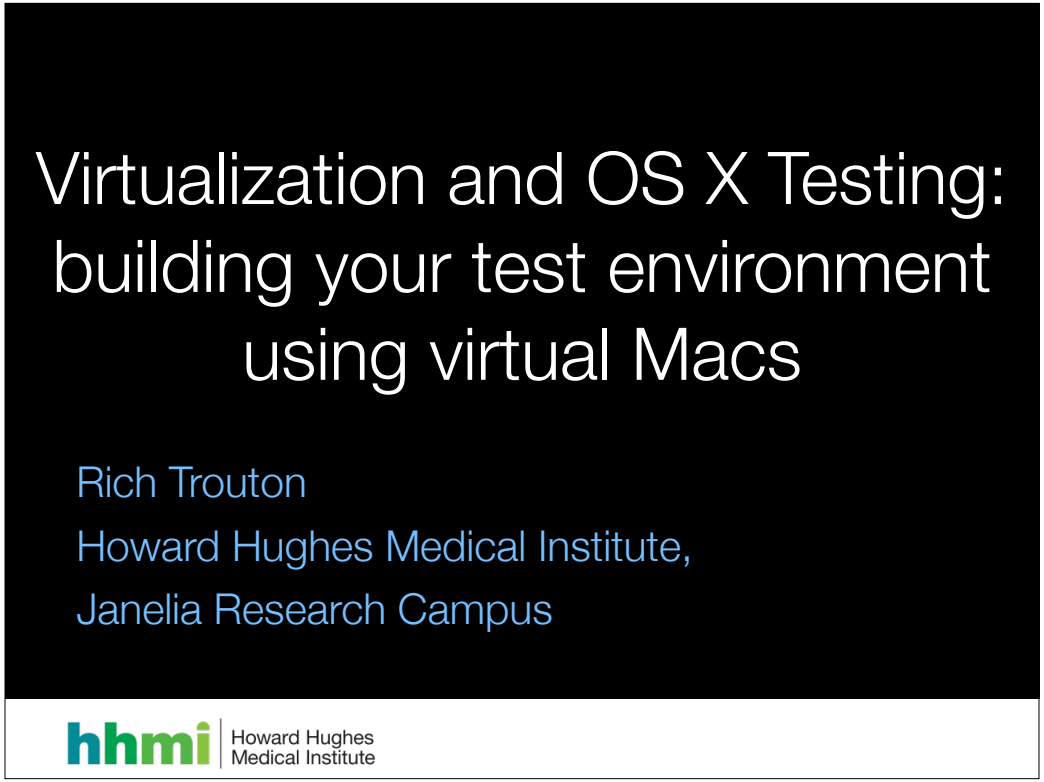
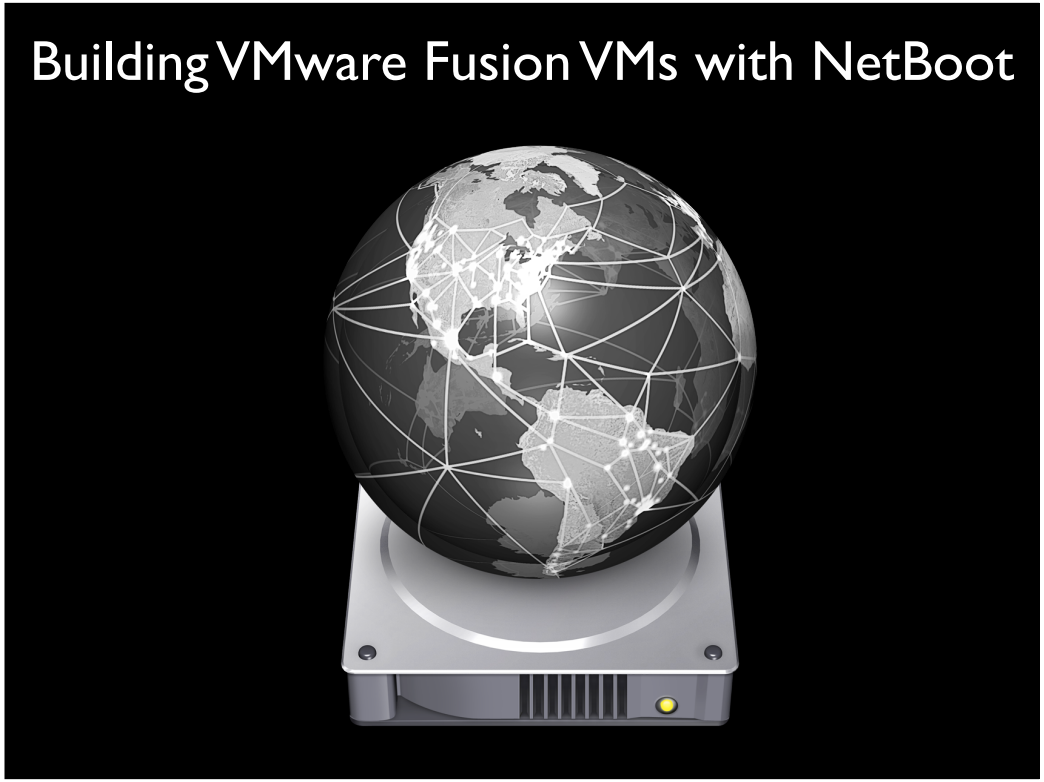


Before we get started, there's two things I'd like to mention. The first is that, all of the sides, speakers' notes and the demos are available for download and I'll be providing a link at the end of the talk. I tend to be one of those folks who can't keep up with the speaker and take notes at the same time, so for those folks in the same situation, no need to take notes. Everything I'm covering is going to be available for download.

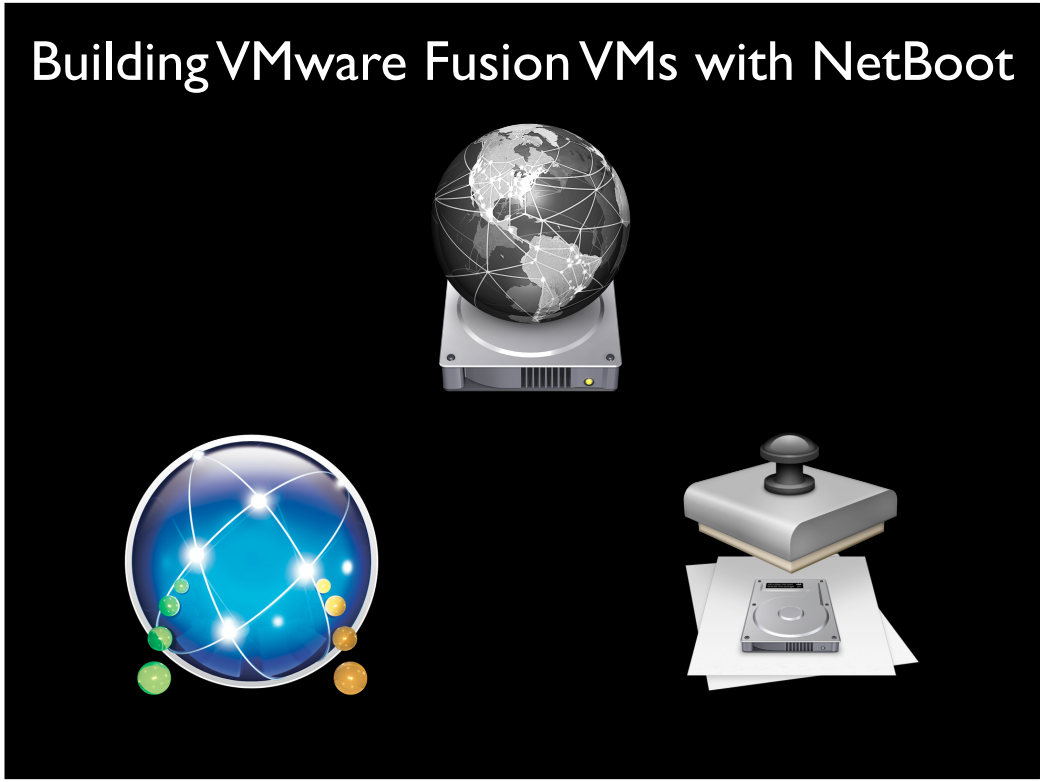
The second is to please hold all questions until the end. If you've got questions, make a note of them and ask me afterwards. With luck, I'll be able to answer most of your questions during the talk itself.

When new software appears, we all need test boxes that match our standard configuration in order to verify that the new software doesn’t adversely affect anything. This usually means admins need an available test box, or they have to go find one.

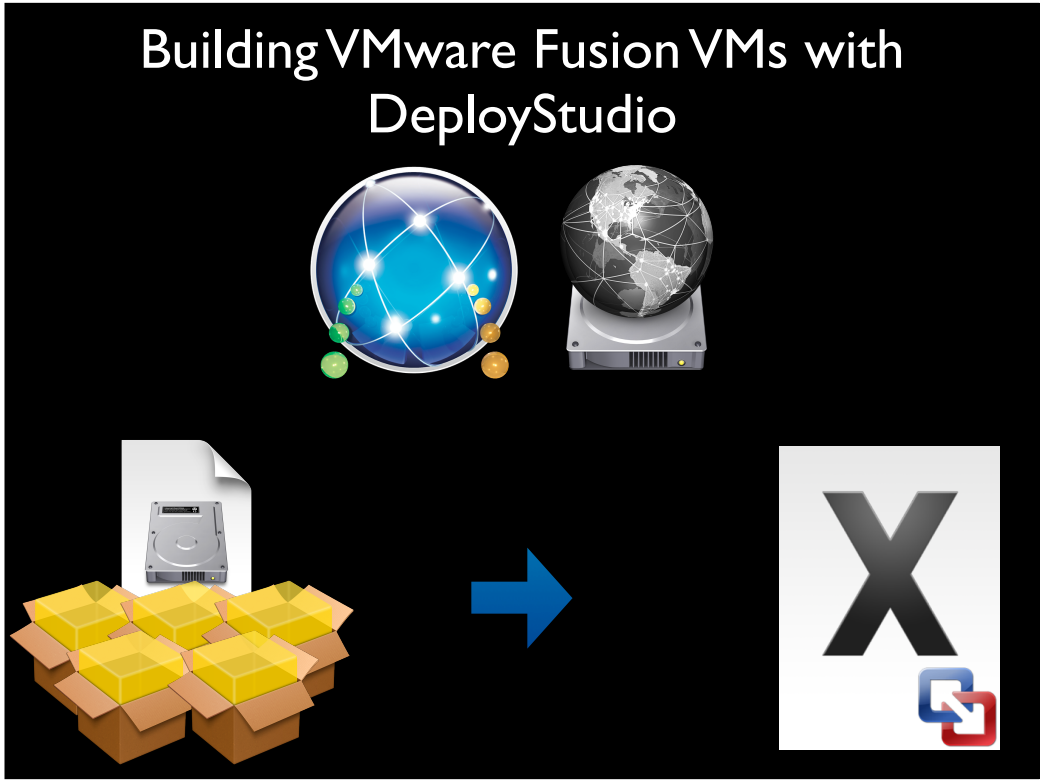
The advent of good virtualization solutions means it’s easier than ever before to perform testing without needing to collect and use physical test boxes. VMware Fusion and VMware ESXi have been the virtualization solutions I've been using, so let's talk about how you can use these tools to quickly build test VMs that match your needs.



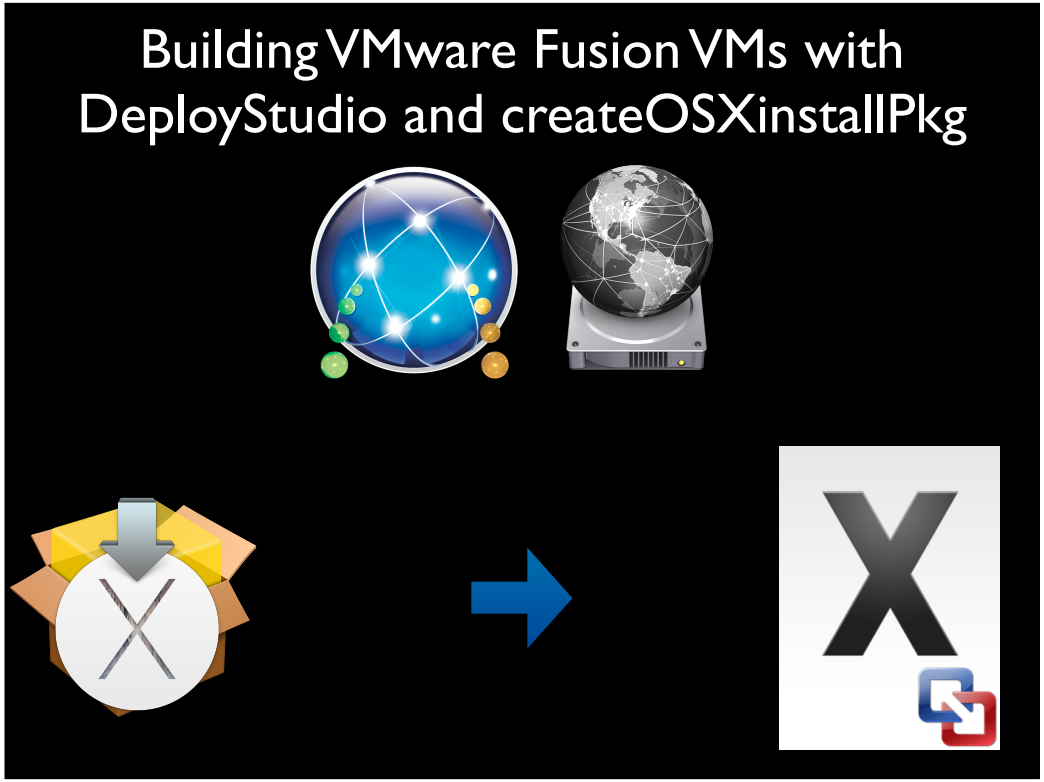
In VMWare Fusion 5, VMware added NetBoot support for virtual machines running Mac OS X. This proved to be an enormous boon to Mac admins who used NetBoot to help set up their machines because they could now build VMs like they built their users' Macs.



Two tools that can be used to build VMs with NetBoot are DeployStudio and Apple's System Image Utility. NetBoot sets associated with these tools allow Mac admins to boot their VM from the relevant NetBoot set and then apply whatever installs and configurations are desired.

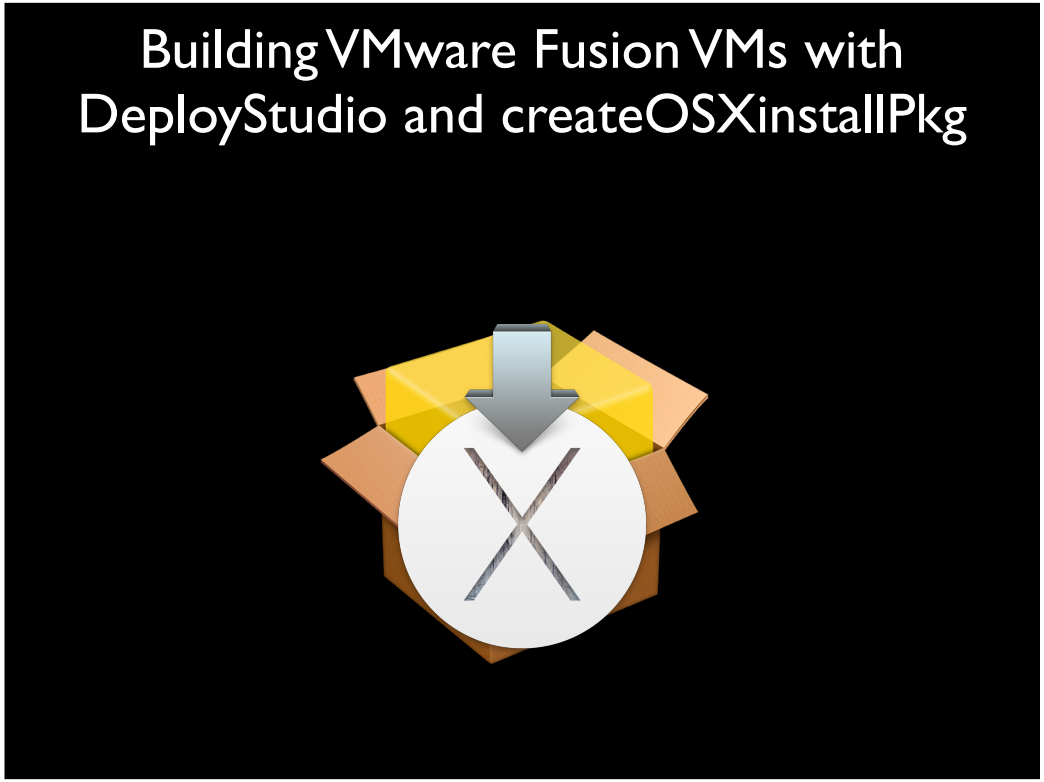


My preferred tool for building VMs is DeployStudio. DeployStudio offers great flexibility when building Macs and that same versatility can also be applied to VMs.



In my own shop, I'm using createOSXInstallPkg to generate OS X installers, which I then use with DeployStudio to help me build VM templates. For those unfamiliar with it, createOSXInstallPkg is a tool created by Greg Neagle at Disney. It is used to build individual installer packages that can install OS X Lion, Mountain Lion, Mavericks or Yosemite in the same way that you may install Microsoft Office or other applications.

