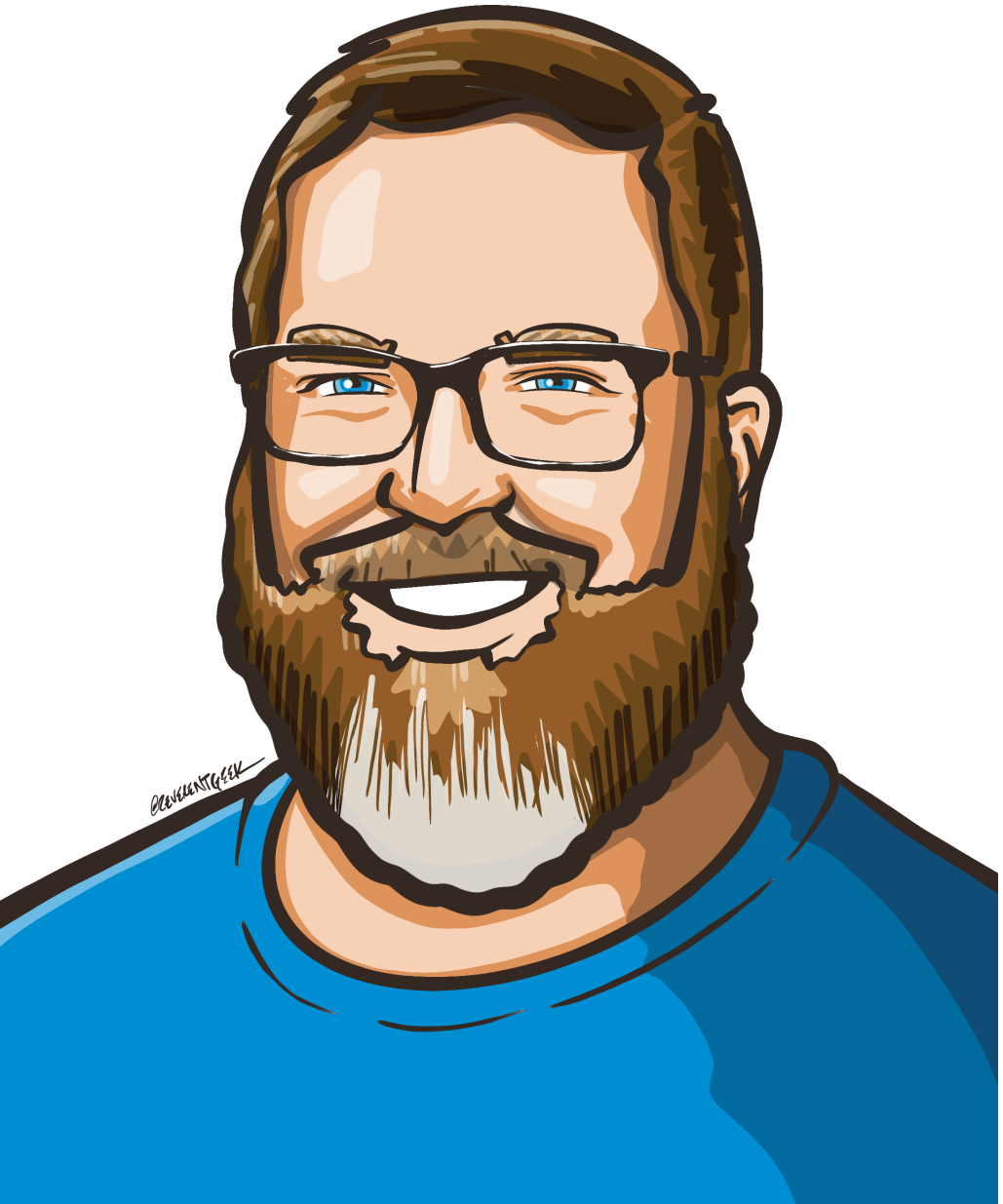
The Power of Dev Containers and GitHub Codespaces

From Local Dev to Cloud Ready

Chris Ayers

https://github.com/codebytes/dev-containers



Chris Ayers

Principal Software Engineer
Azure EngOps AzRel
Microsoft

BlueSky: @chris-ayers.com

LinkedIn: - chris-l-ayers

Blog: https://chris-ayers.com/

GitHub: Codebytes

Mastodon: @Chrisayers@hachyderm.io

~~Twitter: @Chris_L_Ayers~~

Agenda

- Why Dev Containers
- What They Are & How They Work
- Quick Start & Building
- Templates vs Features vs Customizations
- Core Concepts (Image vs Feature vs Template)
- GitHub Codespaces
- Demo
- Advanced: Security, Multi-Service, Prebuilds, Performance & Cost, Debug & Ports, Troubleshooting, Limits
- Resources & Q&A

Why Dev Containers?

