Overview: How Meteor watches your files for changes

When you run your Meteor app in local development mode, the `meteor` command-line tool watches your source code for changes. (This page describes the behavior as of Meteor 1.0.2.1.)

There are two different strategies that `meteor` uses for change watching: efficient system-specific APIs like `inotify` (on Linux) and `kqueue` (on OSX), and a slower but more reliable "stat polling" method. `inotify` / `kqueue` is less reliable because it does not work on every system, and sometimes `meteor` can't even tell if it doesn't work. Stat polling is slower in two ways: it only looks for changes once every "polling interval", and the actual check that it does consumes more CPU.

By default, `meteor` uses both `inotify` / `kqueue` and stat polling, with a polling interval of 5 seconds; if it detects that `inotify` / `kqueue` isn't working, it uses a polling interval of 0.5 seconds.

This means that on most systems, changes are detected very quickly by `inotify` / `kqueue`. On systems where `inotify` / `kqueue` doesn't work, the stat polling fallback still works, but it does use more CPU than if `inotify` / `kqueue` works; and if `meteor` was unable to automatically detect that `inotify` / `kqueue` didn't work, you may have to wait up to 5 seconds for stat polling to notice file changes.

The rest of this wiki page describes two ways in which you can improve the performance of Meteor file watching.

- There are some circumstances where `inotify` doesn't work (causing `meteor` to fall back to the CPU-hungry 0.5 second poll), but a simple configuration change will make `inotify` work.
- There are some circumstances where `inotify` / `kqueue` doesn't work but `meteor` is unable to detect this, so only the high-latency 5 second poll happens. While you can't make `inotify` / `kqueue` work in this case, you can at least tell `meteor` to use a faster poll to detect changes sooner.

Making `inotify` work on Linux systems where it doesn't work

(Note: this section is Linux-specific: it will not work on OS X, and the warning message below should not be displayed there. If you do see this warning on OS X, please file a bug!)

Linux users may see a message saying:

```
=> It looks like a simple tweak to your system's configuration will make many
   tools (including this Meteor command) more efficient. To learn more, see
   https://github.com/meteor/meteor/wiki/File-Change-Watcher-Efficiency
```

This message means that `meteor` tried to use the Linux `inotify` API to watch for file changes, but you've already reached the system-wide limit for the number of files that you can watch (probably in a different program entirely!).

Fortunately, this is easy to fix. Just run this command at your shell, entering your password when prompted:

```
echo fs.inotify.max_user_watches=524288 | sudo tee -a /etc/sysctl.conf && sudo sysctl
-p
```

(Alternatively, use another mechanism to edit `/etc/sysctl.conf` as root and add the line `fs.inotify.max_user_watches=524288` to the bottom, then run `sudo sysctl -p`.)

After running these commands, you should no longer see the "simple tweak" message when running `meteor`. (If you still do, file a bug!) Meteor is now using the efficient `inotify` API

If you do not have superuser access on your computer, you won't be able to run these commands. In that case, you may set the `METEOR_WATCH_FORCE_POLLING` environment variable as described in the next section, which will hide the warning and not try to use `inotify` at all.

Running `meteor` on a network/shared filesystem

Some filesystems don't support `inotify` / `kqueue` at all, but also don't allow `meteor` to detect this error case. In this situation, `meteor` will still detect file changes with the stat poller at the 5 second polling interval, but you may want to poll more often in order to detect changes faster.

The most common filesystems with this issue are network or shared filesystems like NFS or VirtualBox shared folders (vboxsf). For example, if you are using Vagrant or a similar tool to run `meteor` inside a virtual machine, and you are sharing your app source directory between your host machine and the VM, `inotify` may be unable to detect changes made on your host machine. The symptom will be that `meteor` doesn't even start trying to rebuild your app until up to 5 seconds after you make the change.

To improve this situation, on the machine where you are running `meteor` (eg, inside your VM), set the `METEOR_WATCH_FORCE_POLLING` environment variable to any non-empty value. For example, put the line `METEOR_WATCH_FORCE_POLLING=true` into the `.bashrc` file in your home directory.

If you would like finer control over the polling interval (to put it somewhere between the options of 5 seconds and 500 ms), set the `METEOR_WATCH_POLLING_INTERVAL_MS` environment variable to a number.