The advantage of using this tool is that a number of system deployment tools for Macs can deploy the installers created by this tool, allowing OS X installations or upgrades to be performed by the system management tool already in use by a particular IT shop.

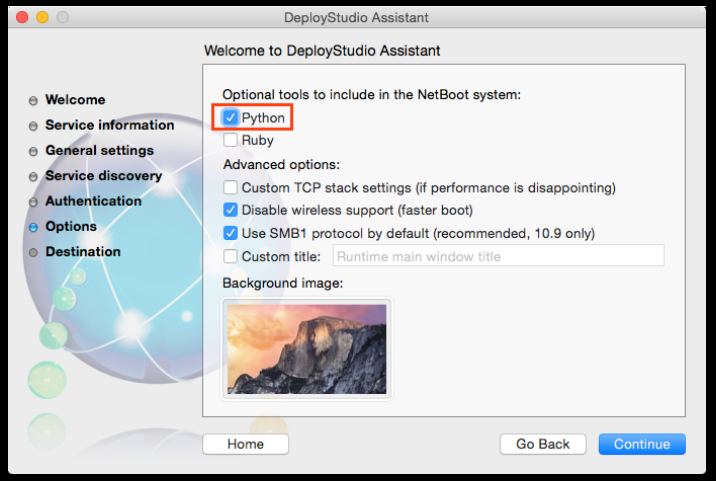


One great thing about using this tool is that createOSXInstallPkg will create an installer package that either installs a stock copy of OS X, or you can add additional packages to the stock OS X install.

There's a couple of guidelines to keep in mind here though. The first is that there is about 350 megabytes of free space available in the OS X installer. This is sufficient space for configuration or bootstrapping packages, but it's not a good idea to add Microsoft Office or similar large installers.

The second is that the limitations of the OS install environment mean that there are a number of installers that won't install correctly. In particular, packages that use preinstallation or postinstallation scripts may fail to run properly when those packages are run as part of the OS installation process. To help work around this limitation, I've developed a solution which I'll be discussing in more detail later in the talk.

Building VMware Fusion VMs with DeployStudio and createOSXinstallPkg

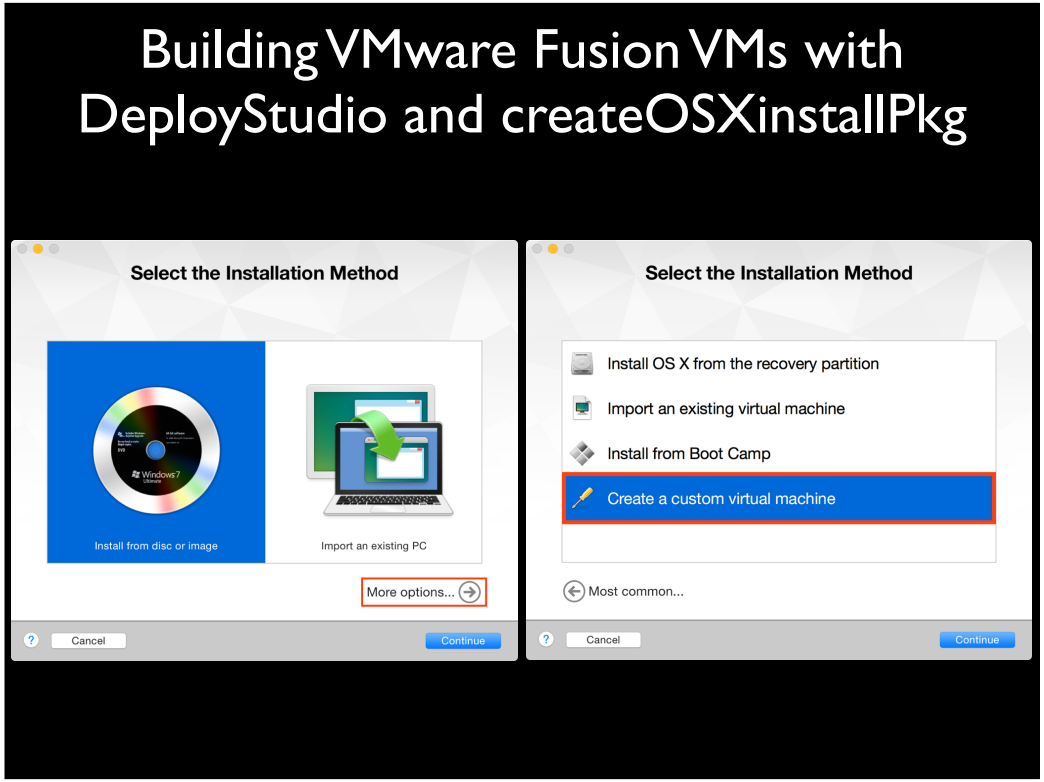


DeployStudio can deploy an OS installer built by createOSXInstallPkg, but you'll need to make sure you've added Python support. To add this support, you will need to create your DeployStudio boot set with Python selected as a tool to include in the bootable system.

As shown on the screen, adding Python is a checkbox option in the DeployStudio Assistant application that's used to create DeployStudio boot sets.

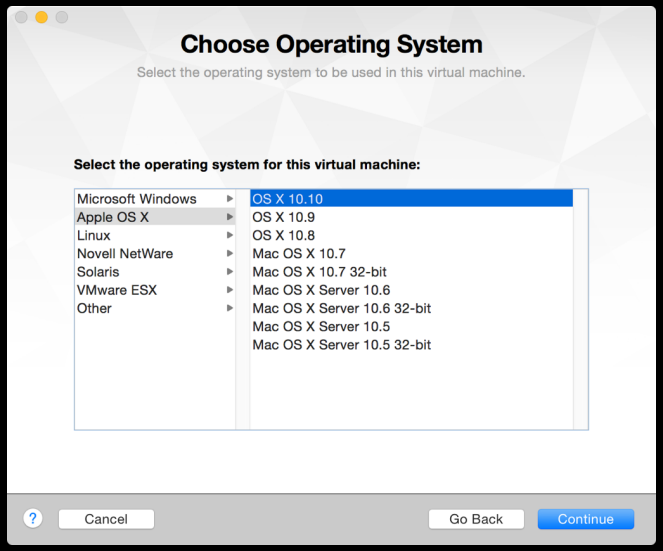
Building VMware Fusion VMs with DeployStudio and createOSXinstallPkg

To deploy the OS installer in a new VM, set up a new workflow in DeployStudio to install the package as a non-postponed install. This is important because it will set up a new VM's empty boot drive with the needed boot support to install OS X.

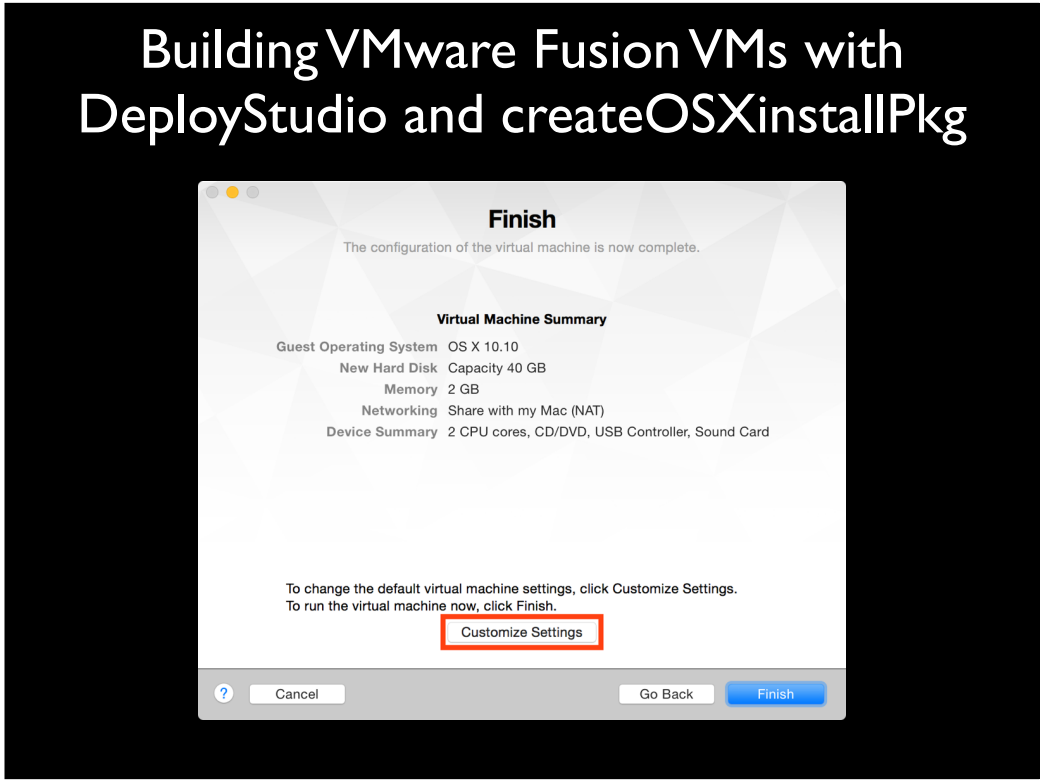


Once the necessary workflow has been set up in DeployStudio, you can set up a new VM. In VMware Fusion, my normal method is to create a customized VM. In the “Create a Virtual Machine” window, you can access this by selecting “More Options”, then selecting “Create a custom virtual machine”.

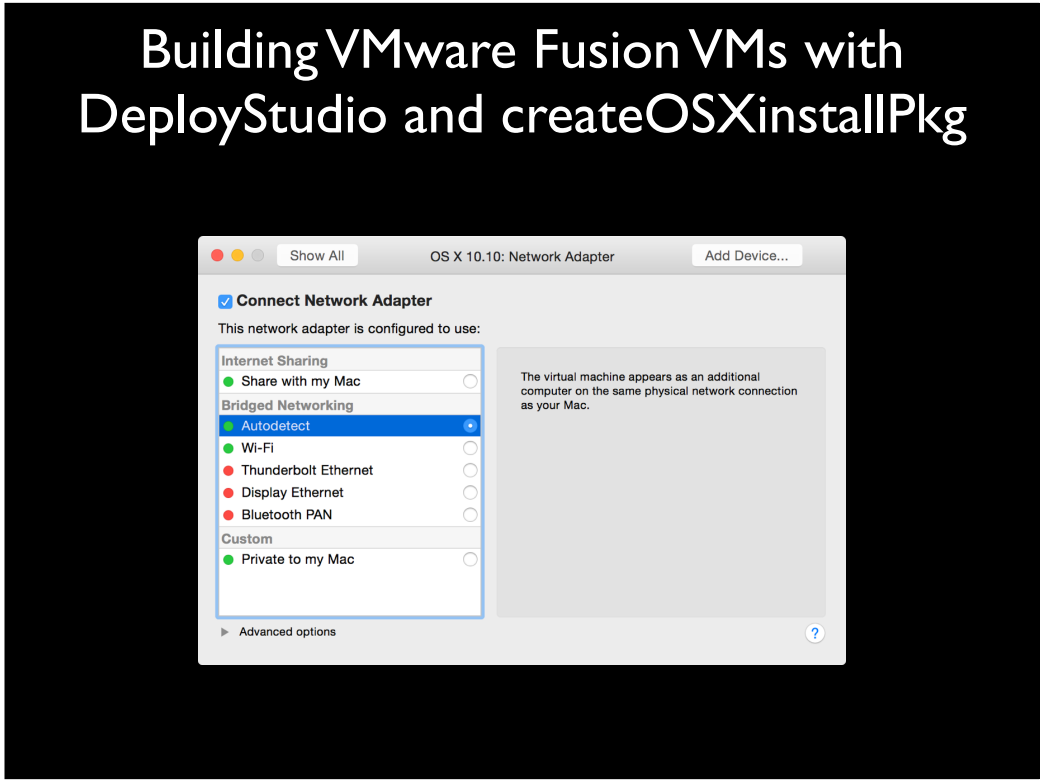
Building VMware Fusion VMs with DeployStudio and createOSXinstallPkg



In the “Choose Operating System” window, set OS as appropriate.

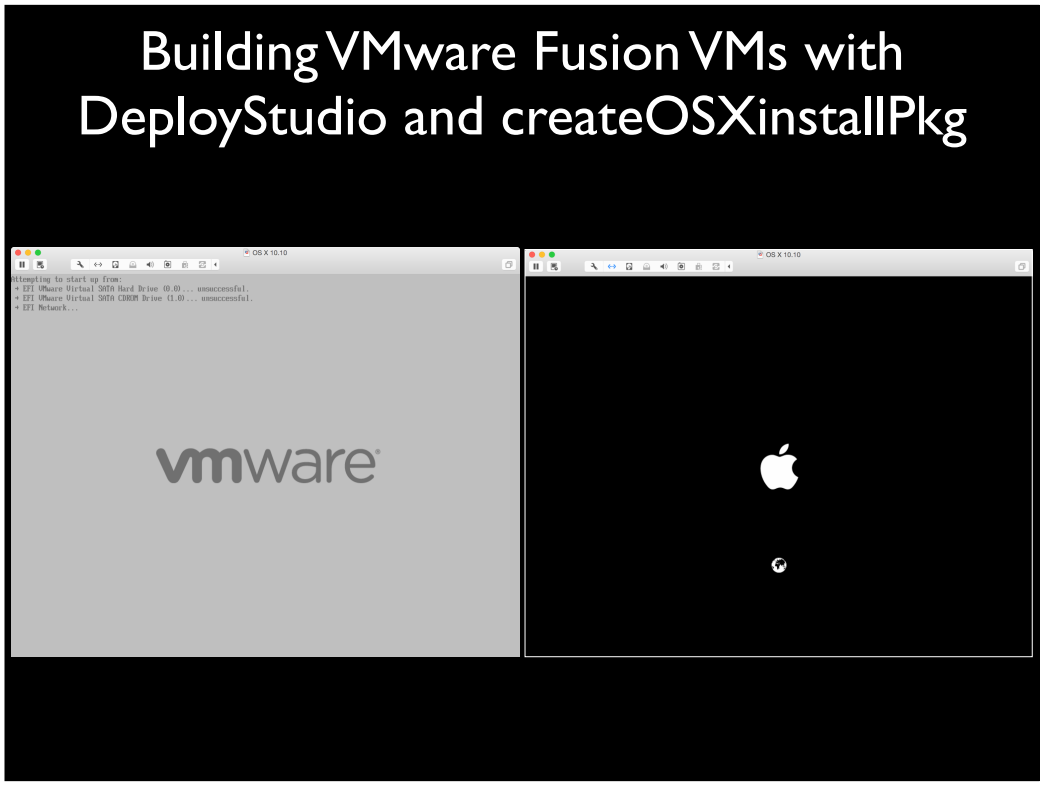


In the Finish window, select Customize Settings. This will allow you to change the VM's settings ahead of its first boot.



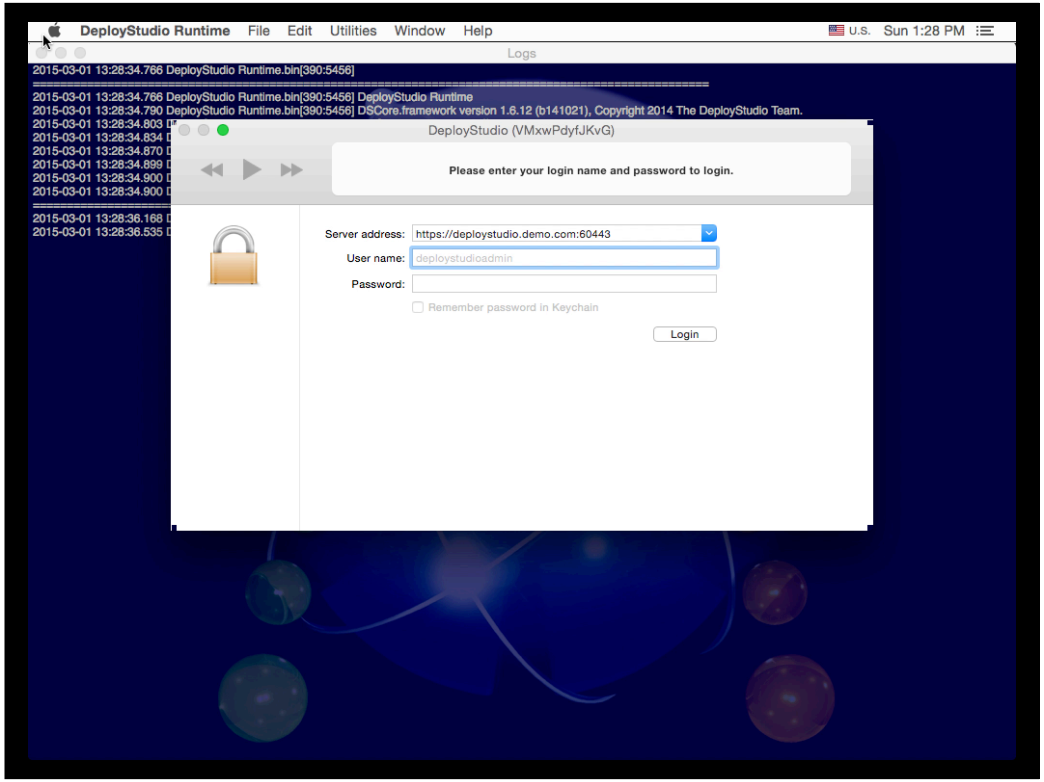
In the Network Adapter settings, select Autodetect under Bridged Networking. This will allow the VM to correctly boot from the NetBoot set. You may also want to adjust the VM's available RAM and other settings at this point, but that's up to you.