- Onboard new contributors quickly
- Consistent tooling and versions for all developers
- Reduce system conflicts and "works on my machine" issues
- Secure, isolated environments for development
- Easy startup tasks and reproducible builds

What Are Dev Containers?

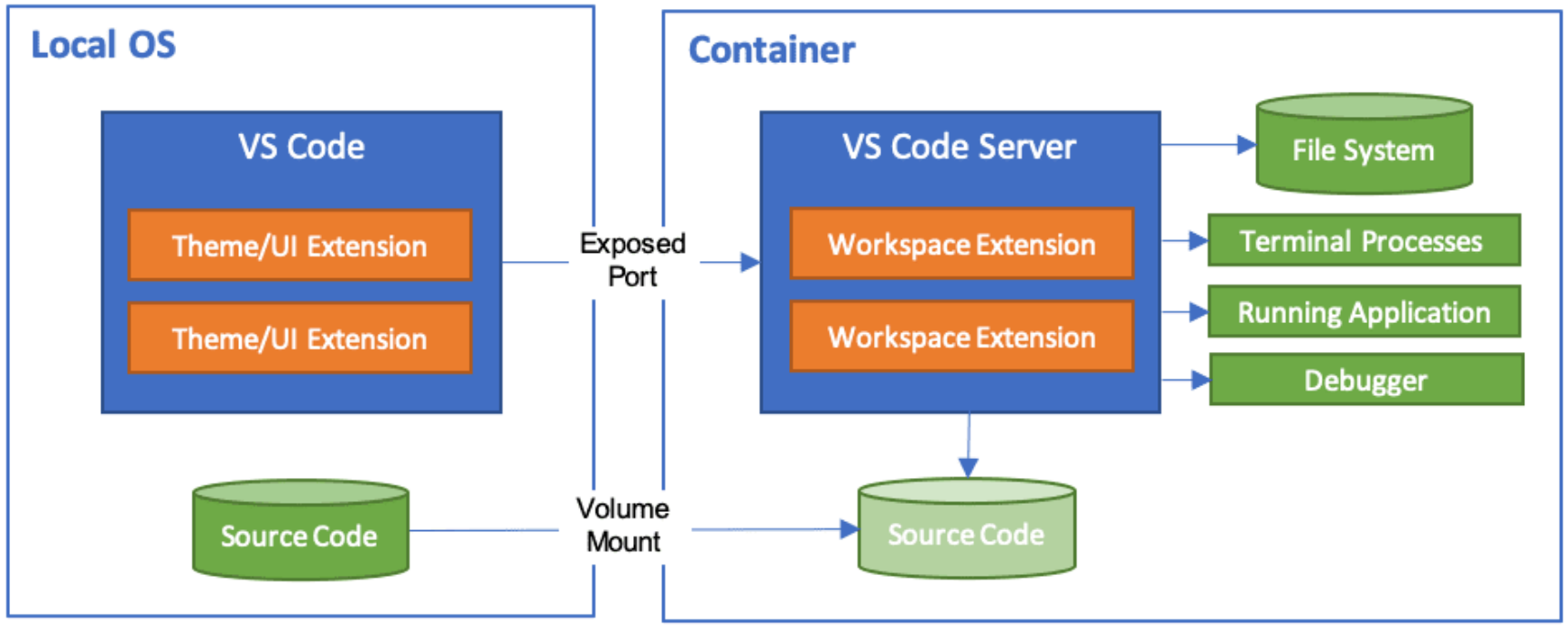
- Containerized environments for development
- Let you open any folder in VS Code with all dependencies ready
- Run apps, tools, or runtimes needed for a project
- Based on the open containers.dev specification

