Setup Project Skills — MCP Server

A locally-running **Model Context Protocol (MCP) server** that implements the `setup-project-skills` Copilot skill.

It auto-detects a project's technology stack and installs / refreshes the matching GitHub Copilot skill folders into `<project>/skills/`, then updates `.vscode/settings.json` so Copilot discovers them.

Tools Exposed

Tool name	Description
`detect_project_type`	Scans a directory and returns the detected technology type(s). Read-only.
`check_project_skills`	Dry-run: shows which skills would be added / are already present. No changes.
`setup_project_skills`	**Incremental setup** — adds new skill folders, leaves existing ones untouched.
`refresh_project_skills`	**Full refresh** — replaces all skill folders with the latest from source.

Detected Project Types

Type	Key indicators
`Blazor`	`.razor` files + `.csproj`
`AspNetCoreApi`	`.csproj` + `Program.cs` (no Razor / MAUI / Designer)
`MAUI`	`Microsoft.NET.Sdk.Maui` or `<UseMaui>true` in `.csproj`
`WinForms`	`.csproj` + `Program.cs` + `.designer.cs` / `.vb`
`Android`	`AndroidManifest.xml`, `.kt`, or `.java` files
`Frontend`	`package.json` + `.ts`/`.tsx`, or `package.json` without `.csproj`
`CppCMake`	`.cpp`, `.vcxproj`, or `CMakeLists.txt`
`Unknown`	None of the above (common skills only are installed)

A single project can match multiple types (e.g. Blazor always also gets Frontend skills).

Skill Source Resolution

The server searches these paths (in order) for a local clone of `awesome-copilot`:

1. `G:\Repos\frontlook-admin\AI_HELPERS\awesome-copilot`
2. `G:\Repos\frontlook-admin\awesome-copilot`
3. `%USERPROFILE%\repos\awesome-copilot`
4. `%USERPROFILE%\awesome-copilot`
5. `C:\src\awesome-copilot`
6. **Auto-clone** from `https://github.com/frontlook-admin/awesome-copilot`

You can override with the `skillSourcePath` parameter on any tool.

Prerequisites

- .NET 9 SDK — for running locally
- Docker — for container usage
- VS Code with GitHub Copilot extension (for MCP discovery)
- `git` in PATH (needed only if the awesome-copilot repo must be auto-cloned)

Running Locally (dotnet)

```
cd AgentWorkflowBuilderMcp
dotnet run
```

The server speaks the MCP stdio protocol on stdin/stdout.

Docker

Build locally

```
docker build -t setup-project-skills-mcp .
```

Run with Docker (mount your project directory)

```
docker run --rm -i \
  -v /path/to/awesome-copilot:/opt/awesome-copilot:ro \
  setup-project-skills-mcp
```

Pass `skillSourcePath` as a tool argument pointing to `/opt/awesome-copilot` inside the container.

Pull from GitHub Container Registry

After publishing (see below), pull with:

```
docker pull ghcr.io/<your-github-username>/skillmcp:latest
```

Publishing to GitHub

1. Create a GitHub repository

```
git init
git add .
git commit -m "feat: initial MCP server for setup-project-skills"
git remote add origin https://github.com/<your-username>/<your-repo>.git
git push -u origin main
```

2. Automatic Docker image (GitHub Actions)

The workflow at `.github/workflows/docker-publish.yml` builds and pushes a Docker image to **GitHub Container Registry (GHCR)** automatically:

Trigger	Published tag
Push to `main`	`latest`, `main`, `sha-<short>`
Tag `v1.2.3`	`1.2.3`, `1.2`, `latest`
Pull request	Build only (not pushed)

No secrets need to be configured manually — the workflow uses the built-in `GITHUB_TOKEN`.

VS Code MCP Configuration

The `.vscode/mcp.json` file in this repo registers the server automatically.
Open the workspace in VS Code and GitHub Copilot will discover and use the tools.

Output: What Gets Written

After `setup_project_skills` or `refresh_project_skills` runs:

```
<project>/
  skills/
    git-commit/
    conventional-commit/
    csharp-async/          ← type-specific skills
    ... etc.
    skills.json            ← manifest with detected type + installed skills
```

```
    .vscode/
      settings.json          ← chat.promptFilesLocations entry added
```