The VM is now ready for the next steps. In this example, the VM is configured to use Yosemite as its OS, but has a formatted and completely empty boot drive.

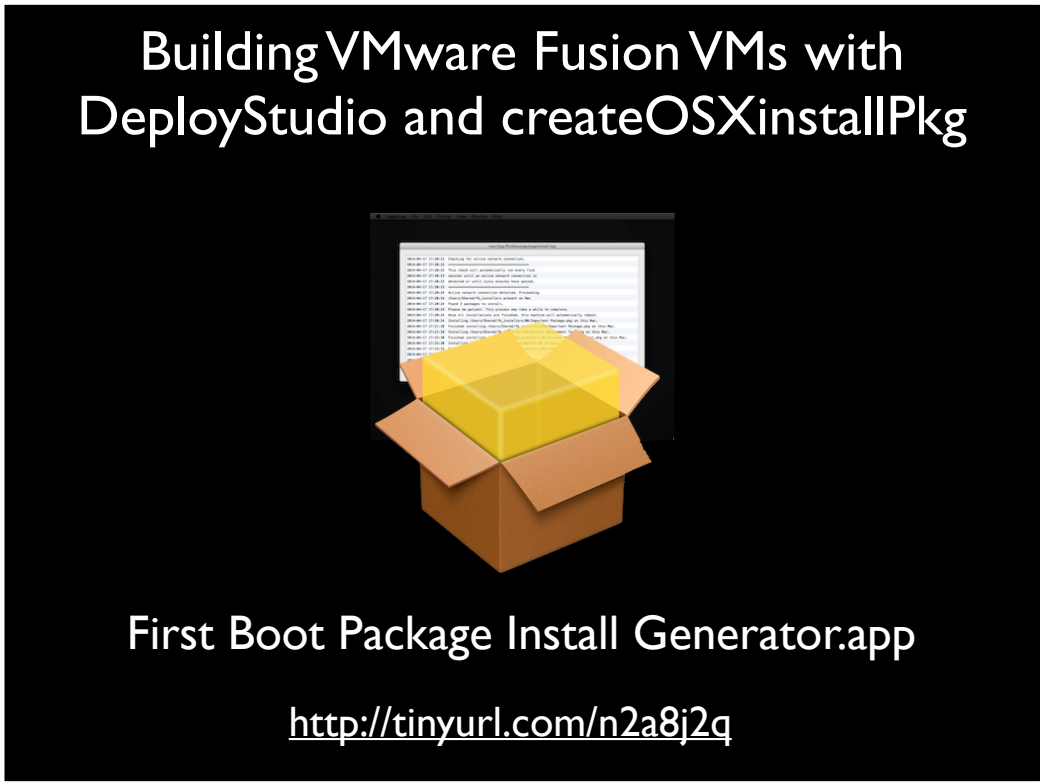


If the DeployStudio boot set is set as the default boot set on the NetBoot server, you can start the VM and then do nothing.

The VM should boot to DeployStudio automatically after failing to boot from either the VM's hard drive or optical drive. Alternatively, you can also hold down the N key on your keyboard to boot the VM from the default NetBoot set.



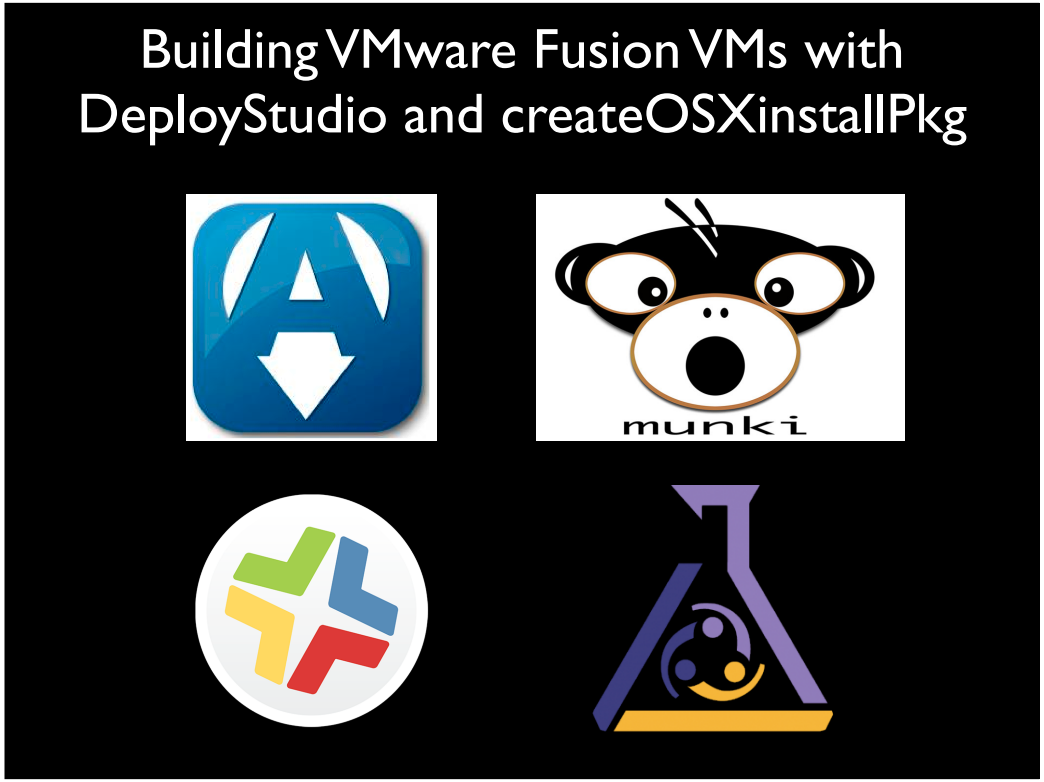
The example just described will set up a VM with only Yosemite installed, but you can customize further. In my own shop, I normally reboot back to DeployStudio and run additional workflows on the VM.



As mentioned previously, the limitations of the OS install environment mean some packages won't install correctly. In particular, packages that use preinstall or postinstall scripts as part of their normal installation process may fail to run properly in the OS install environment.

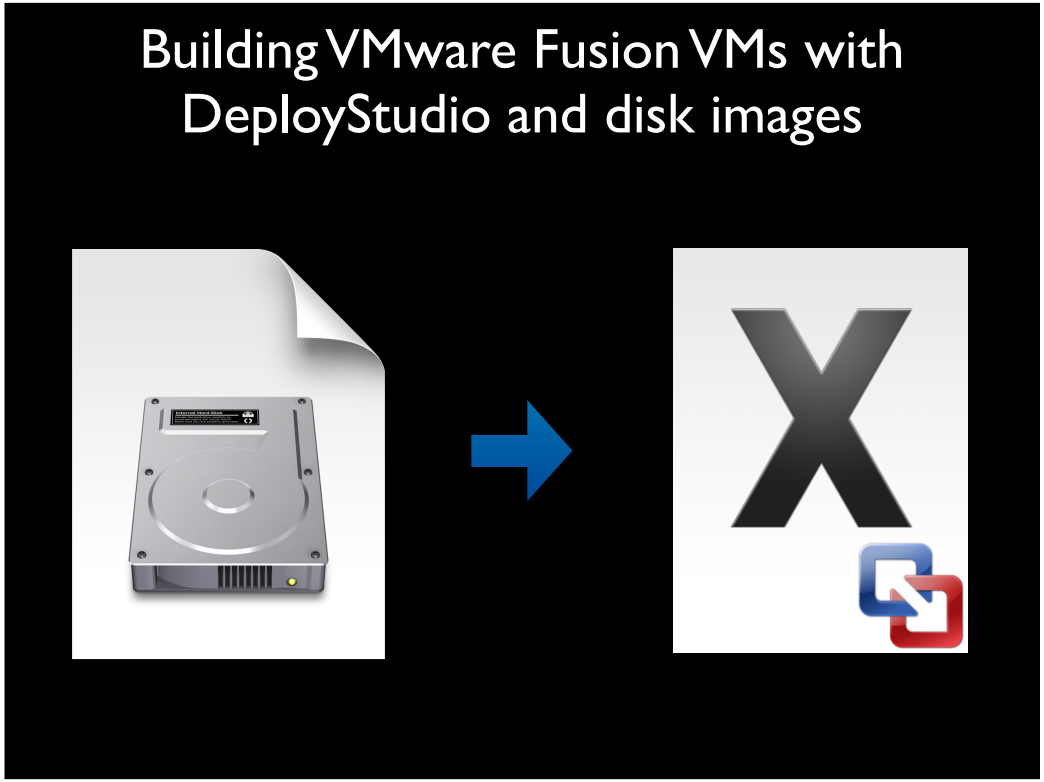
To help work around this limitation, I've developed First Boot Package Install Generator.app, an application that generates installer packages that enable other packages to be installed during the initial boot of either a Mac or a VM. This solves the issue because the installers are no longer running in the OS X install environment and can run any associated preinstall or postinstall scripts.

The First Boot Package Install Generator installer, along with the app's components and scripts, are available from GitHub using the link on the screen.



One potential use of a first boot package would be to allow you to add a systems management agent like Casper, Puppet, Absolute Manage or others to the OS installer. Once the agent reported in, the systems management tool could have its agent install additional software and scripts to configure the VM.

One management tool that would not require First Boot Package Install is Munki, which was also developed by Greg Neagle. Munki's tools can be added directly to a createOSXInstallPkg-built OS X installer.



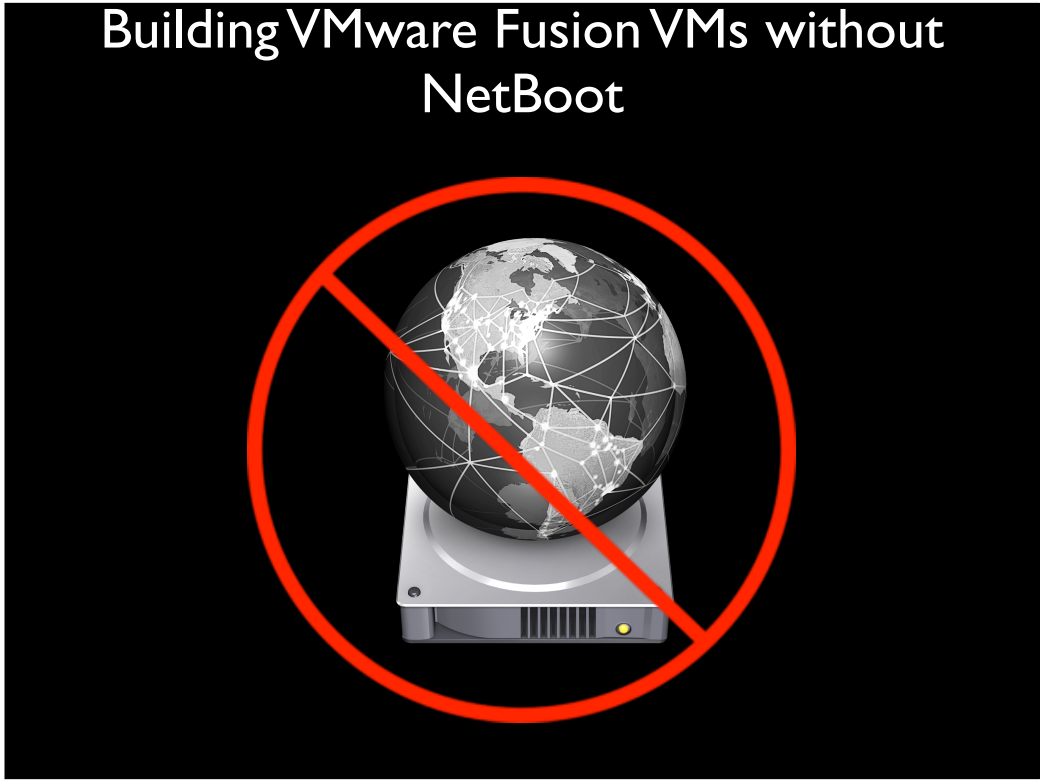
If you don’t want to use createOSXInstallPkg though, you don’t have to. You should be able to install a disk image into a VM like you can onto a Mac.

Building VMware Fusion VMs with DeployStudio and disk images

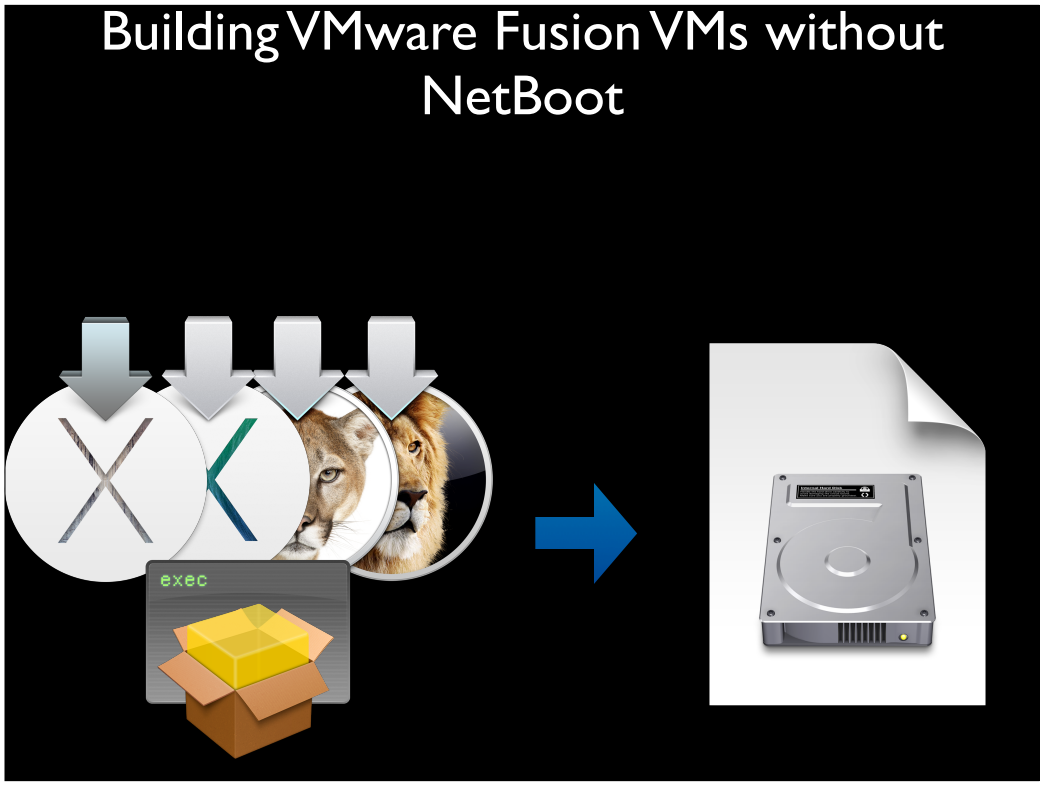
AutoDMG

https://github.com/MagerValp/AutoDMG/

For building disk images, I'd recommend using Per Olofsson's AutoDMG tool. I'd also like to note that "AutoDamage" is how Per has indicated he wants it to be pronounced, I have nothing but warmth and affection with regards to this tool.



As described previously, my preferred way to create VMs is by leveraging NetBoot and DeployStudio, but not all environments have access to NetBoot or DeployStudio.



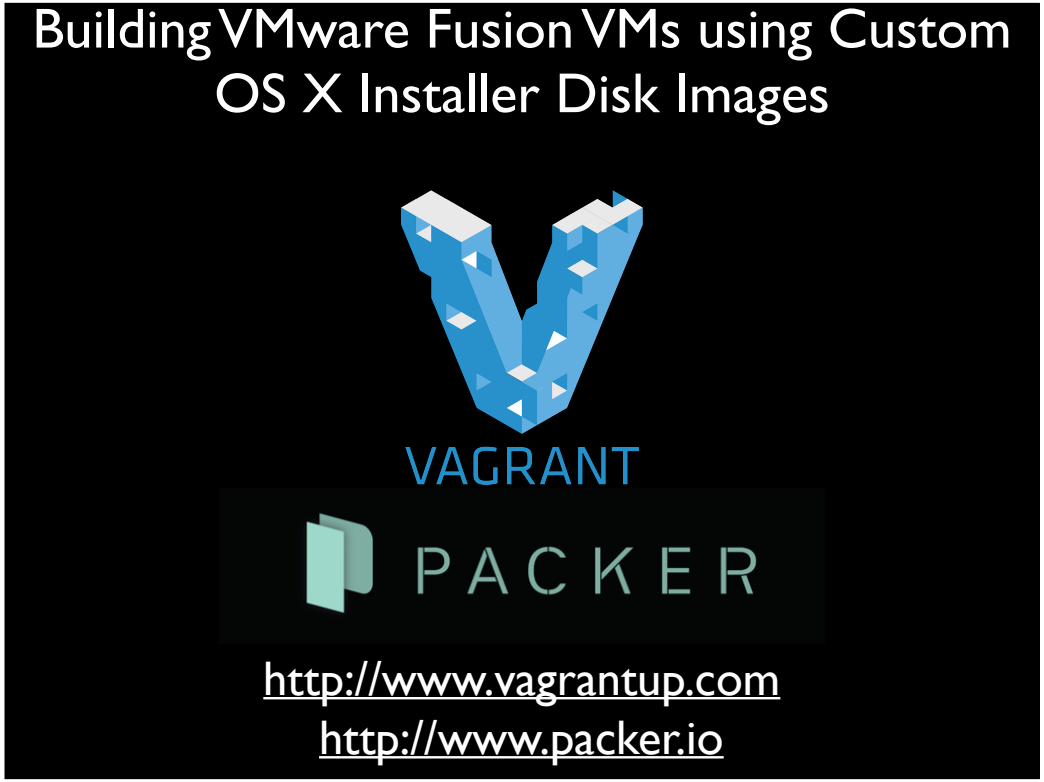
For those environments, there's scripted ways to create customized Lion, Mountain Lion, Mavericks, or Yosemite installer disk images for use with VMware Fusion. This allows the creation of OS X VMs that can configure themselves in an automated fashion without needing access to either NetBoot or server resources.