Prerequisites

- VS Code (Dev Containers extension, Codespaces extension)
- Docker (local or remote)
- GitHub account (for Codespaces)
- Optionally: IntelliJ IDEA, other containers CLIs like podman

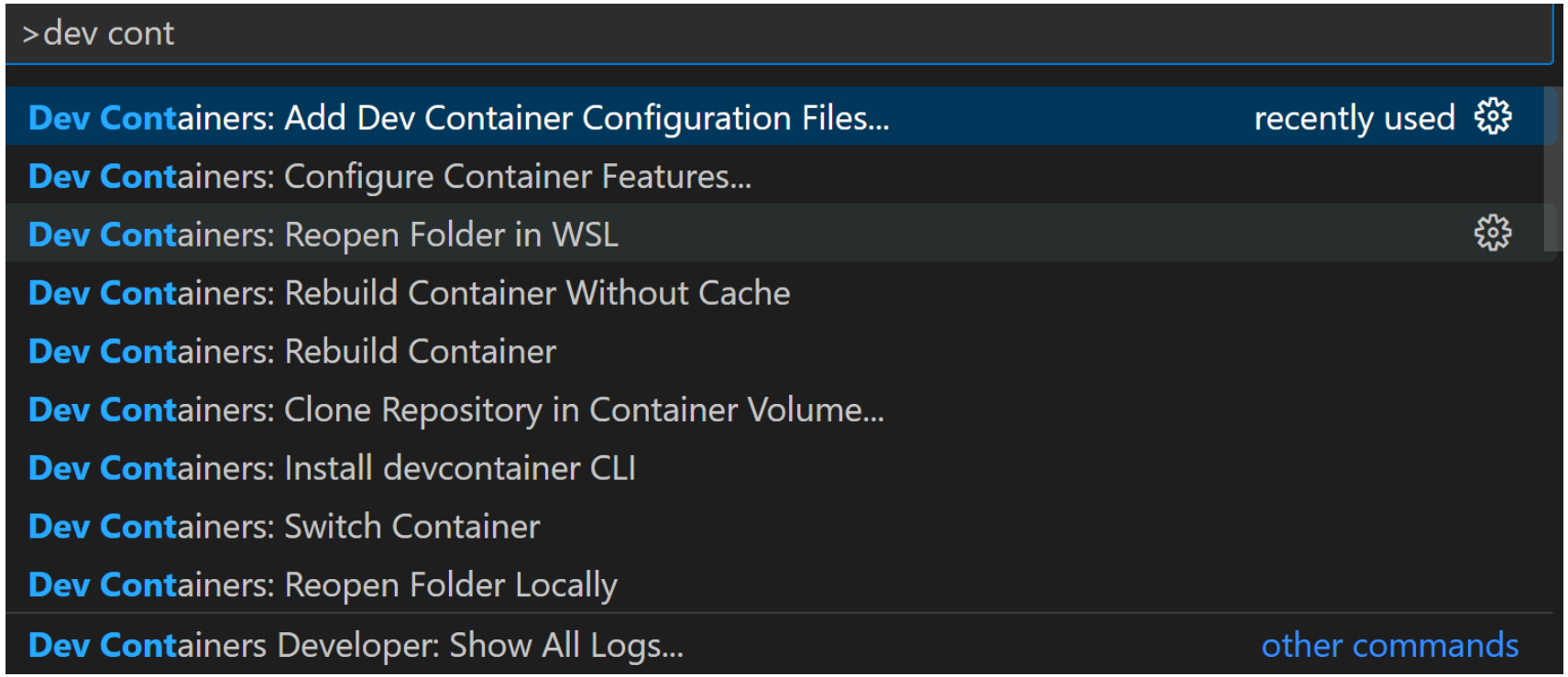
How Dev Containers Work

Your editor talks to a containerized environment with all dependencies, keeping your local system clean.



