Watcher

The error message you're encountering in Visual Studio Code, "Visual Studio Code is unable to watch for file changes in this large workspace (error ENOSPC)," typically indicates that the system has run out of "user watches." This is because both Linux and VS Code have limits on how many file system watches they can have open at a time, and if you're working with a large workspace, you can easily exceed this limit.

This limit is controlled by the `fs.inotify.max_user_watches` setting in Linux. When this limit is not high enough to handle the number of files VS Code is trying to watch, you'll see the ENOSPC error, which stands for "Error NO SPaCe" on the device for user watches.

To resolve this issue, you can increase the number of inotify watches allowed by the system. You can do this by running the following command:

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

This command sets the max_user_watches to a higher number (524288) and then reloads the sysctl configuration with `sudo sysctl -p`.

Regarding memory usage, VS Code extensions like Pylance can indeed consume a significant amount of memory, especially if they are analyzing a large codebase. The memory usage can also increase if multiple extensions are running simultaneously or if the workspace is very large.

The file watcher used by VS Code is separate from `esbuild`'s watcher, but both are trying to monitor changes to the filesystem, which can lead to high memory usage if they are both active on a large number of files. If you're running VS Code on the same system where your development server is running, and both are watching a large workspace, this could contribute to high memory usage.

To mitigate the issue, you could:

1. Increase the system's file watch limit as described above.
2. Close unnecessary files and folders in VS Code to reduce the workspace size.
3. Disable or configure extensions to limit their memory usage. For example, you can configure Pylance to only analyze open files.
4. Ensure that your development server (like `esbuild`) is not watching the entire workspace if it's not necessary.

If you're running VS Code on a server or a resource-constrained environment, you might want to consider using it on your local machine and pushing changes to the server, rather than directly working on the server. This would help to keep the server's resources dedicated to serving the application rather than being used up by the development environment.