Building VMware Fusion VMs using Custom OS X Installer Disk Images

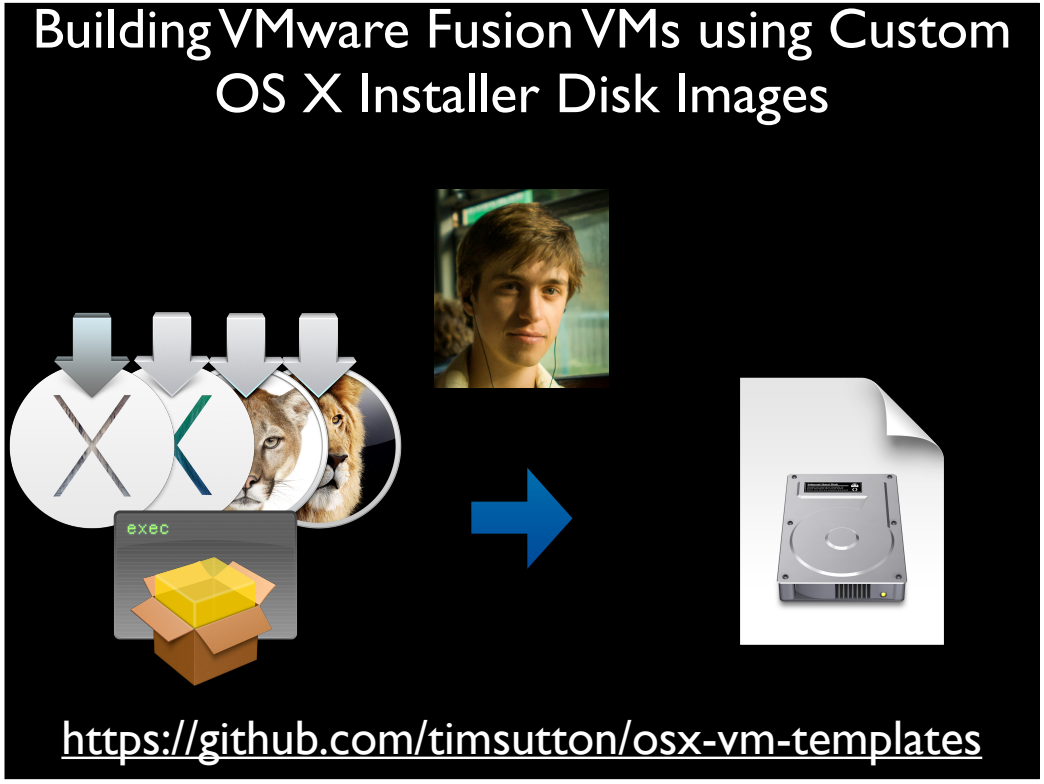
Tim Sutton

https://github.com/timsutton

Tim Sutton from Concordia University in Montreal, Canada was the first person I know of who applied this to OS X VMs, as part of his work with Vagrant and Packer.



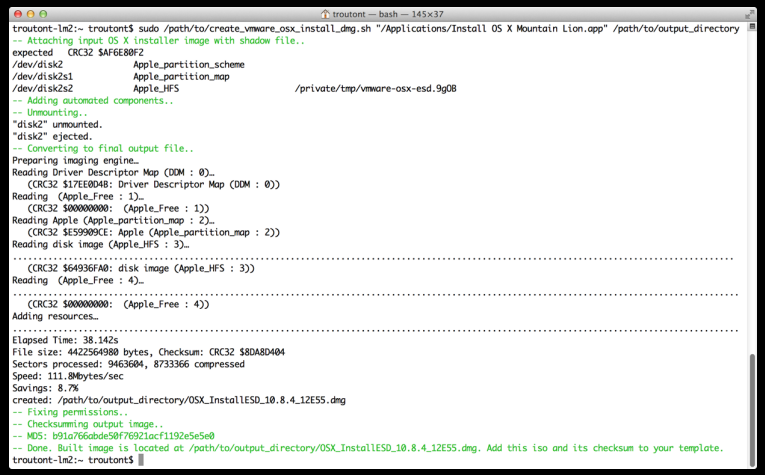
If you haven't previously heard of it, Vagrant is a tool for building virtual machines in a consistent, repeatable way using automated workflows. It's popular with development teams because it allows everyone working on a project to spin up identical virtual machines that contains their shared development environment. In turn, Packer is a complementary open-source tool that is used to build OS images for Vagrant to use when building new VMs.



As part of his work with Packer and VeeWee, another open-source tool for building OS images for Vagrant, Tim developed a script that would convert an OS X installer into a ISO disk image that either VeeWee or Packer could use.

Building VMware Fusion VMs using Custom OS X Installer Disk Images

create_os_x_vm_install_dmg.sh

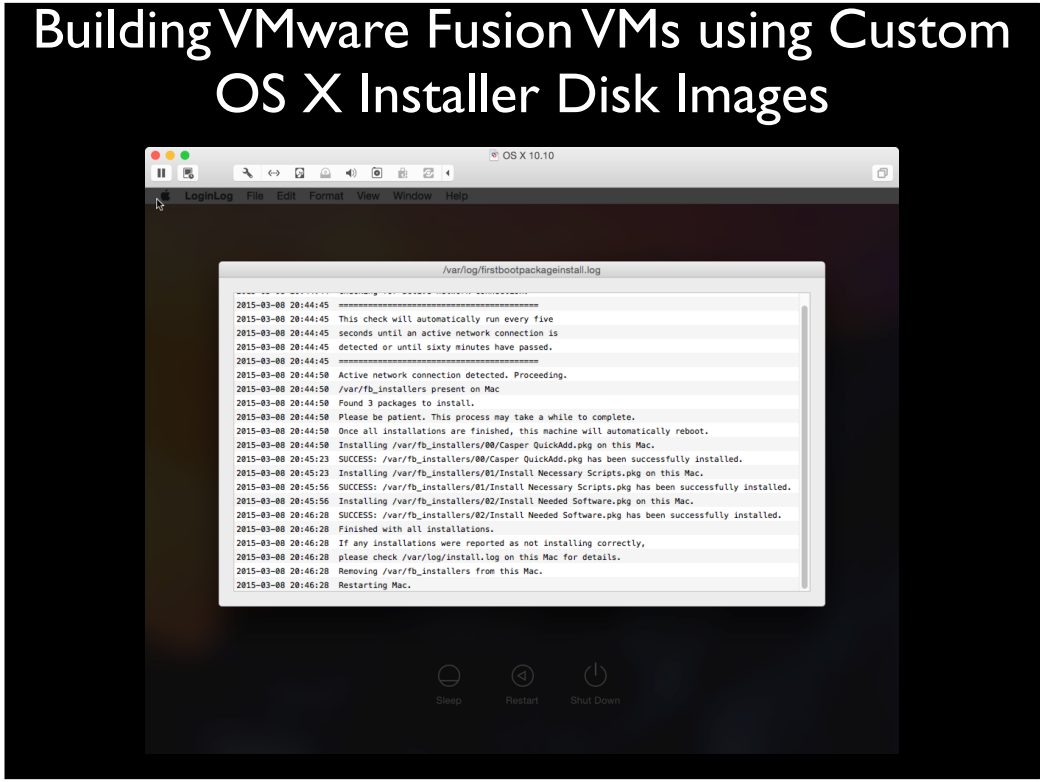


http://git.io/2L4uag

I was able to build on Tim's work to develop a script which creates a customized OS X installer that can be used in VMware Fusion without the need for either Vagrant or Packer. My method uses First Boot Package Install to provide customization for the OS X install.

As long as everything is configured according to the directions, users of this script will be able to produce a VMware-ready installer disk image that will install OS X and First Boot Package Install into a new VM.

When the VM boots, OS X and the first boot package will automatically install on the VM's boot drive. Once the installation completes, the VM will then reboot.



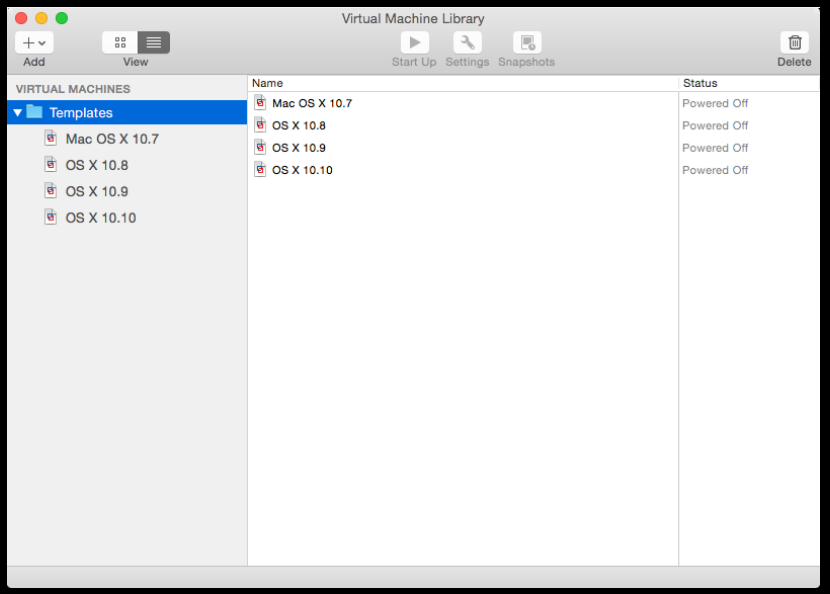
On reboot, a log will be displayed while the packages included with the first boot package are installed onto the VM. Once the packages finish installing, the VM will automatically reboot again.

After the second reboot, the VM will be set up with the desired applications and settings.

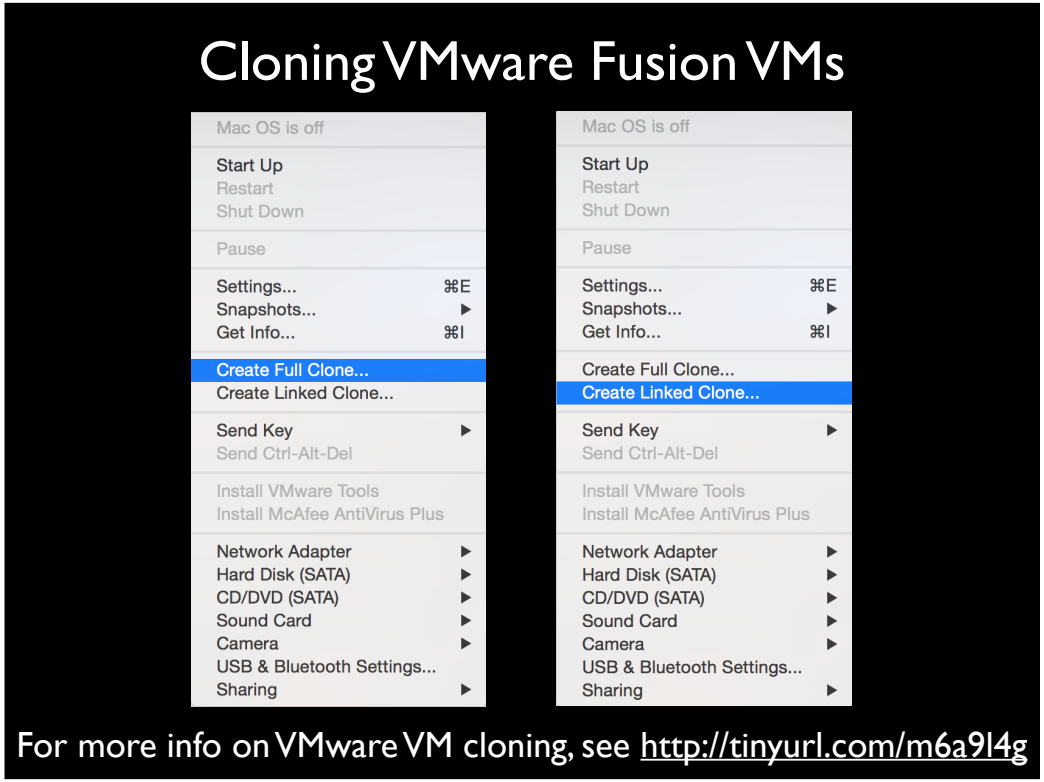
I'd previously mentioned this in connection with createOSXInstallPkg, but you can also use your existing systems management tools with a custom OS X install disk image to help you build and configure your virtual machines. In this scenario, you can build a simple installation process for your VMs that installs just the OS and your management tool's agent.

Once the agent on the VM phones home to the management service, the systems management tool can have its agent download and install additional software and scripts to configure the VM. Properly configured, this approach would allow VMs to be built with either no or very little effort on your part.

Working with VMware Fusion VMs



Once I've got virtual machines built in VMware Fusion, I prefer to use those as templates, or parent VMs. That way, I can use what I've built as a source for other VMs to follow.



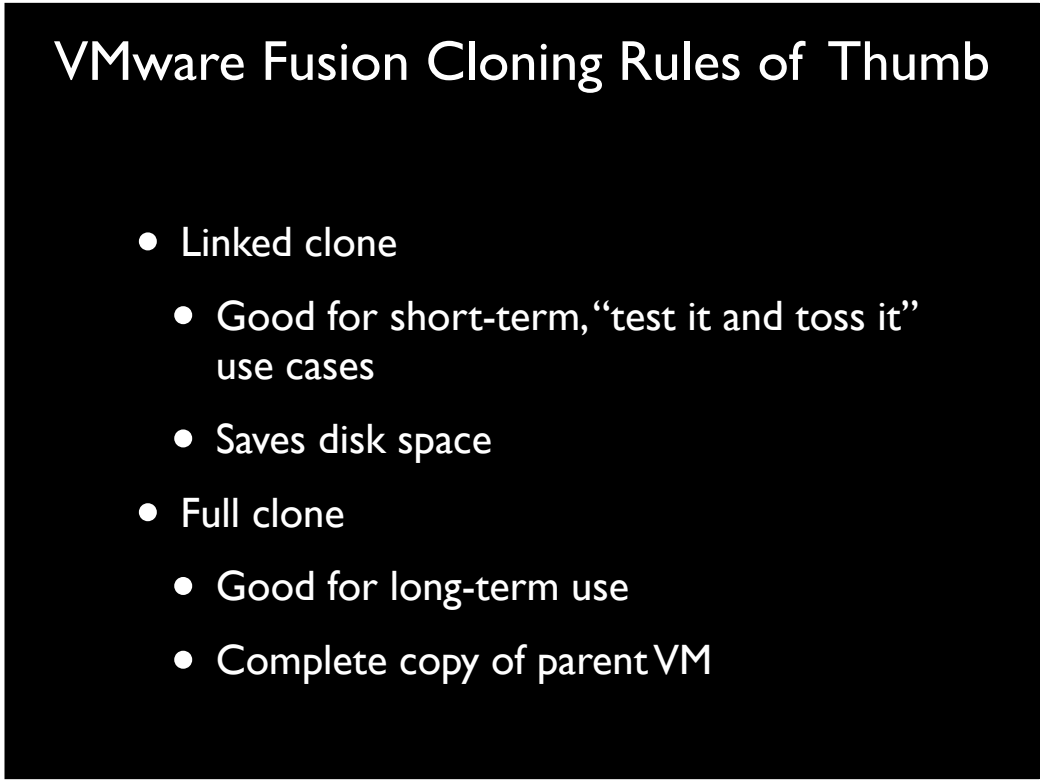
VMware Fusion 6 Professional added some functionality to facilitate this way of working, by bringing the ability to clone VMs from VMware Workstation on Windows and adding it to VMware Fusion on the Mac. There's two ways that you can clone a VM in VMware Fusion 6 or later. The first is by making a full clone and the second is by making a linked clone.

What's the difference between the two ways of cloning?

A full clone is an independent copy of a virtual machine. Ongoing operation of a full clone is entirely separate from the parent virtual machine and either the clone or the parent VM can be deleted without affecting the other.

A linked clone is made from a snapshot of the parent VM. A snapshot preserves the state and data of a virtual machine at a specific point in time. The state includes what's stored in memory, running applications and anything else the VM was doing at that time.