Building a Dev Container

From the Command Palette:



Templates, Features, Customizations

- Templates: project bootstrap (scaffold config)
- Features: add tools/runtimes (idempotent, versioned)
- Customizations: editor settings, extensions, startup tasks, port forwarding

Image vs Feature vs Template

Concept	When to Use	Change Frequency	Pinning Strategy
Base Image	Need different language/runtime base or OS	Low	Tag (e.g. :1-20-bookworm) or digest
Feature	Add/upgrade a CLI (gh, kubectl), language, or utility	Medium	Major tag (node:1) or minor for guarantees
Template	Starting a new repo / adding devcontainer from scratch	One-off	Template version (commit / release)

Principles & Practices

Principles:

- Keep Dockerfile minimal; use Features for add-ons
- Templates are for initial setup, not ongoing changes
- Prebuild images for slow, shared setup (language servers, SDKs)

Practices:

- Everyone needs it & rarely changes? → Bake into image/Feature
- Project-scoped & changes often? → postCreateCommand
- Personal? → Dotfiles/Settings Sync

Managing Extensions Deep Dive

- Add via customizations.vscode.extensions
- Opt-out: prefix with '-' to remove inherited extension
- Default extensions (user setting): dev.containers.defaultExtensions
- Force location (rare): remote.extensionKind overrides
- Keep lean: only language / tooling essentials; reduces startup & index time

Extensions Example

```
"customizations": {
  "vscode": {
    "extensions": [
      "ms-azuretools.vscode-docker",
      "-dbaeumer.vscode-eslint"  // opt-out inherited
    ]
  }
}
```

Port Forwarding

- Auto port detection: output parsing ("Listening on 3000") triggers forward suggestion
- forwardPorts: auto-forward on start; portsAttributes: label, onAutoForward, visibility
- Published vs Forwarded:
  - Forwarded: appears as localhost to app (ideal for dev auth flows)
  - Published: network-accessible (team demos, external callbacks)
- Browser preview & automatic HTTPS upgrade (if dev server supports)

Example portsAttributes

```
"portsAttributes": {
  "3000": { "label": "Web UI", "onAutoForward": "openBrowser" },
  "9229": { "label": "Node Inspector", "elevateIfNeeded": true }
}
```

Minimal devcontainer.json Example

```
{
  "image": "mcr.microsoft.com/devcontainers/typescript-node:1-20-bookworm",
  "features": {
    "ghcr.io/devcontainers/features/github-cli:1": {},
    "ghcr.io/devcontainers/features/node:1": { "version": "20" }
  },
  "forwardPorts": [3000],
  "customizations": {
    "vscode": {
      "extensions": ["dbaeumer.vscode-eslint", "ms-azuretools.vscode-docker"],
      "settings": { "editor.formatOnSave": true }
    }
  },
  "postCreateCommand": "npm ci",
  "remoteUser": "node"
}
```

Environment Variables in Dev Containers

Set environment variables in `devcontainer.json`:

```
{
  "containerEnv": { "MY_ENV_VAR": "value" },
  "remoteEnv": { "API_TOKEN": "${localEnv:API_TOKEN}" }
}
```

- `containerEnv`: for all processes in the container
- `remoteEnv`: for the VS Code server and extensions

Env Vars: Container vs Remote (Advanced)

- **containerEnv**: set at container start (Dockerfile, Compose, or devcontainer.json)
- **remoteEnv**: injected at attach time (user, secrets, paths)
- Use dynamic values: `${localEnv:VAR}` or `${containerWorkspaceFolder}`
- `userEnvProbe`: merge shell profile exports into remoteEnv ("loginShell", "interactiveShell", or "none")

Mounting Local Folders

You can mount local folders or files into your dev container using the `mounts` property:

```
{
  "mounts": [
    "source=${localWorkspaceFolder}/.npmrc,target=/root/.npmrc,type=bind,consistency=cached",
    "source=${env:HOME}/.ssh,target=/root/.ssh,type=bind,consistency=cached"
  ]
}
```

This is useful for sharing config files, SSH keys, or other resources from your host.

Non-Root User Best Practice

Always run as a non-root user for security. Most official images set `remoteUser` to `vscode` or `devcontainer` by default.

