

HOWTO: Run a Virtual Machine at Boot

Disclaimer: This is a personal document and is not official or endorsed by VMware. Feedback, suggestions, and edits are welcome.

This document is intended for someone who wants to run a VMware Fusion virtual machine at boot, and assumes basic familiarity with both OS X and Fusion, as well as [A Beginner's Guide to VMware Fusion](#).

If you want to be notified of changes and additions to this document, you can use the "Receive email notifications" action in the sidebar on the left. Please use the comments below only for things *specific to this document* (e.g. inaccuracies); general questions (including questions about getting this to work) are better off in the [discussion section](#).

Motivation

Unlike Windows or Linux, where you can choose from Workstation/Server/Player/etc., on the Mac, Fusion is currently our only product. While Fusion is intended to be a consumer product, it shares the common VMware code base, and so gets a bunch of features for free. With a little bit of tweaking, you can run a virtual machine at boot and in the background. For example, you might want to do this on a server, or with a virtual machine that provides services to users on the Mac.

Prepare the Virtual Machine

This section is mostly optional, but makes it easier to interact with your virtual machine. Add the following lines to the .vmx config file:

```
msg.autoAnswer =
"TRUE"
signal.suspendOnHUP =
"TRUE"
signal.powerOffOnTERM =
"TRUE"
RemoteDisplay.vnc.enabled =
"TRUE"
RemoteDisplay.vnc.port =
"5902"
```

- **msg.autoAnswer** is because sometimes Fusion wants to prompt you with some information (e.g. Tools aren't installed). Without this, the virtual machine will sit around waiting for an answer, but we don't want that since the idea is to run automatically.
- **signal.suspendOnHUP** and **signal.powerOffOnTERM** give us ways to suspend or shut down the virtual machine by sending signals (via `kill`, e.g. `kill -HUP \$PID` where \$PID is the ID of the vmware-vmx process).
- **RemoteDisplay.vnc.enabled** and **RemoteDisplay.vnc.port** give us a way to connect to the virtual machine. Note that by default, Fusion virtual machines already come with RemoteDisplay.vnc.port; edit this rather than creating a duplicate entry.

Test your virtual machine by running it normally in Fusion, connecting via VNC, and suspending and/or powering off.

Run at Boot

You can have the virtual machine automatically restart when stopped (shuts down, suspends, crashes, etc.) or run (automatically) only once per boot. You will need administrator access for this part. A graphical editor for launchd scripts which you might find useful is [Lingon](#).

I've tested this on a Mac mini running 10.5.4. 10.4.x has slightly different launchd syntax and defaults; you'll probably need to tweak the plists slightly. Note I'm *not* a launchd expert, this section is based on what I learned via Googling and playing with scripts. Improvements welcome.

With Fusion 2.0b2, you can use "vmrun start ..." instead of calling "vmware-vmx -x ..."; vmrun is meant for scripting and is probably the better way to go.

Run Continuously

This method will restart the virtual machine if it dies (shuts down, crashes, etc.). This makes the HUP signal not very useful, because immediately after the virtual machine suspends, launchd will respawn the virtual machine and it will resume again.

Create a plain text file in /Library/LaunchDaemons, let's call it com.example.fusion-as-server. Paste in the following:

```
<?xml version="1.0"
encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>Label</key>
  <string>com.example.fusion-as-server</string>
  <key>ProgramArguments</key>
  <array>
    <string>/Library/Application Support/VMware
Fusion/vmx/Contents/MacOS/vmware-vmx</string>
    <string>-x</string>
    <string>/Users/etung/Virtual
Machines/Test.vmwarevm/Test.vmx</string>
  </array>
  <key>RunAtLoad</key>
  <true/>
  <key>UserName</key>
  <string>etung</string>
  <key>KeepAlive</key>
  <true/>
</dict>
</plist>
```

Be sure to change the path in **ProgramArguments** to point at whatever virtual machine you want to run, and **UserName** to the owner of that virtual machine. You probably also want to give it a more descriptive name and **Label**.

In [Re: running OSX Server with Fusion as a daemon and OSX server as host?](#), [Dr. Wo](#) suggests not using KeepAlive with 2.0b2 because vmware.vmsg may not be found, causing launchd to keep respawning (or something).

Run Once

This method runs the virtual machine once at boot, but if the virtual machine stops (shuts down, crashes, suspends, etc.) it won't get restarted.

Running once is slightly more tricky than running all the time. The problem is that in order to run a virtual machine, various Fusion kexts need to be loaded. If we run the virtual machine before the kexts are loaded, vmware-vmx looks around, doesn't see anything to talk to, and gives up and dies. We can get away with it in the continuous case because launchd will keep respawning the virtual machine, and eventually it will succeed (after the Fusion kexts come up). In the run-once case, the lack of kexts means that the virtual machine doesn't run. We want to make sure the kexts are loaded before trying to run the virtual machine.

Unfortunately, launchd provides no way to order tasks. [Apple's recommendation](#) is to use IPC. However, from an end-user's point of view, this is not a practical possibility. Instead, let's try a hack.

When the kexts are loaded by boot.sh, various files are created. We'll make a script that waits for the creation of one of these files before starting the virtual machine. Let's call it test.sh, located in your home directory.

Note: I don't think you can use the launchd WatchPaths or QueueDirectories parameters instead of a script. First, they appear to take precedence over the LaunchOnlyOnce parameter, so it wouldn't do the right thing. According to [this blog post](#), WatchPaths only works if the file always exists (the files used by boot.sh don't, they're created and destroyed). QueueDirectories watches an entire directory; the vmnet-*vmnet*.pid files are in a directory with a bunch of other stuff, so are unsuitable; vmnet-dhcpd-vmnet*.leases changes whenever a DHCP client connects or disconnects (so stopping a virtual machine would trigger it to run again). If you wanted to use these, you could probably modify boot.sh to create a sentinel file you could watch for.

TODO: Should use kextstat to see if the Fusion kexts are loaded instead of the file test

```
#!/bin/bash
while [! -e /var/run/vmnet-bridge-vmnet0.pid]; do
    sleep 1
done

"/Library/Application Support/VMware
Fusion/vmx/Contents/MacOS/vmware-vmx" -x "/Users/etung/Virtual
Machines/Test.vmwarevm/Test.vmx"
```

Be sure to change the path to wherever your virtual machine actually is. chmod the script to be executable (e.g. ``chmod +x ~/test.sh``).

Create a plain text file in /Library/LaunchDaemons, let's call it com.example.fusion-as-server. Paste in the following:

```
<?xml version="1.0"
encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Label</key>
    <string>com.example.fusion-as-server</string>
    <key>ProgramArguments</key>
    <array>
        <string>/Users/etung/test.sh</string>
```

HOWTO: Run a Virtual Machine at Boot

```
</array>
<key>RunAtLoad</key>
<true/>
<key>ExitTimeOut</key>
<integer>0</integer>
<key>LaunchOnlyOnce</key>
<true/>
<key>UserName</key>
<string>etung</string>
</dict>
</plist>
```

Be sure to change **UserName** to the owner of that virtual machine, and the path in **ProgramArguments** to point at wherever the script actually is (you probably don't actually want to have it in your home directory). You probably also want to give it a more descriptive name and **Label**.