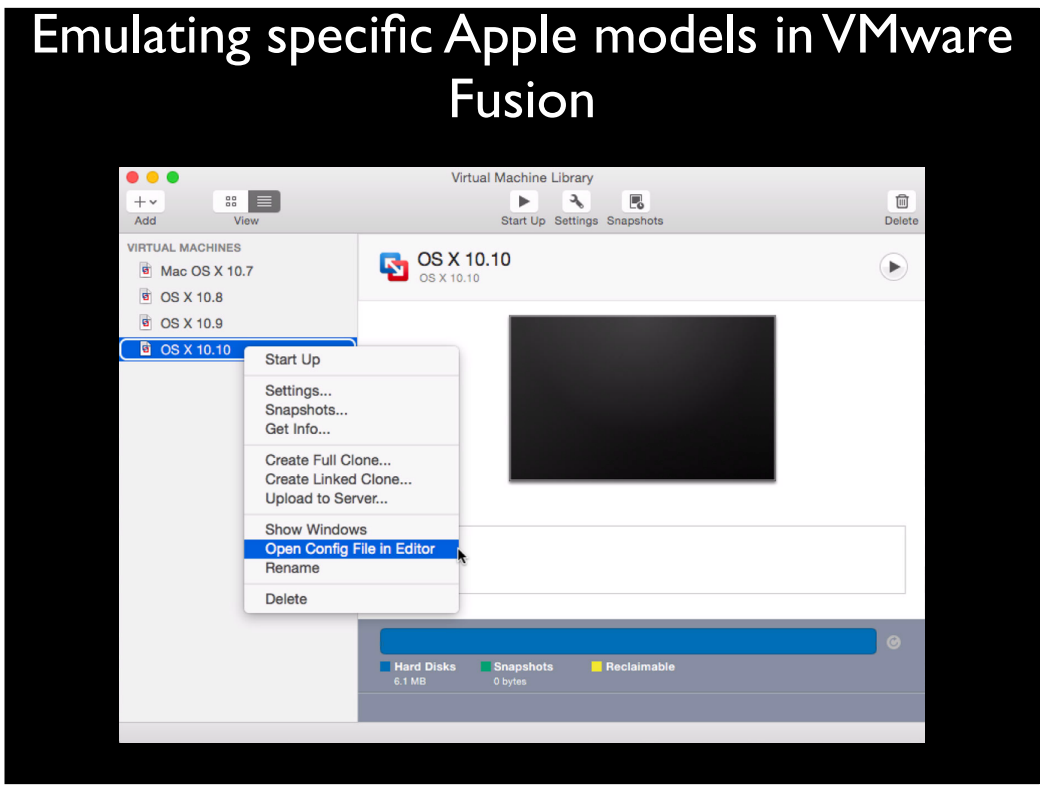
Using a snapshot saves space because the linked clone can reference all files available on the parent VM as of the time of the linked clone's creation, but it doesn't actually store a complete copy of the files. Because a linked clone is a



When I'm using clones in Fusion, I have the following rules of thumb. I do a lot of testing where I'm using a particular clone once or twice for a specific purpose, then tossing it. In this case where the test VM will only be around a short time, a linked clone makes a lot of sense because it saves on space and I don't have to worry about keeping it for the long term.

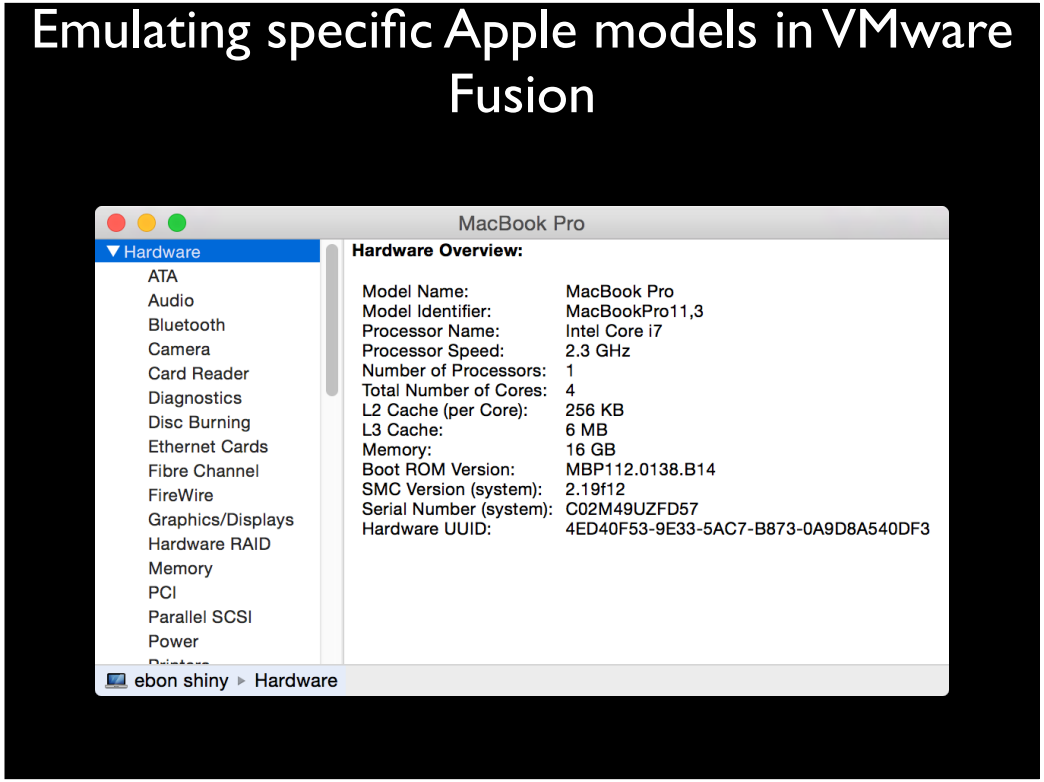
If I'm planning for a particular clone to stay around for a while, I make a full clone. This way, I don't have to worry about keeping track of the clone's parent VM because the full clone is a fully self-contained copy of the parent VM.

Emulating specific Apple models in VMware Fusion

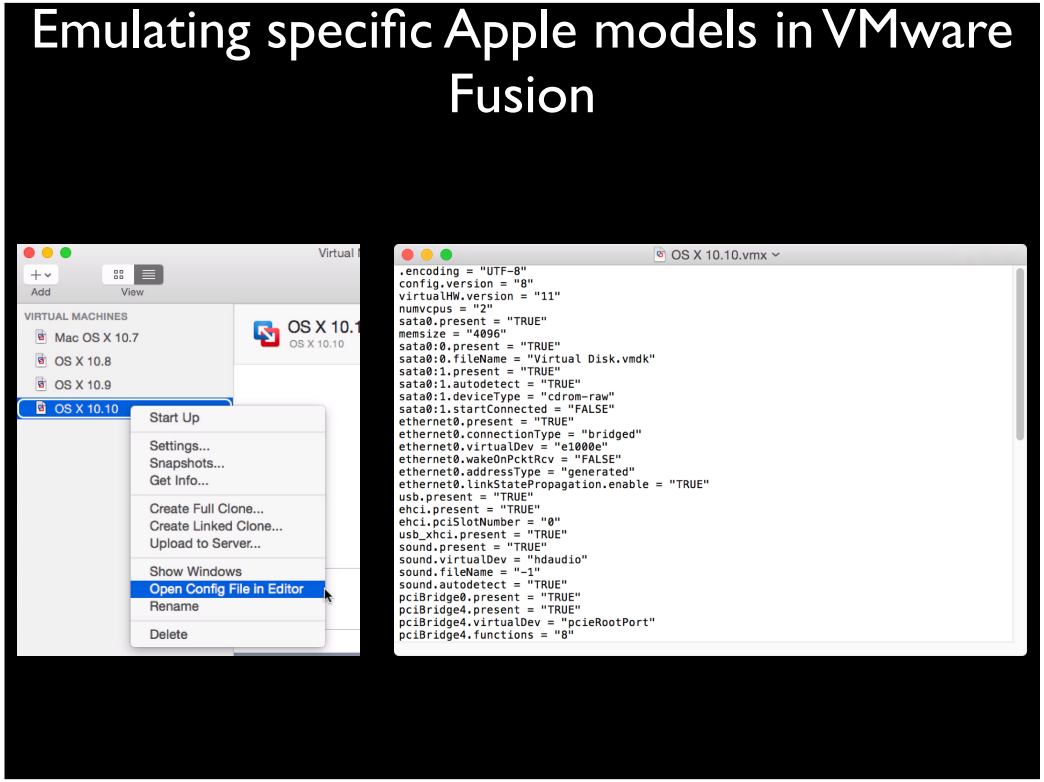


Once you have a VM built, you may want to edit it to emulate a specific Mac model. One reason for doing this would be to test model-specific updates from Apple's Software Update.

Emulating specific Apple models in VMware Fusion



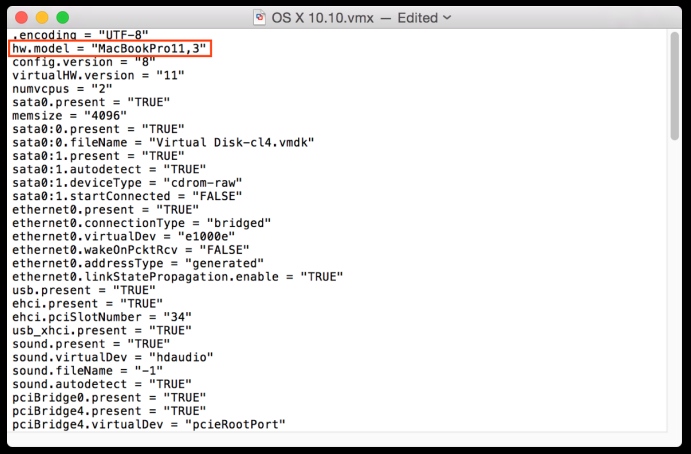
The first step is to locate the model identifier of the Mac you want to emulate. One way to do this is by checking in System Profiler on an appropriate machine. In the case of the machine on the screen, we're using the model identifier for a 2013 Retina MacBook Pro.



To set your VM to report itself as a specific Mac model, you would need to add hardware model settings to your VM's configuration settings. To do this, select the VM you want and make sure it's not running. Next, hold down the Option key on your keyboard and right-click the virtual machine.

Next, select Open Config File in Editor. This will make the VM's configuration available for editing.

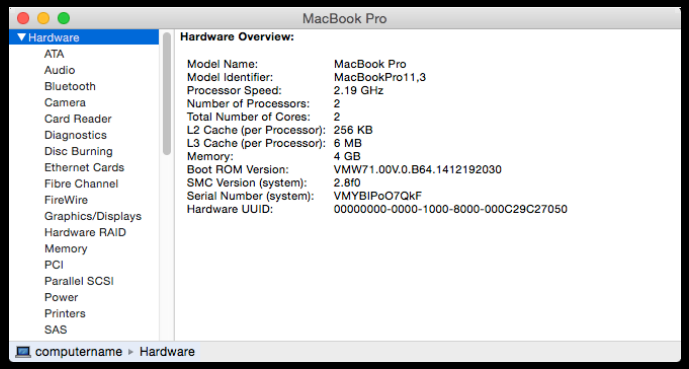
Emulating specific Apple models in VMware Fusion



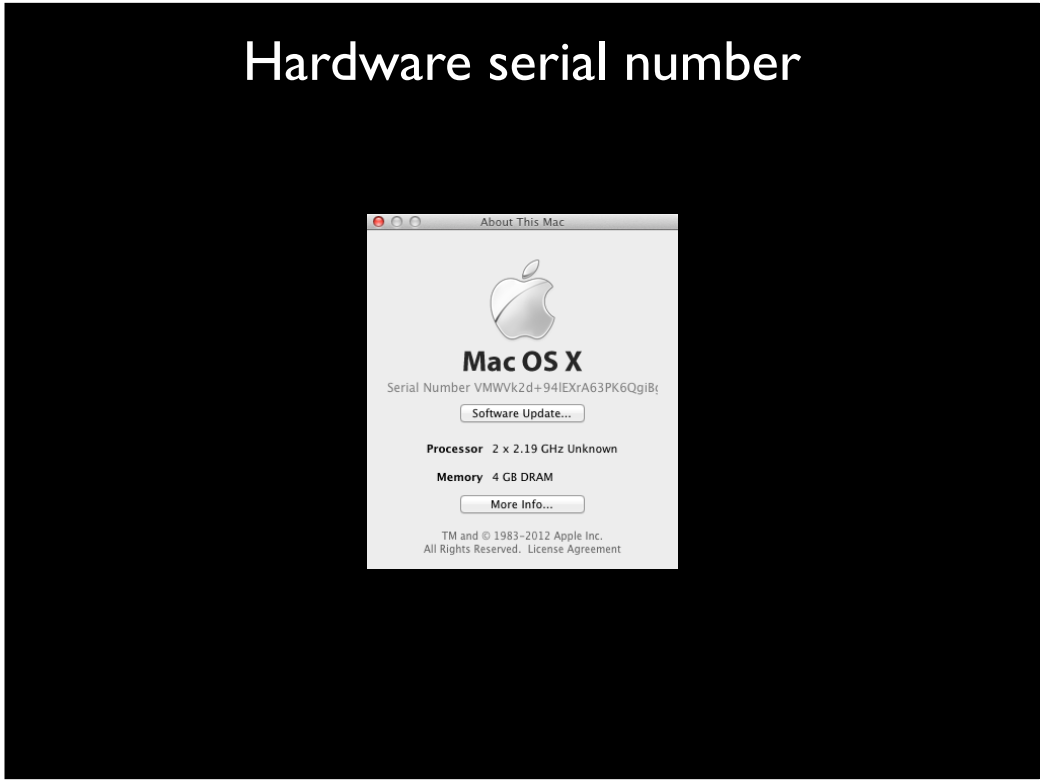
Add the following line to the VM configuration:
hw.model = "model_here"

In the configuration editor, you would add a line like that shown on the screen, substituting the actual model where I've got "model_here". Once your edits are finished, save your changes.

Emulating specific Apple models in VMware Fusion

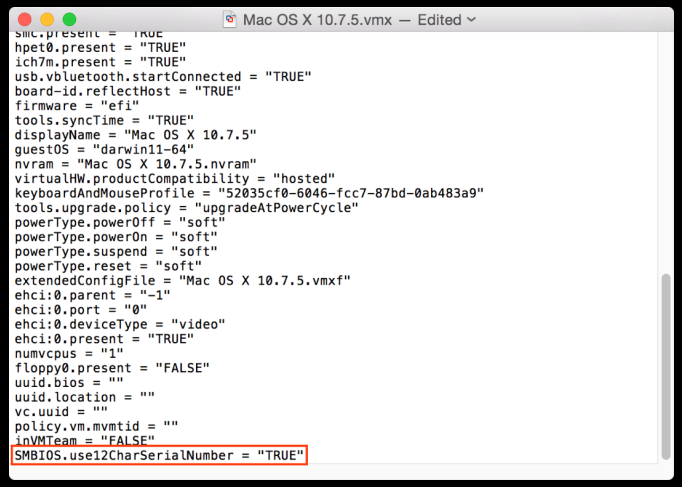


The next time you launch the VM, it should identify itself as being the specified Mac model. In the case of our example, the VM should identify itself as a MacBook Pro.



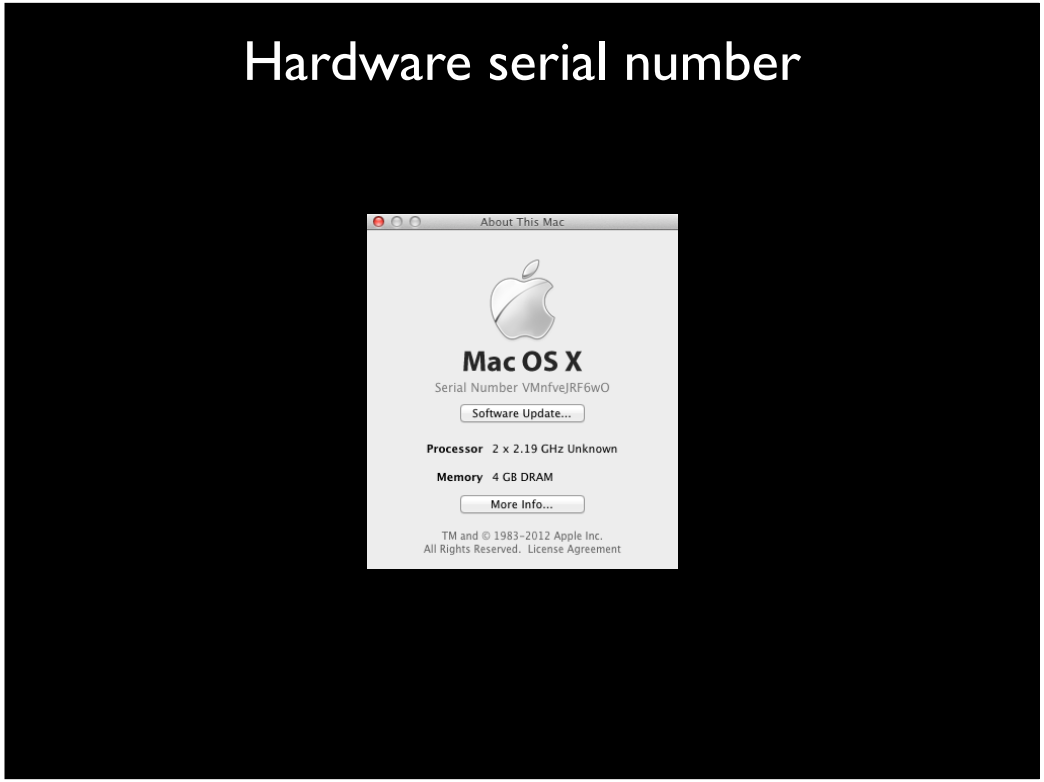
One issue you may run across in VMware Fusion VMs is that some services won't seem to work properly even though they look like they should. In this case, I recommend checking what the hardware serial number is set to be. In older guest OSs, like 10.7 or 10.8, this number may be longer than the 12 characters that Apple is expecting a Mac's serial number to be.