To enforce non-root:

```
{
  "remoteUser": "vscode"
}
```

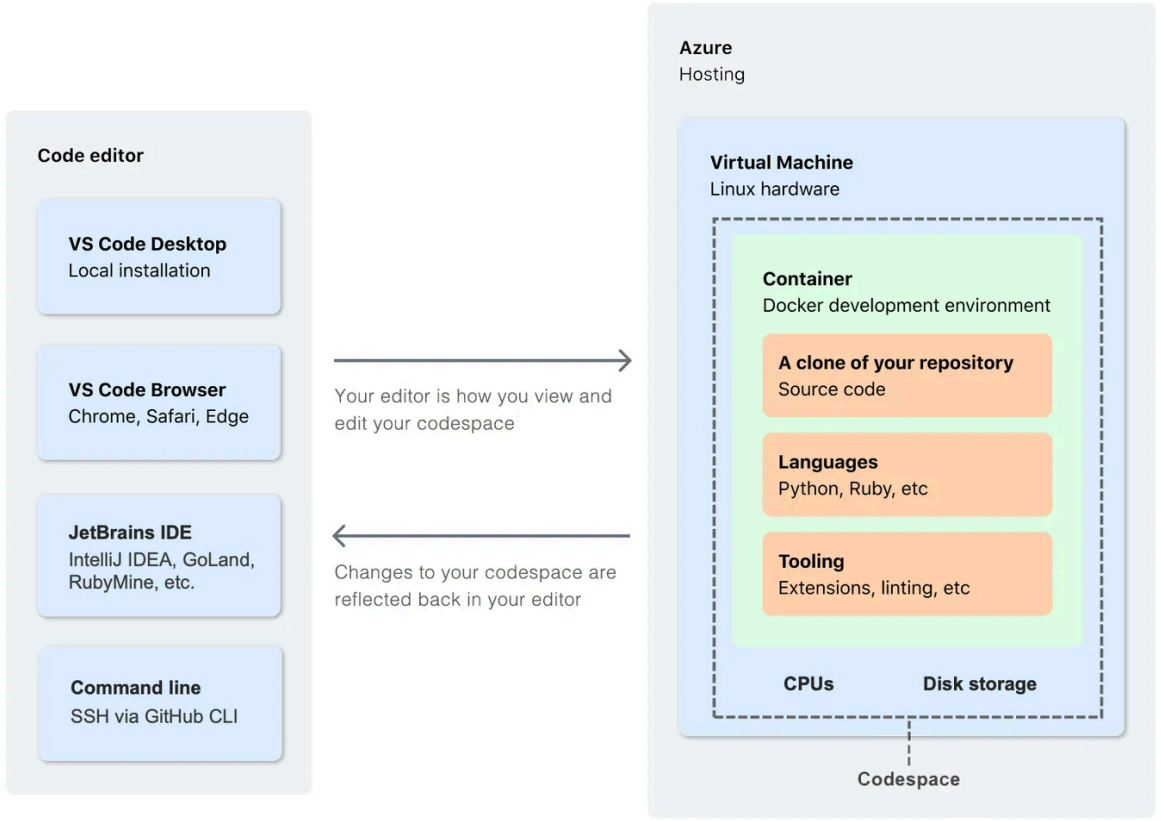
Or add the non-root-user Feature if needed.

More: Add a non-root user

GitHub Codespaces: What & Why

- Cloud-hosted, instant dev environments for any repo
- No local Docker or setup required
- Consistent, secure, and disposable workspaces
- Fast onboarding: prebuilds, dotfiles, and settings sync

GitHub Codespaces: Architecture



Codespaces: Key Features

- Start a dev environment from any branch, PR, or template
- Prebuilds: dependencies and tools ready before you connect
- Port forwarding, shared terminals, and Live Share for collaboration
- Personalization: dotfiles, settings sync, user secrets

Codespaces: How to Use

- Open any repo in a codespace from GitHub.com, VS Code, or CLI
- Choose machine size, region, and devcontainer config
- Suspend, resume, or delete codespaces as needed
- Works with VS Code (browser/desktop), JetBrains Gateway, and GitHub CLI

