy type "host" to launch an instance inside the Dedicated Host.

```
$tenancy_type = "host"
```

6. Indicate the AMI ID to be deployed in the Dedicated Host. There are Microsoft licensing restrictions for Dedicated Hosts. AWS and AWS Marketplace AMIs for Windows cannot be used. Ordinarily, you would specify the AMI ID of your imported image here. However, if the import task you started earlier is still running in the background, that AMI is not available yet. In order to demonstrate how to deploy instances to a Dedicated Host you can use an Amazon Linux AMI as a placeholder for the next few tasks.

```
$my_ami = (Get-EC2Image -Filters @{Name = "name"; Values =
"Amazon_CentOS*"}).ImageID
```

7. Launch the instance inside the Dedicated Host. Once again, the only difference is the requirement to provide a key-pair when launching an AWS AMI.

```
$host_instance = New-EC2Instance
    ` -ImageId $my_ami
    ` -Tenancy $tenancy_type
    ` -InstanceType $instance_type
    ` -SubnetId $SubnetObj.SubnetID
    ` -PrivateIpAddress $private_IP
    ` -securityGroupId $SGObj.GroupID
    ` -KeyName $key_pair
```

8. Create a name tag for the new instance. The last two lines are a single cmdlet.

```
$Tag = New-Object amazon.EC2.Model.Tag
$Tag.Key = 'Name'
$Tag.value = "DedicatedHost-Instance"
New-ec2Tag -ResourceID
$host_instance.runninginstance[0].instanceID -Tag $Tag
```

9. List the instances running on your Dedicated Host.

```
(get-EC2hosts).Instances
```

10. You must terminate all instances on a Dedicated Host before you can release it.

```
Remove-EC2Instance –InstanceId
    ` $host_instance.Instances[0].InstanceId -Force
```

11. Finally, release the Dedicated Host. The command below reports successful and unsuccessful attempts to release hosts. It doesn't report success until all running instances have been terminated. Repeat this command until your host-id is listed in the Successful column.

```
$dedicated_host = get-EC2hosts | Select-Object -first 1
Remove-EC2Hosts -HostId $dedicated_host.HostId –Force
```

12. Switch back to the EC2 Dashboard in your browser. In the navigation pane, choose **Dedicated Hosts** to confirm that DedicatedHost-Instance has been terminated. You might need to refresh the console display.

Conclusion

This paper has demonstrated how to use Windows PowerShell and VM Import/Export to import a custom Windows Server image into Amazon EC2. You can adapt and reuse the PowerShell code snippets to automate the process in your own AWS account.

In addition to VM Import/Export, consider using the AWS Server Migration Service. It currently supports VMware vCenter, and support for additional image formats is coming soon.

Contributors

The following individuals and organizations contributed to this document:

- Scott Zimmerman, Solutions Architect, AWS

Further Reading

For additional information, please consult the following sources:

- Getting Started with Amazon EC2 Windows Instances http://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/EC2Win_GetStarted.html