Using 12 character hardware serial number



Add the following line to the VM configuration:
SMBIOS.use12CharSerialNumber = "TRUE"

In these cases, VMware has included a way to generate a serial number that is 12 characters long so you can address this issue by adding settings to your VM's configuration file. To apply this, shut down the VM and then open the configuration editor. In the configuration editor, you would add a line like that shown on the screen. Once your edits are finished, save your changes and restart the VM.



When the VM starts up, the serial number should now be no longer than 12 characters long. This may solve some issues with profiles not applying and other odd problems.

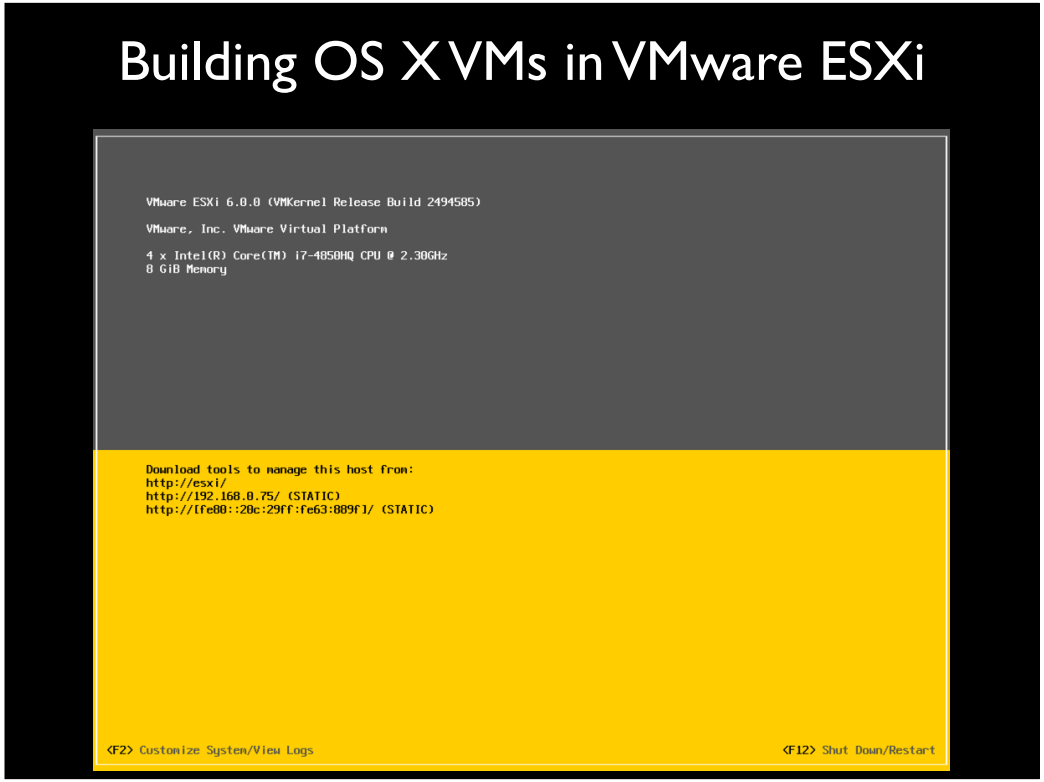
This option is enabled by default in VMware Fusion for VMs running Mavericks or later, but you may need to set it for VMs running Lion or Mountain Lion.

At this point, I'm going to start talking about OS X VMs and VMware's ESXi server. Before I do though, I want to say some things to hopefully save some questions later.

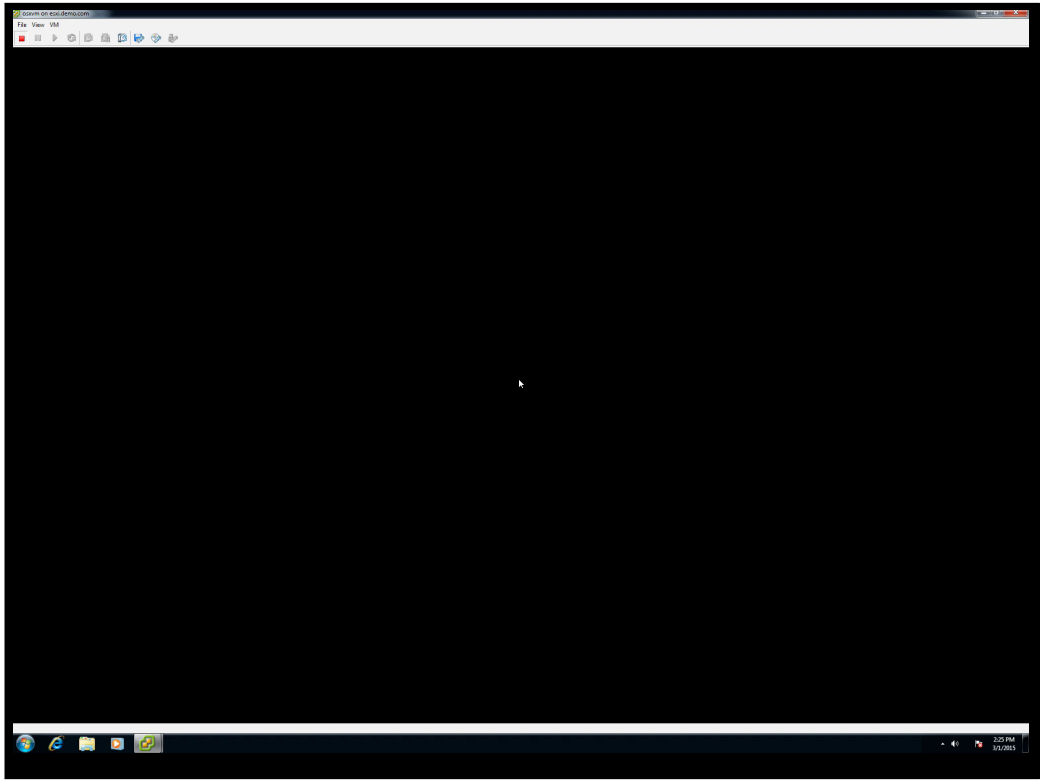
Anytime I mention Apple, VMware and ESXi in proximity to each other, I almost always have a conversation that goes like this. Someone asks if VMware now supports running OS X on non-Apple hardware. I tell them that no, OS X VMs are supported only on Apple hardware. The person usually responds that it would be nice if they could run OS X on their ESXi or vSphere setup like they can with other operating systems.

I fully agree with that opinion, it would be nice. Running OS X on non-Apple hardware is not a technical problem. It is a license issue. The people who can address this issue are at the location shown on the screen.

Building OS X VMs in VMware ESXi



Now that it's understood that I'm only talking about ESXi running on Apple hardware, let's talk about hosting OS X VMs on ESXi. In particular VMware brought some support over to ESXi 5.5 and 6 which completely changed how I built VMs on ESXi.

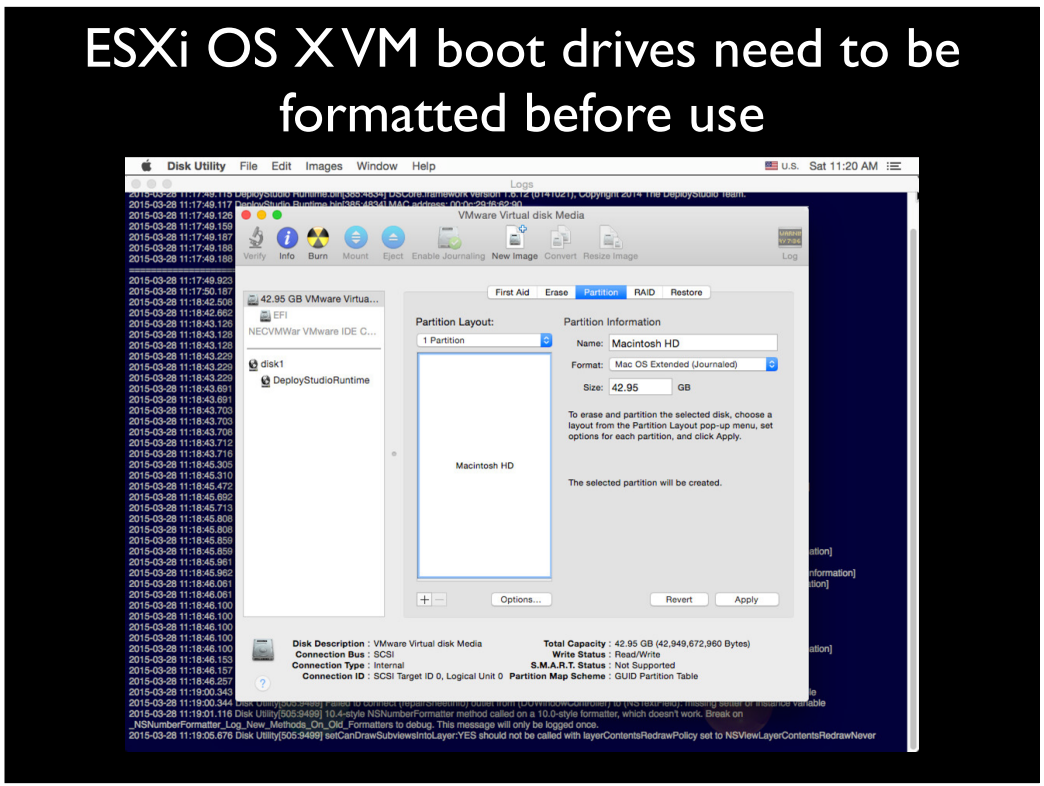


What changed things was that VMware added NetBoot support for ESXi-hosted OS X VMs. In fact, you can stand up a NetBoot server in one ESXi OS X VM and use it to boot other ESXi-hosted VMs.

Needless to say, this greatly simplified my build process because I can now leverage the same NetBoot deployment tools that I'm using with VMware Fusion.

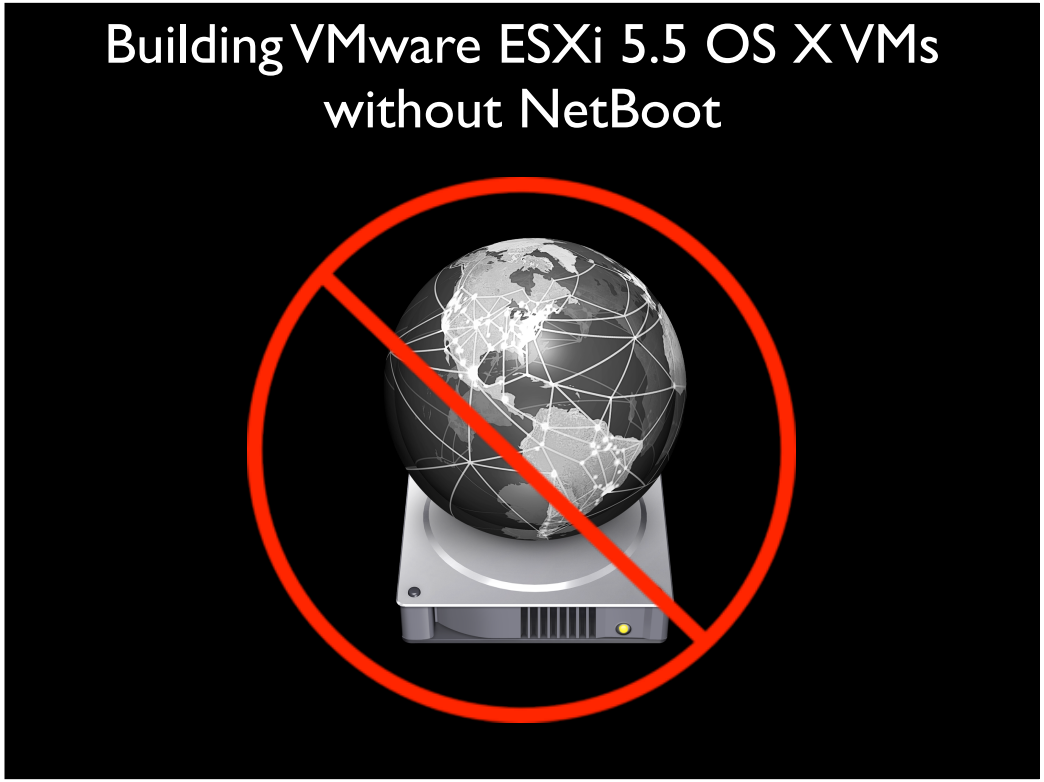
Support for NetBoot in ESXi has all but eliminated any need for me to build OS X VMs in Fusion first and then transfer them to an ESXi server.

ESXi OS X VM boot drives need to be formatted before use



One difference between building OS X VMs in Fusion and building them in ESXi is that Fusion will format the VM's boot drive as part of the creation of the VM. ESXi-hosted OS X VMs need to have their drives formatted. Unfortunately, DeployStudio's disk partitioning tools may not able to correctly detect the unformatted drive on an ESXi-hosted OS X VM like they can on a Mac, so it's difficult to include an automated format of the drive as part of the VM build process.

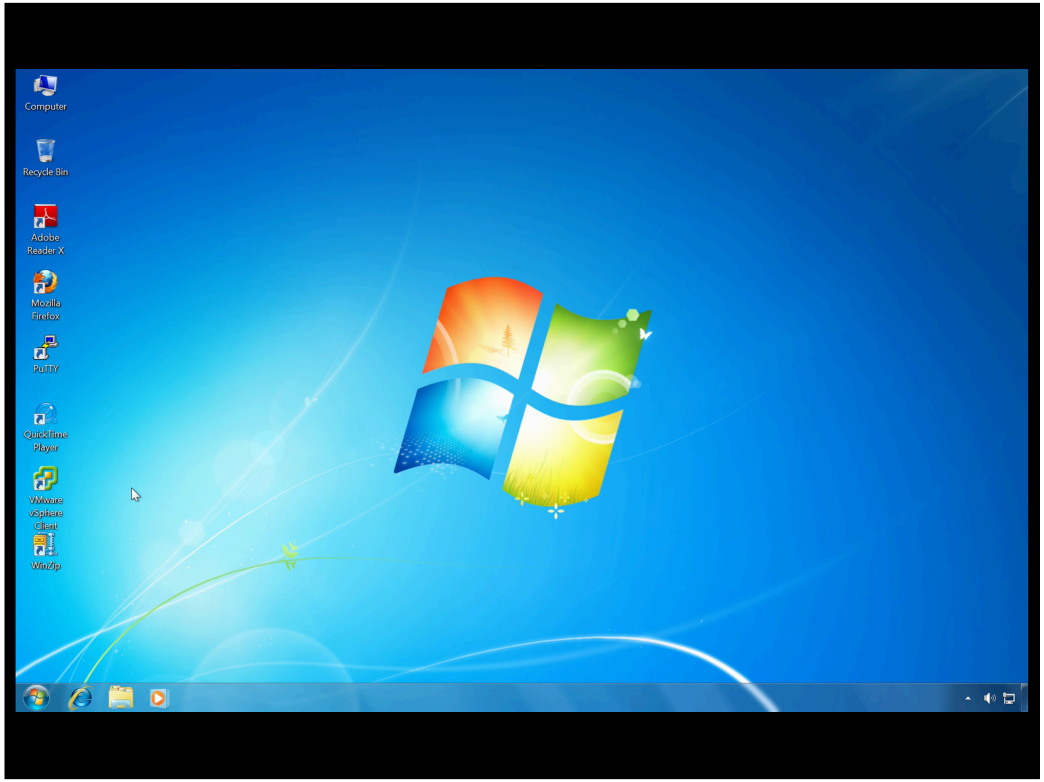
Fortunately, this issue can be solved while still booted into DeployStudio. One of the tools available on the DeployStudio boot set is Apple's Disk Utility, which can be used to format the drive before running a DeployStudio workflow on it.



Even with the ability to host a NetBoot server on ESXi, there are going to be environments where you can't use NetBoot. You can solve this issue also, using a tool we'd looked at previously for VMware Fusion.

One of the functions built into the custom OS X installer build script we discussed earlier is the option of creating an ISO disk image for use with ESXi.

Selecting the option in the script to create an ISO for ESXi will create two disk images, one for VMware Fusion and the other being an ISO for ESXi. You can use the ISO as an installer disk for an ESXi VM, much like you can use an OS X disk image as an installer disk for VMware Fusion VMs.