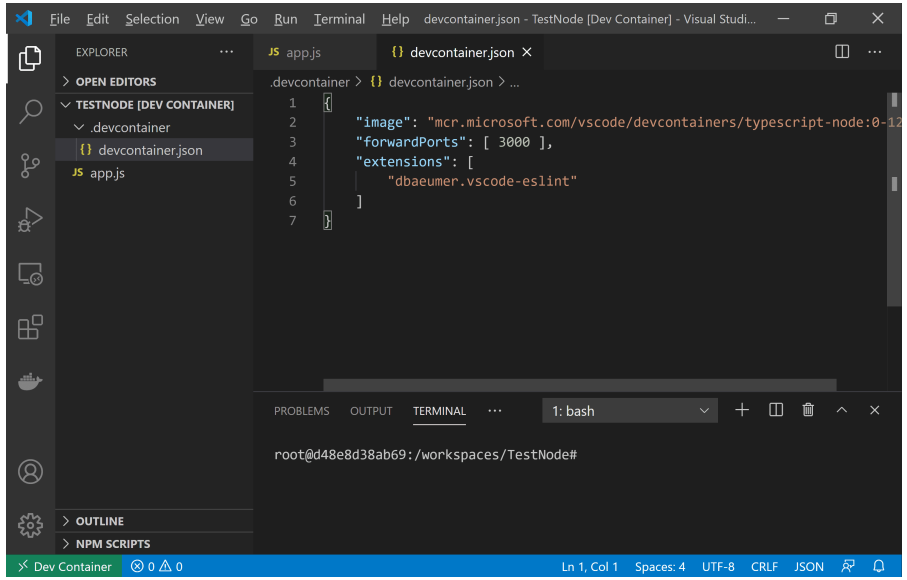
Codespaces: Limitations & Cost

- Linux-only (x86_64); no Windows containers
- Single container unless using Compose
- Bind mounts slower on macOS/Windows (prefer volumes)
- Pay for compute, storage, and prebuilds (beyond free quota)
- See docs.github.com/codespaces for details

GitHub Codespaces: Billing & Cost Control

- Pay for compute hours, storage, and prebuild usage (beyond free quota)
- Key levers: machine size, idle timeout, auto-delete unused codespaces
- Strategies: right-size, prebuild only critical branches, monitor usage/cost
- Always check current docs for pricing details

Demo Time


File Edit Selection View Go Run Terminal Help devcontainer.json - TestNode [Dev Container] - Visual Studi...
EXPLORER
app.js
devcontainer.json
OPEN EDITORS
TESTNODE [DEV CONTAINER]
.devcontainer
devcontainer.json
app.js
.devcontainer > devcontainer.json > ...
{
"image": "mcr.microsoft.com/vscode/devcontainers/typescript-node:0-12
"forwardPorts": [3000],
"extensions": [
"dbaeumer.vscode-eslint"
]
}
PROBLEMS OUTPUT TERMINAL
1: bash
root@d48e8d38ab69:/workspaces/TestNode#
OUTLINE
NPM SCRIPTS
Dev Container
0 0
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF JSON

Advanced Topics

Dev Container CLI

The `devcontainer` CLI lets you build, run, and automate dev containers from the terminal—no VS Code required.

- `devcontainer build --workspace-folder .` Build a dev container image
- `devcontainer up --workspace-folder .` Start a dev container
- `devcontainer exec --workspace-folder . bash -lc "npm test"` Run a command inside

github.com/devcontainers/cli

Prebuild & CI Integration Example

devcontainer.json (simplified):

```
{
  "image": "ghcr.io/your-org/your-prebuilt:latest",
  "postAttachCommand": "npm run dev"
}
```

Prebuild CI Workflow (GitHub Actions)

```
name: Prebuild Dev Container
on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]
```

Prebuild CI Workflow (continued)

```yaml
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: devcontainers/ci@v0.3
        with:
          imageName: ghcr.io/your-org/your-prebuilt
          push: true
          runCmd: |
            npm ci
            npm test --if-present
```

