Remote TextMate Development via SSH and Rsync

2009

2 min read • 452 words

I am a huge fan of TextMate. In my opinion, it is by far the greatest text editor ever conceived by mankind

> TextMate defined the modern code editor experience with innovations like snippets, bundles, and fuzzy file searching—features now standard in editors like VS Code and Sublime Text. This 2009 perspective captures the pre-cloud development era when local tooling was paramount.

. It has a couple of shortcomings, however. One of which is that it has no built-in FTP or SFTP support. Remote file editing is a bit of a bear here if you like to view folders in the project drawer on the side.

Options for remote editing with TextMate:

- Cyberduck FTP client
- MacFUSE + SSHFS
- Rsync + SSH

Cyberduck

The Cyberduck option is very useful. While in the FTP client, you simply click "Edit in TextMate" and the client will download the file for you, open it in your editor, and – here's the awesome part – it automatically uploads the file every time you save it

> This save-and-sync workflow predates modern solutions like Live Share or remote development containers, showcasing the ingenuity required for remote development before cloud-native tooling.

. This works great when working with one file at a time. The drawback, however, is when working with large projects. Toggling between many files can be an albatross without the project drawer.

MacFUSE + SSHFS

MacFUSE + SSHFS works great, and allows you to mount an SSH folder as a mountpoint on your local system. You can open this folder with TextMate

> FUSE (Filesystem in Userspace) represents an elegant solution to remote development—making remote directories appear local. The performance issues described here highlight the network latency challenges that would later drive the development of more sophisticated remote development solutions.

. This is perfect for smaller projects. However, with larger projects, this makes opening the folder in TextMate almost unbearable as it checks the status of every single file.

Rsync + SSH

So here's the final solution: Rsync + SSH. This allows me to automatically sync my working copy with my server and allow for snappy file interactions without having insane latencies

> Rsync's differential sync algorithm—transferring only changed portions of files—made it ideal for remote development workflows. This approach of maintaining local copies while syncing changes presaged modern distributed version control systems like Git.

!

To remotely sync over SSH, run the following code:

```
rsync -avz -e ssh remoteuser@remotehost:/remote/dir /target/dir/
```

Hint: If your remote working copy is a subversion checkout, you can add `--cvs-exclude` into the rsync parameters, and it will exclude the ".svn" folders!

You can then open this directory in TextMate and make all the changes you want, and then sync after! There is also a wonderful TextMate Bundle for Remote Rsync + SSH within TextMate.