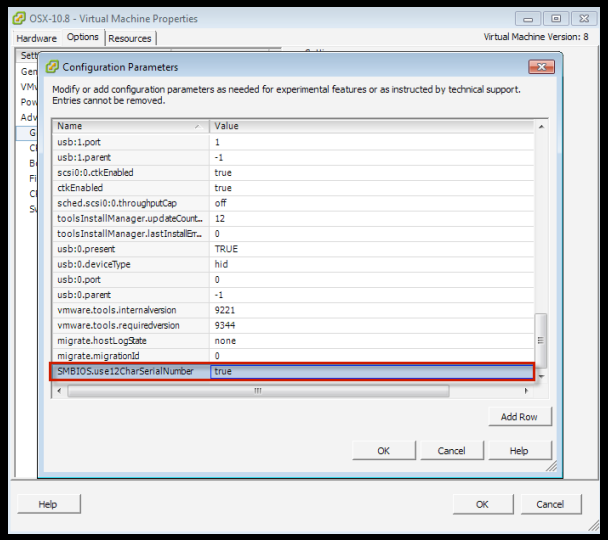


Let’s take a look at what the process of building an ESXi-hosted VM with a custom OS X installer looks like. In this instance, we’re building a VM using Yosemite.

Hardware serial number for ESXi-hosted VMs

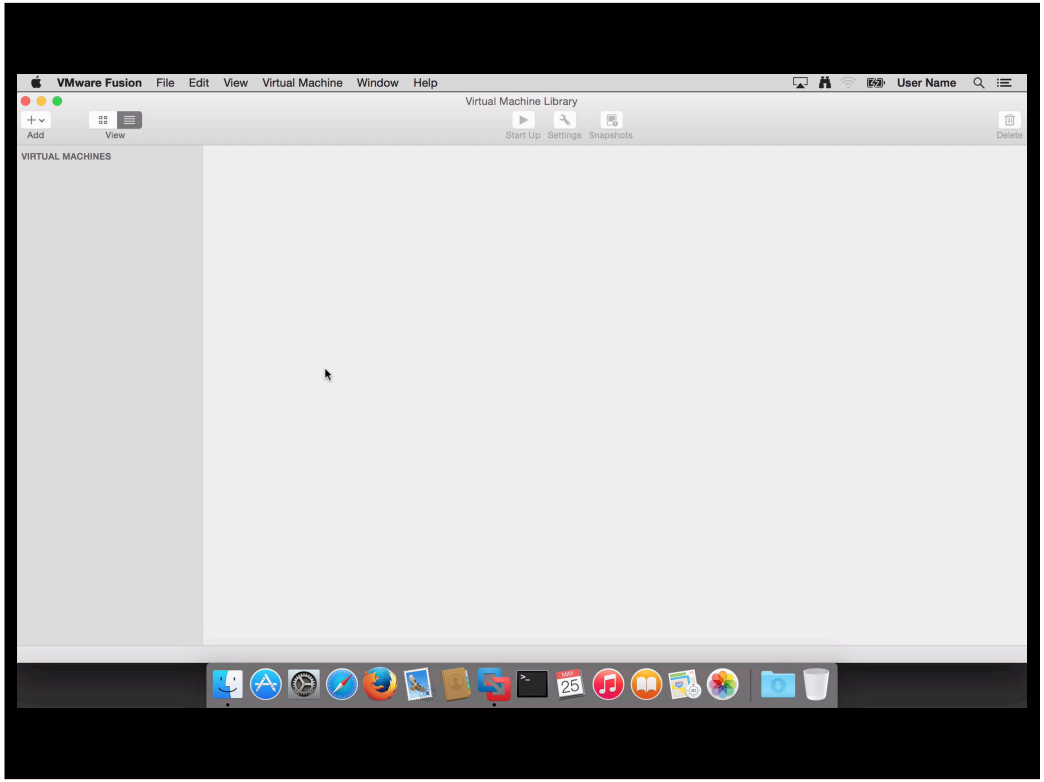
One issue you may run across in ESXi VMs is that the hardware serial number may be set to be longer than the 12 characters that Apple is expecting a Mac's serial number to be.

Hardware serial number for ESXi-hosted VMs

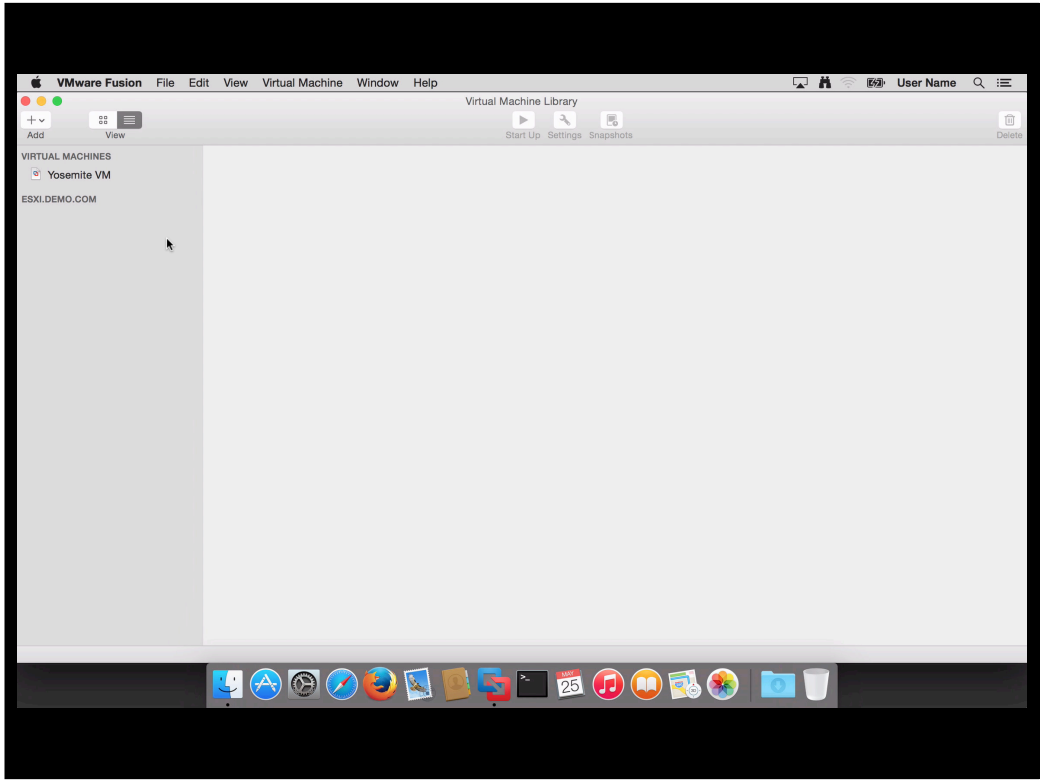


Name: SMBIOS.use12CharSerialNumber
Value: TRUE

You can fix this using roughly the same method that you would use in VMware Fusion. To apply this, shut down the VM and then open the configuration editor. In the configuration editor, you would add a line like that shown on the screen. Once your edits are finished, save your changes and restart the VM.



Once you have your VMs stood up on your ESXi server, a new feature in VMware Fusion 7 Professional is that you can connect to your ESXi server from Fusion. Depending on your management needs, you may be able to use Fusion instead of the Windows vSphere client.



Another capability is being able to upload VMs from Fusion to an ESXi server.

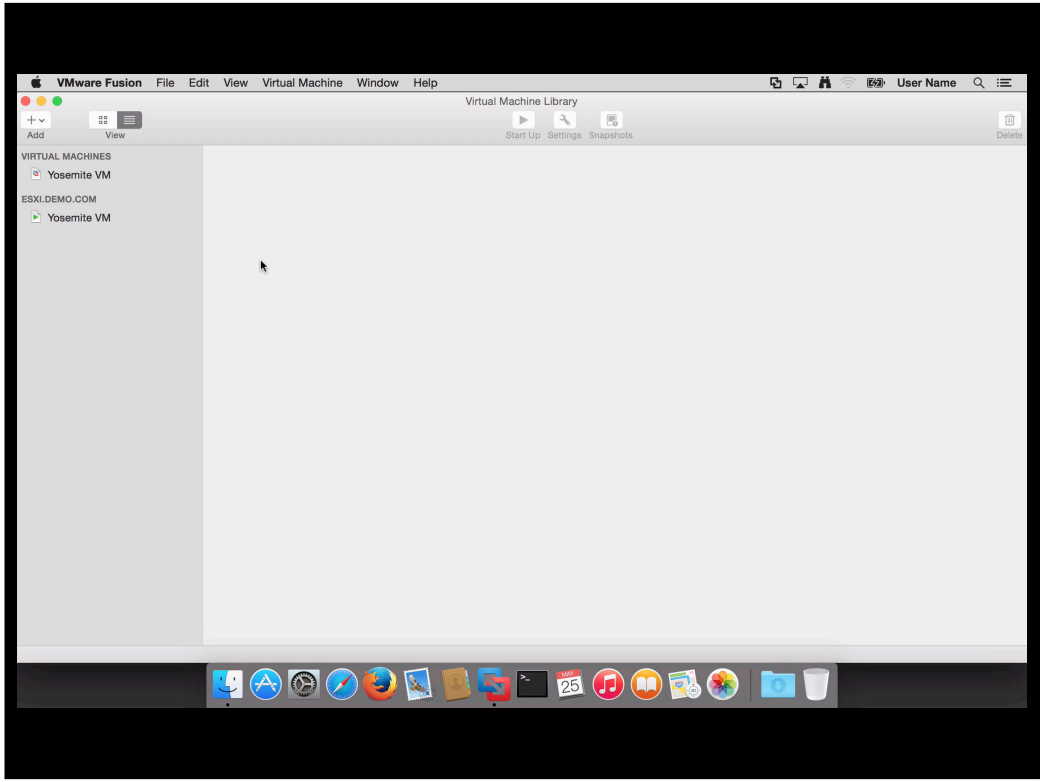
Uploading and downloading VMs

```
sata0:0.present = "TRUE"
sata0:1.startConnected = "FALSE"
sata0:1.deviceType = "atapi-cdrom"
sata0:1.fileName = "cdrom0"
sata0:1.present = "TRUE"
sound.startConnected = "FALSE"
sound.virtualDev = "hdaudio"
sound.fileName = "hdaudio"
sound.present = "TRUE"
vmci.filter.enable = "TRUE"
guestOS = "darwin13-64"
toolScripts.afterPowerOn = "TRUE"
toolScripts.afterResume = "TRUE"
toolScripts.beforeSuspend = "TRUE"
toolScripts.beforePowerOff = "TRUE"
tools.syncTime = "TRUE"
uuid.bios = "56 4d 7b 2e e4 48 ad bd-a0 b7 3c db f4 b1 21 0b"
uuid.location = "56 4d 7b 2e e4 48 ad bd-a0 b7 3c db f4 b1 21 0b"
vc.uuid = "52 e3 0e 68 b2 f1 6f fe-5e 80 d3 f1 a2 fa e6 5a"
smc.present = "TRUE"
```

Add the following line to the VM configuration:
smc.present = "TRUE"

There is an issue to be aware of when uploading or downloading VMs between VMware Fusion and ESXi 6, which is that the necessary smc.present = “TRUE” attribute for an OS X VM’s .vmx configuration file will not be transferred.

The smc.present = "TRUE" attribute allows a VM with OS X as the guest OS to check and detect that it’s running on Apple hardware. Without this check succeeding, the VM cannot verify it is running on Apple hardware. Unless that check succeeds, OS X VMs will not complete startup successfully and will appear to hang.

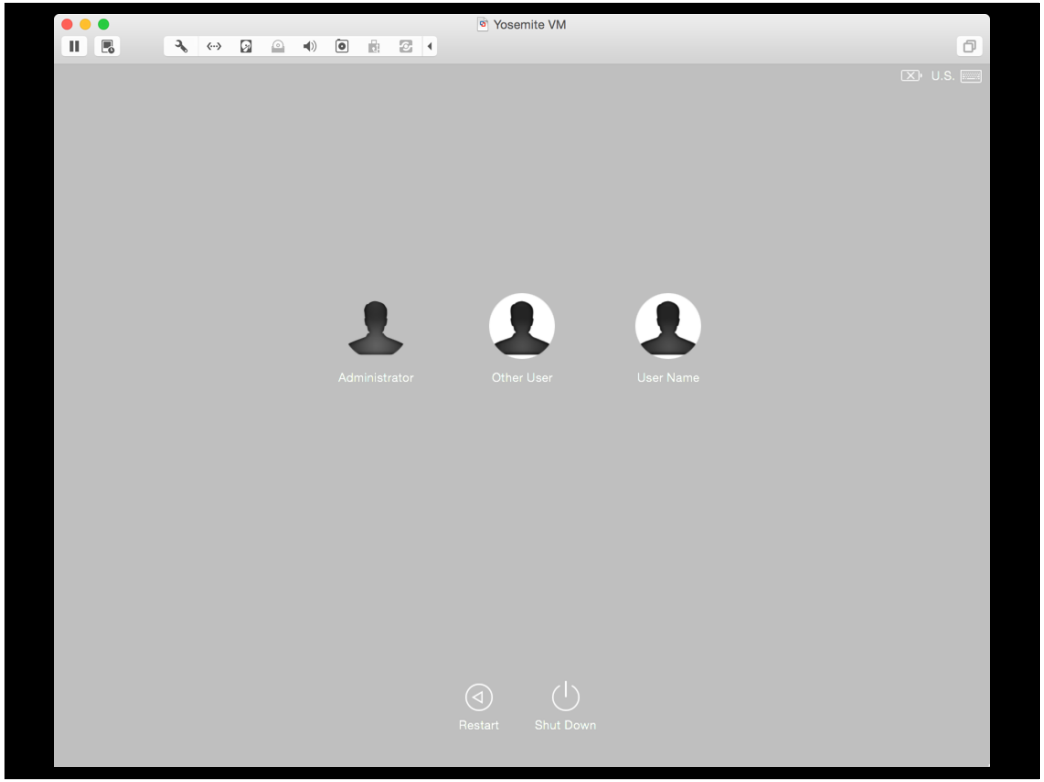


Fusion can also be used to start a remote console session for ESXi-hosted VMs.

Testing with Virtual Machines

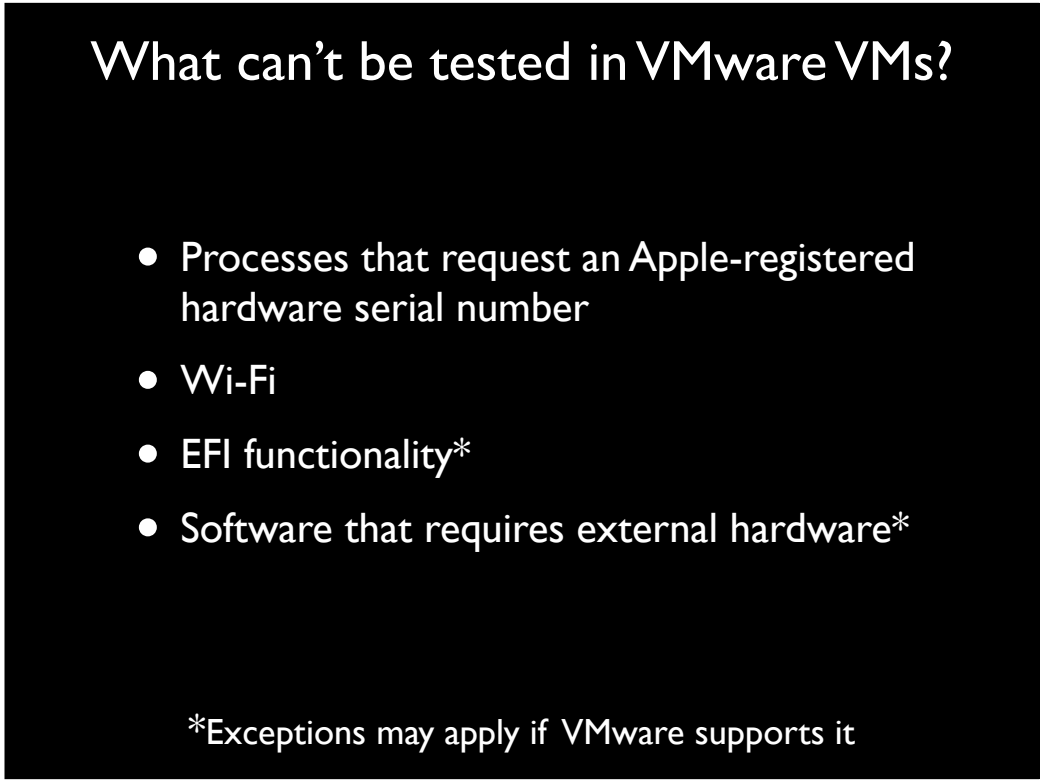
- What can be tested?
  - Software deployment
  - OS upgrades
  - Profiles
  - FileVault 2
  - NetBoot sets
  - Much more!

Once you have your virtual machines stood up, what can you test? It may be more accurate to say, what can't you test?



One area when I use VMs a lot is with FileVault 2 testing. This has been handy to me in particular because I can snapshot a VM and capture it in an unencrypted state before proceeding to encrypt the VM. If the testing doesn't go like I expected, I can roll back to the snapshot and have an instantly decrypted VM that's ready for the next round of testing.

This is my favorite thing that I found that I could test in a VM.



