# Forklifting to AWS: An Option for Migration to AWS



TriNimbus Cloud Management Solutions
www.trinimbus.com info@trinimbus.com

## Table of Contents

## Introduction

This article discusses some of the considerations when moving existing physical or virtual environments to AWS using a process colloquially known as ‘forklifting’.

Depending on one’s definition, ‘forklifting’ may mean different things: it can denote moving to AWS by means of imaging the computers’ disks, including disks with the operating system, uploading the images to AWS, and launching EC2 instances using these images. In this scenario neither applications nor the underlying operating system of the computers being moved is re-installed.

Alternatively, ‘forklifting’ can denote a process in which the EC2 instances are created from stock AWS AMIs that include pre-installed operating system, the applications are re-installed, and application data is moved or replicated as needed. However, no configuration changes are made to take advantage of the AWS infrastructure. Instead, the AWS deployment replicates the original setup one-to-one.

Before getting into the details of forklifting to AWS, note that a general recommendation by AWS is “don’t do it”. While certain scenarios of forklifting are officially supported, other scenarios are not supported and may or may not work. Furthermore, by forklifting one is missing a lot of flexibility and benefits provided by cloud environments.

## Migrating from VMware to AWS

Migrating VMware virtual machines that were initially created in VMware is one of the scenarios that is officially supported – but even in this case the AWS environment imposes certain constraints that have to be satisfied before attempting forklifting.

VMware machines that were created by virtualization of physical machines (a process known as P2V) may not forklift to AWS. This process often works (subject to constraints discussed below), but it is not officially supported by AWS and may not work.

## Using VMware vCenter

The simplest approach to migration of VMware virtual machines to AWS is using AWS Management Portal for vCenter (http://aws.amazon.com/ec2/vcenter-portal/). This installs as a vCenter plugin and enables management of AWS resources in vCenter, including migration of VMware machines to AWS.

An alternative approach when using VMware vCenter is to export the virtual machines in OVF format and then import them using AWS VM Import Tools as described below (see also http://pubs.vmware.com/vsphere-4-esx-vcenter/index.jsp?topic=/com.vmware.vsphere.vmadmin.doc_41/vc_client_help/importing_and_exporting_virtual_appliances/t_export_a_virtual_machine.html).

## AWS VM Import Tools

AWS provides tools for importing VMDK, VHD, and RAW files and converting them to EC2 instances (http://aws.amazon.com/ec2/vm-import/).

When using this tool one has to pay extra attention to the underlying type of the VMDK files. The complication is that the extension 'VMDK' is used for a number of different file types (see http://www.forensicswiki.org/wiki/VMWare_Virtual_Disk_Format_(VMDK) for more info). While all of these file types represent content of virtual disks, they do so in various forms, and only one of them is accepted by the AWS import tools.

VMDK files used to run VMware virtual machines (e.g. by VMware Player) are in a format that is not compatible with AWS VM Import tools. These VMDK files need to be converted to a format recognized by AWS VM Import tools manually, before uploading to AWS. There are a number of conversion tools that can be used for this purpose:

- VMware OVFTool installed together with VMware Player and other VMware products
- qemu-img or VBoxManage as mentioned on http://spin.atomicobject.com/2013/06/03/ovf-virtual-machine/

## Third Party Conversion and Migration Tools

There are a number of third party vendors offering tools for converting and migrating virtual machines between different environments. Here are a few examples:

- Racemi (http://www.racemi.com/aws)
- DoubleTake (http://www.visionsolutions.com/products/dt-move.aspx)
- RiverMeadow (http://www.rivermeadow.com)
- NetIQ's PlateSpin (http://www.platespin.com)

## Prerequisites

Regardless of how a virtual machine is migrated to AWS, there are certain prerequisites that have to be fulfilled for the imported machine to work properly. AWS documentation related to this subject includes:

- http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/VMImportPrerequisites.html
- http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/PreparingYourVirtualMachine.html
- http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/VMImportTroubleshooting.html

In our experience, out of all considerations described in the above AWS documentation, the following items are most likely to apply:

- Make sure all network adapters are set to DHCP.
- Make sure .NET Framework 3.5 is installed.
- Disable firewalls, anti-virus programs, and other means of hardening your system.
- Disks other than primary one have to be dismounted and imported separately (same comments on VMDK/OVF conversion as above). Some of the above-mentioned commercial tools can work around this limitation and deal with multi-disk setups automatically.

## Possible Complications

Following is a list of the most complications one can encounter when forklifting existing environments to AWS.

## Microsoft ActiveDirectory Domain Controllers

Domain Controllers (DCs) are typically set-up with static IPs. This may be problematic to import to AWS. Also, the nature of domain controller software and domain membership is such that trying to launch a Windows image with pre-installed and pre-configured ActiveDirectory software is going to be problematic.

The easiest workaround is setting-up new DCs in AWS 'from scratch', joining them to the original domain over a VPN, replicating the ActiveDirectory databases, and then de-commissioning the original DCs.

## Clustering

Some clusters (e.g. file servers using MS Clustering Services or Linux DRBD) rely on 'floating (virtual) IP' pattern. Because of sandboxing and other security features of AWS this pattern does not easily translate to AWS. Implementing this pattern in AWS requires additional scripting – to make the operating system and clustering software AWS-aware.

Other clusters (e.g. MS SQL Server) can be implemented in AWS without major complications. See e.g. http://aws.amazon.com/whitepapers/microsoft-wsfc-sql-alwayson/.

## Shared Storage

Solutions based on shared storage may be hard to forklift as there is no direct equivalent of ‘shared storage’ in AWS.

Even if one can get such setup working 'in principle' (e.g. setting up dedicated EC2 instances as iSCSI targets, or using AWS Storage Gateway), the performance may suffer because of the additional level of indirection for disk access.

## Large Storage Devices

Very large storage devices (more than 1TB as of September 2014) are not directly supported by AWS and have to be emulated (RAID 0, NAS), or the applications using them need to be re-configured to not require such storage.

### Physical-to-Virtual Migration

As per the above AWS documentation, migration of VMs that were created by P2V migration (as opposed to VMs originally installed in the virtual environment, e.g. under VMware) is not supported.

To clarify: VMs created by P2V migration will often import to AWS just fine, but there may be hard-to-predict circumstances when the import will not work. While AWS will try to assist with these problems, the outcome is not guaranteed. Instead, AWS strongly recommends that the servers are re-built directly in AWS, or at least from scratch under VMware, and only then imported to AWS.

## Alternatives to Forklifting

While forklifting an existing physical or virtual environment is a possibility, and in certain scenarios may work very well, one should consider other options for migrating to AWS:

- Databases may be best backed-up, restored, synchronized (replicating on premise versions with the AWS ones), and then switched over. Third party database replication tools such as Attunity (http://www.attunity.com/) may prove useful during this process.
- Solutions relying on shared storage or very large storage devices should be re-designed, or use of third party cloud NAS should be considered.
- Application servers may be best re-installed from scratch.
- Windows Domain Controllers may be best moved as suggested above.
- Applications that do not clearly separate data and application layers may be candidates for forklifting.
- Load-balanced and high-availability setups may be best re-designed using AWS Auto-Scaling Groups and Elastic Load Balancers.
- High-availability solutions relying on ‘floating (virtual) IP’ patterns should be re-designed, or additional tooling has to be developed to make the servers AWS-aware.
- Backup and disaster recovery procedures may also be re-designed to take advantage of functionality of EBS Snapshots.