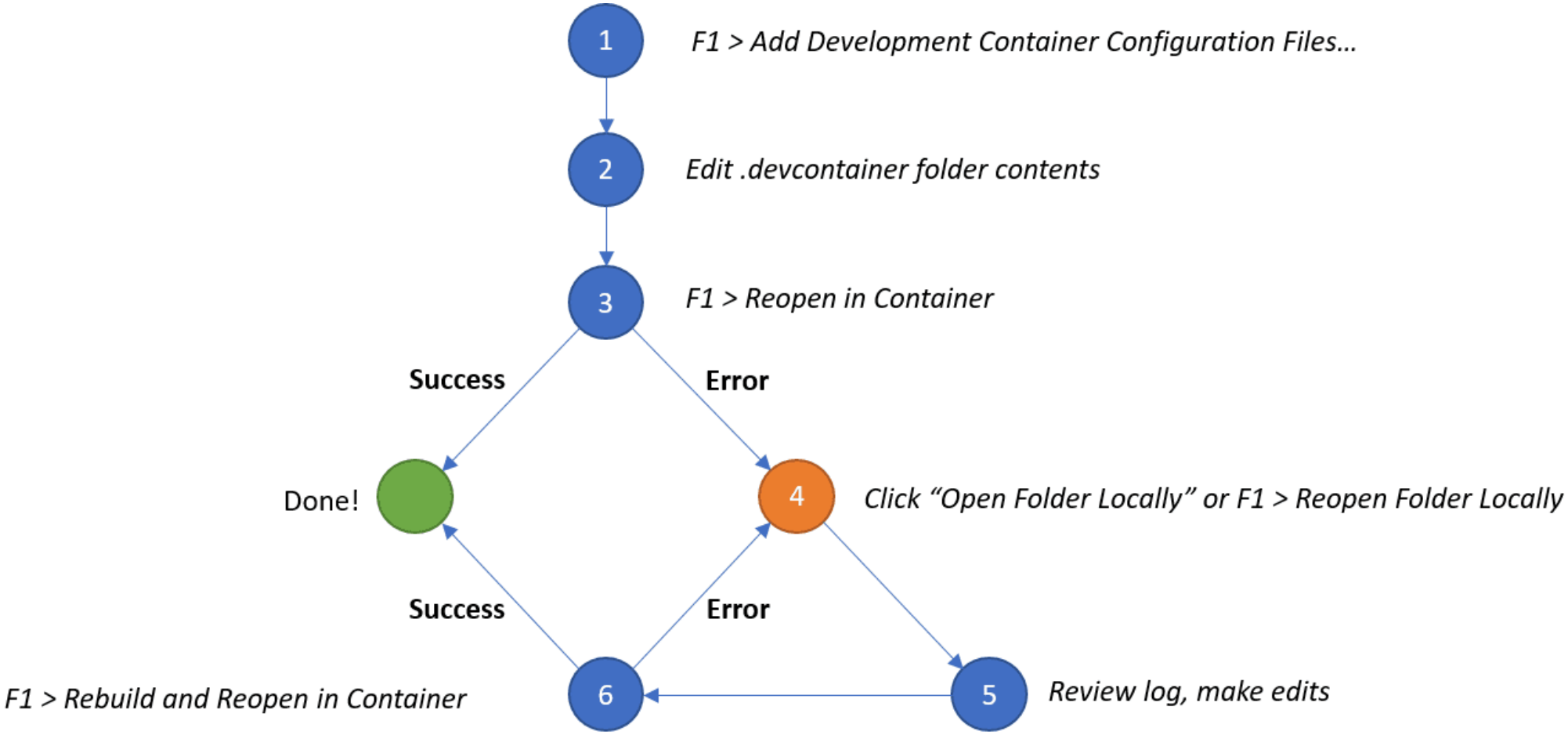
Tips & Tricks: Git Line Endings (Windows/WSL)

On Windows/WSL, you may see many modified files due to line ending differences between host and container.

- Add a `.gitattributes` file to enforce consistent line endings:

```
* text=auto eol=lf
*.{cmd,[cC][mM][dD]} text eol=crlf
*.{bat,[bB][aA][tT]} text eol=crlf
```

Troubleshooting & Recovery



Resources

Links

- Dev Container Templates
- Dev Container Features
- Dev Containers Tutorial
- Beginner's Series to Dev Containers
- devcontainer.json Reference
- Dev Container CLI
- Advanced Config (VS Code)

Chris Ayers

BlueSky: @chris-ayers.com

LinkedIn: - chris-l-ayers

Blog: https://chris-ayers.com/

GitHub: Codebytes

Mastodon: @Chrisayers@hachyderm.io

~~Twitter: @Chris_L_Ayers~~