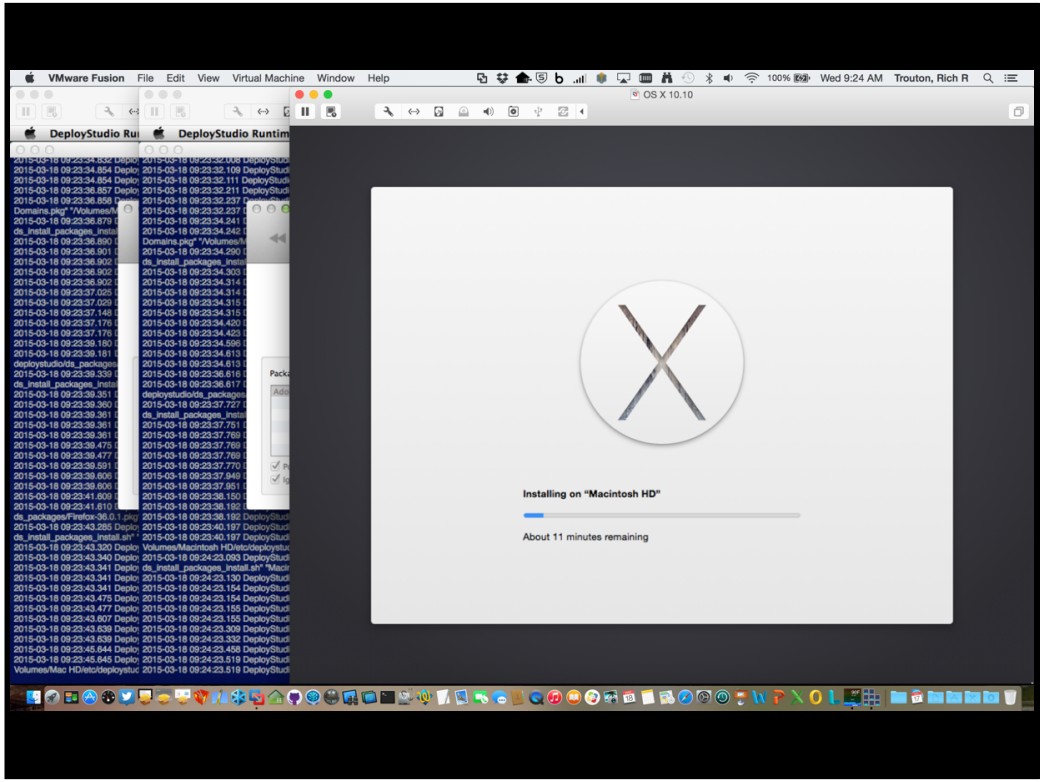
VMs do have some limitations because they are software constructs and not actual hardware. Here's what I've found can't be tested in a virtualized environment.

Anything involving having a Apple-registered hardware serial number / sending hardware serial number back to Apple - This includes iCloud services like Find My Mac and Messages. It also applies to getting hardware-specific OS installers via Recovery HD.

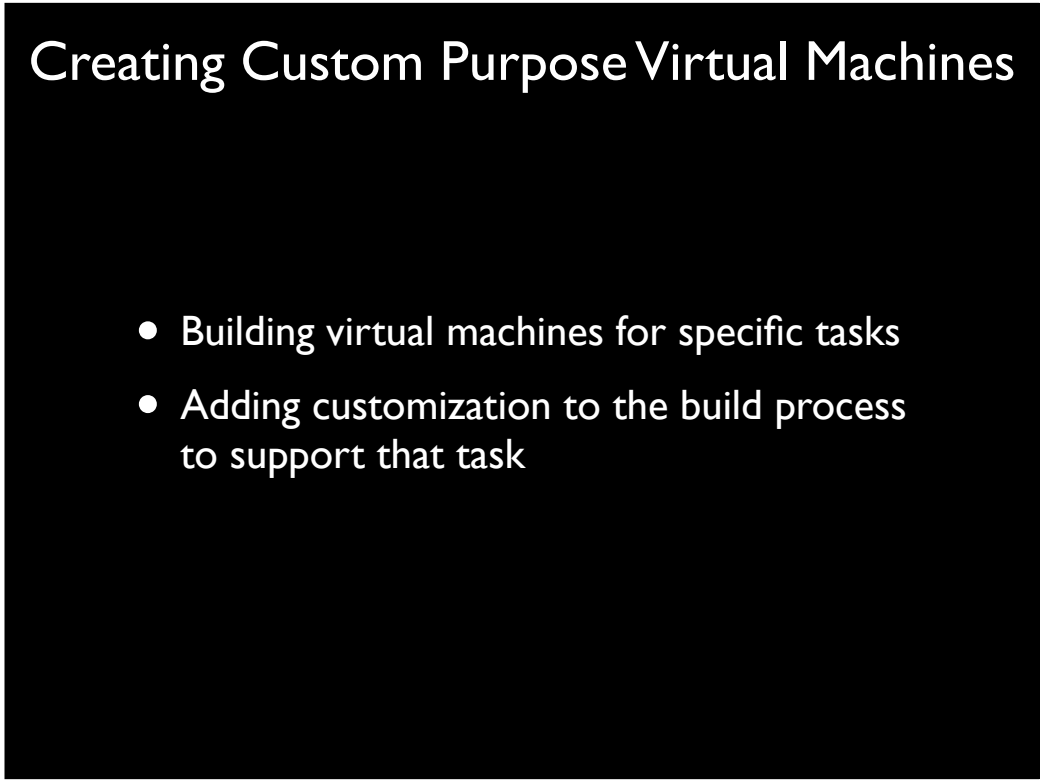
Most things involving EFI - Functions like Apple Internet Recovery or holding down the Option key to get a list of bootable volumes will not work. However, some things involving EFI work specifically because VMware made them work. For example, both NetBoot and FileVault 2 work fine in a VMware VM.

Wireless connections - Your VM doesn't have a WiFi card, though it may talk to your network via your Mac's WiFi connection. You can test in a VM to make sure that your WiFi settings apply, you can't test to verify that they work.

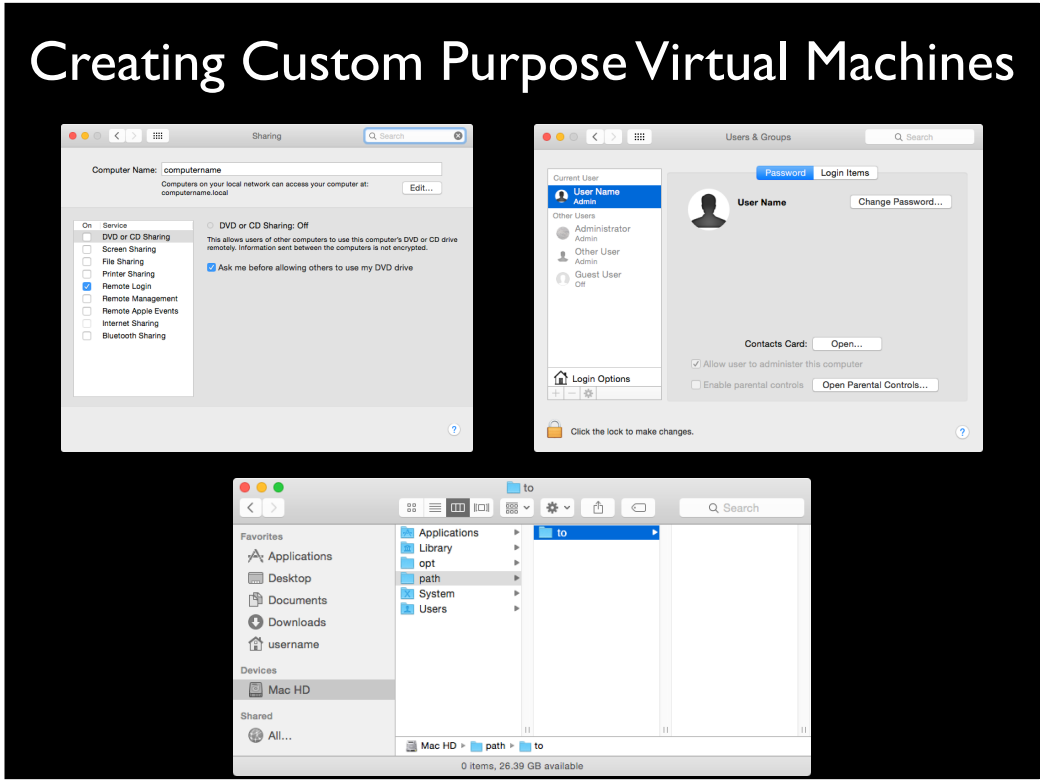
As Mac admins, we're always against the clock for testing. In my own environment, I've become reliant on virtual machines to speed up my development and testing cycle while reducing the physical footprint of my test environment. Instead of needing multiple machines to test changes to my deployment workflows, my testing now takes place almost exclusively on a quad-core Retina MacBook Pro with 16 GBs of RAM.



Likewise, having NetBoot and DeployStudio available to build test VMs means that I can be testing multiple workflows simultaneously on the same laptop. If a particular build hits a problem, I can discard that VM, fix the problem, then quickly create a new VM with the updated build.

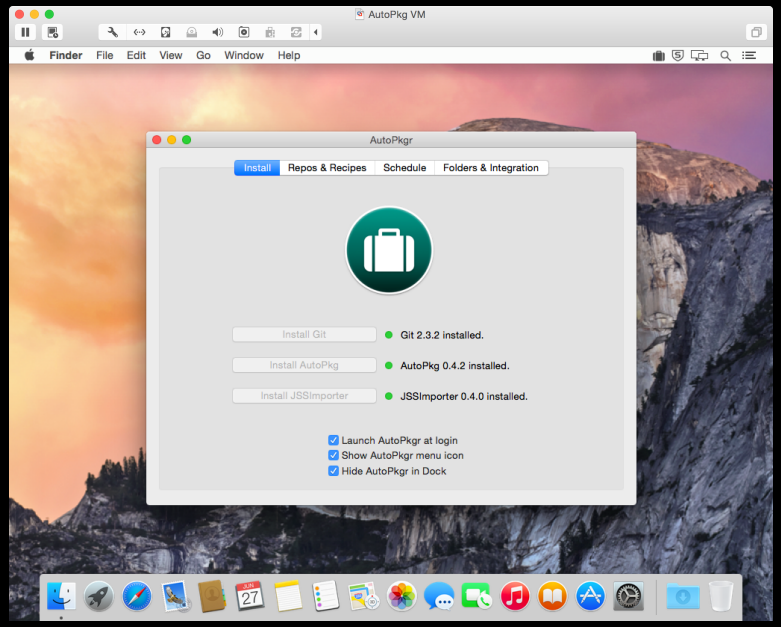


In addition to testing your Macs, another use for VMs is building ones for specific purposes. In this case, you would be building a VM to support one or more specific tasks.

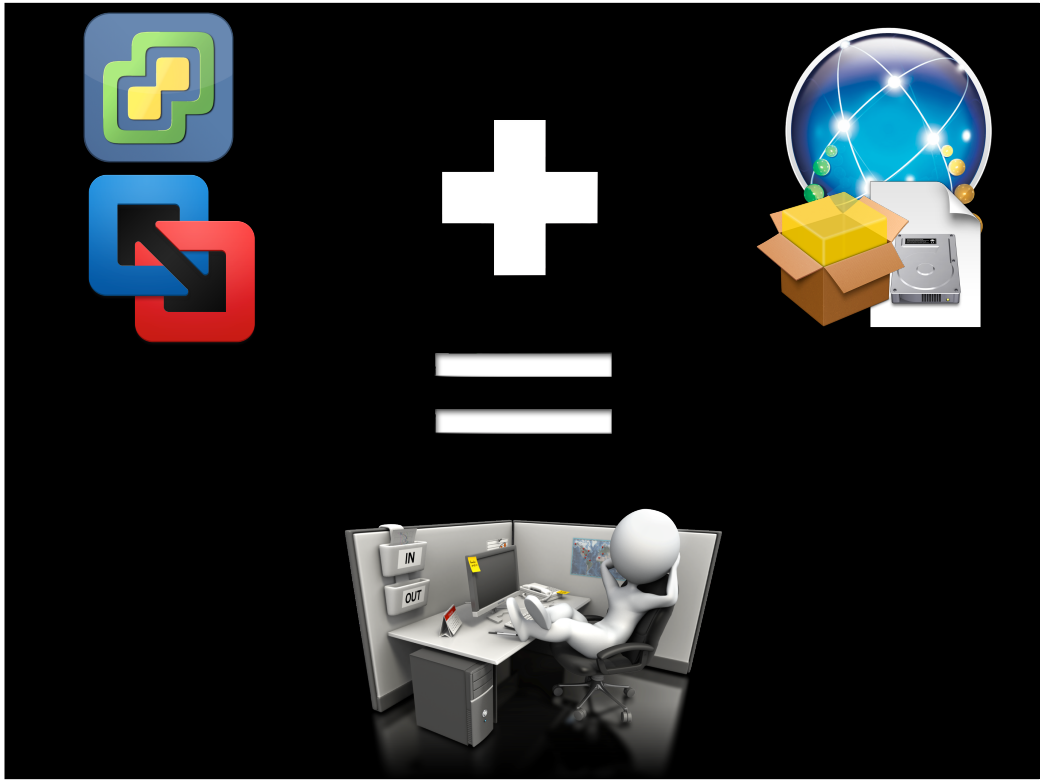


For example, I write a lot of documentation and it's been useful to me to have VMs readily available with certain characteristics. To support this, I have workflows which build VMs to have specific hostnames, user accounts with generic names and custom directories. This gives me a standard environment when writing documentation, where everything is set up with the generic names and locations I prefer to use. Among other things, this greatly speeds up my ability to take screenshots and screencapture movies for use with my documentation.

Creating Custom Purpose Virtual Machines



Another way I've been leveraging custom purpose VMs is using them to host and run tools like AutoPkg and AutoPkgr. These are open-source tools I use to download new software updates from vendors and upload them to my system management tool. I can build a VM on an ESXi server and have the VM dedicated to running this particular job full time, rather than tying up a physical Mac with the same job. In the event of a problem, being able to snapshot or clone the VM allows me to back it up quickly before troubleshooting the issue.



To sum up, virtual machines plus automated build processes equals more time for you.

Virtual machines running OS X can be resource hogs, as you will need to assign at least two processors as well as 3 - 4GBs of RAM to have them run at usable speeds. That said, the time and resource savings realized by using virtual machines instead of actual hardware should help make the case for investing in one or two speedy Macs running virtual machines instead of a multitude of actual test machines.

Useful Links

- Upgrading an ESXi server from 5.5 to 6.0: https://derflounder.wordpress.com/2015/03/25/upgrading-an-esxi-server-from-5-5-to-6-0/
- Setting up ESXi 6.0 on a 2012 Mac Mini Server: https://derflounder.wordpress.com/2015/03/24/setting-up-esxi-6-0-on-a-2012-mac-mini-server/
- Managing ESXi-hosted virtual machines using VMware Fusion Professional: http://derflounder.wordpress.com/2014/11/23/managing-esxi-hosted-virtual-machines-using-vmware-fusion-professional/
- Emulating specific Apple models in VMWare Fusion VMs: http://derflounder.wordpress.com/2013/02/19/emulating-specific-apple-models-in-vmware-fusion-vms/

Useful Links

- Apple Mac Pro 6,1 (black) officially supported on ESXi 5.5 Patch03: http://www.virtuallyghetto.com/2014/10/apple-mac-pro-61-black-officially-supported-on-esxi-5-5-patch03.html
- ESXi 6.0 support for Mac Mini: http://www.virtuallyghetto.com/2015/02/esxi-6-0-works-ootb-for-apple-mac-mini-mac-pro.html
- How to run Nested Mac OS X guest on ESXi VM on top VMware Fusion?: http://www.virtuallyghetto.com/2014/08/how-to-run-nested-mac-os-x-guest-on-nested-esxi-on-top-vmware-fusion.html
- How to change hardware serial number for Mac OS X Guest: http://www.virtuallyghetto.com/2013/10/how-to-change-hardware-serial-number.html

Useful Links

- Creating customized OS X installer disk images for VMware Fusion: http://derflounder.wordpress.com/2013/08/02/creating-customized-os-x-installer-disk-images-for-vmware-fusion/
- VMware custom OS X installer script adds support for VMware ESXi: http://derflounder.wordpress.com/2013/08/10/vmware-custom-os-x-installer-script-adds-support-for-vmware-esxi/
- create_vmware_osx_install_dmg script updated with Mavericks support: http://derflounder.wordpress.com/2013/10/23/create_vmware_osx_install_dmg-script-updated-with-mavericks-support/
- Using VMware's Standalone Remote Console for OS X with free ESXi: https://derflounder.wordpress.com/2015/04/18/using-vmwares-standalone-remote-console-for-os-x-with-free-esxi/

Useful Links

- Automatically Partition Virtual Hard Drives when Netbooting OS X Images from ESXi 5.5: http://cobbservations.wordpress.com/2013/10/15/automatically-partition-virtual-hard-drives-when-netbooting-os-x-images-from-esxi-5-5/
- Building OS X Test Environments in VMWare Fusion w/ System Image Utility: http://cobbservations.wordpress.com/2013/06/11/92/
- First Boot Package Install Generator.app: http://derflounder.wordpress.com/2014/10/19/first-boot-package-install-generator-app/
- createOSXinstallPkg: https://github.com/munki/createOSXinstallPkg

Downloads

PDF available from the following link:

http://tinyurl.com/PSU2015vmPDF

Keynote slides available from the following link:

http://tinyurl.com/PSU2015vmKeynote