Bash - Add Symbolic Links Subdirectories

To efficiently add these symbolic links to all subdirectories that are missing them, you can use a `find` loop. This command will check each subdirectory and only create the link if a file or link named `.env` or `.secret` doesn't already exist.

The Command

Run this from the parent directory where your subdirectories are located:

Bash

```
find . -maxdepth 1 -type d ! -path . -exec sh -c '
    cd "$1"
    [ ! -e .env ] && ln -s /mnt/volume1/docker/shared.env .env
    [ ! -e .secret ] && ln -s /mnt/volume1/docker/secret.env .secret
' _ {} \;
```

Breakdown of the Logic

- **`find . -maxdepth 1 -type d ! -path .`**: This finds all directories exactly one level down from your current location, excluding the current directory (`.`) itself.
- **`-exec sh -c '...' _ {} \;`**: This executes a small shell script for every directory found.
- **`[ ! -e .env ]`**: This is a "logical gate." It checks if the file `.env` **does not** exist. If it's missing, it proceeds to the `&&` command to create the link.
- **`ln -s [target] [link_name]`**: Creates the symbolic link.

A Safer Alternative (Dry Run)

If you want to see which directories will be affected before actually making changes, you can use this "echo" version:

Bash

```
find . -maxdepth 1 -type d ! -path . -exec sh -c '
    cd "$1"
    [ ! -e .env ] && echo "Linking .env in $1"
    [ ! -e .secret ] && echo "Linking .secret in $1"
' _ {} \;
```

Quick Note on Relative vs. Absolute Paths

Since you are using **absolute paths** for your targets (`/mnt/volume1/...`), these links will work regardless of which subdirectory they live in. If you ever move the `docker` folder, you'll need to update the links, but for a fixed mount point like yours, this is the most reliable method.

Add Folder Exceptions

To exclude specific subdirectories, you can use the `! -name` flag (for simple folder names) or `-prune` (for more complex paths).

Here are the two best ways to do it, depending on how many folders you need to skip.

Option 1: Excluding by Folder Name (Simple)

If you just want to skip a few specific folders by name (e.g., `backup` and `logs`), use this version. The `! -name "..."` flags tell `find` to ignore those specific directories.

Bash

```
find . -maxdepth 1 -type d ! -path . ! -name "backup" ! -name "logs" -exec sh -c '
    cd "$1"
    [ ! -e .env ] && ln -s /mnt/volume1/docker/shared.env .env
    [ ! -e .secret ] && ln -s /mnt/volume1/docker/secret.env .secret
' _ {} \;
```

Option 2: Excluding a List (Cleanest for many folders)

If you have a long list of exceptions, you can use a **regex** pattern. This is much easier to read and maintain. Replace `folder1|folder2|folder3` with the names of the directories you want to skip.

Bash

```
find . -maxdepth 1 -type d ! -path . -not -regextext ".*(folder1|folder2|folder3)$" -exec sh -c '
    cd "$1"
    [ ! -e .env ] && ln -s /mnt/volume1/docker/shared.env .env
    [ ! -e .secret ] && ln -s /mnt/volume1/docker/secret.env .secret
' _ {} \;
```

Pro-Tip: The "Dry Run"

I highly recommend running a "Dry Run" first to verify your exceptions are working correctly. This will print the folders it **would** have acted on without actually creating any links:

Bash

```
find . -maxdepth 1 -type d ! -path . ! -name "exclude_me" -exec echo "Processing: {}" \;
```

Key Flags Used

- **`! -name "folder"`**: Tells find: "Do NOT include items with this name."
- **`! -path .`**: Prevents the command from trying to create links in the current parent directory (which would fail anyway since `.env` likely already exists there).
- **`-maxdepth 1`**: Ensures the command only looks at the folders right in front of you, rather than digging into sub-sub-directories.

Revision #1
Created 11 April 2026 19:56:45 by Steve Ling
Updated 11 April 2026 19:58:34 by Steve Ling