VSCode Container Project Architecture

This document provides a comprehensive overview of the VSCode Container Project architecture, components, and workflows.

Project Overview

The VSCode Container Project provides a secure, containerized development environment with the following key features:

- **Secure VS Code Server**: HTTPS-enabled code-server with automatic certificate generation
- **Let's Encrypt Integration**: Optional production-grade certificates via Cloudflare DNS
- **Documentation Automation**: GitHub Actions for building AsciiDoc and PlantUML files
- **Development Tools**: Pre-installed PlantUML and AsciiDoctor extensions

Architecture Components

Container Services

The project consists of two main containerized services:

VS Code Service (`docker-vsc/`)

- Code-server with SSL support
- Automatic certificate generation (mkcert/OpenSSL)
- Pre-installed development extensions
- Direct HTTPS access on port 8443

Certificate Generator (`docker-certbot/`)

- Let's Encrypt certificate generation
- Cloudflare DNS-01 challenge support
- Runs on-demand via Docker Compose profiles

Documentation System

Source Files

- `.asciidoc` files for main documentation
- `.plantuml` files for diagrams
- `.adoc` and `.puml` files for includes/partials

Generated Output

- HTML and PDF documentation in `docs/`
- PNG diagrams generated alongside source files
- Automated builds via GitHub Actions

Workflows

Development Workflow

1. Developer configures environment files
2. Starts containers with `docker-compose up`
3. Accesses VS Code at `https://localhost:8443`
4. Creates documentation and diagrams
5. Commits changes trigger automatic builds

Certificate Generation Workflow

1. Configure Cloudflare API credentials
2. Run certificate generator with profile
3. Certificates stored in Docker volume
4. Integration with VS Code service (future enhancement)

Documentation Build Workflow

1. Changes to `.asciidoc` or `.plantuml` files trigger GitHub Actions
2. AsciiDoc files converted to HTML/PDF in `docs/`
3. PlantUML files converted to PNG in source directories
4. Generated files committed back to repository

Security Considerations

- All web traffic encrypted via HTTPS
- Password-protected access to VS Code
- Environment files excluded from version control
- Containers run as non-root users where possible
- Certificates automatically generated and managed

File Organization

The project follows a clear structure with containers in subdirectories:

```
docker/
 ├──── docker-compose.yaml          # Main orchestration
 ├──── docker-vsc/                  # VS Code container
 └──── docker-certbot/              # Certificate generator
images/                           # PlantUML diagrams
docs/                             # Generated documentation
.github/workflows/                # Automation
```

Diagrams

The following diagrams illustrate the system architecture:

- Configuration Overview: Project structure and relationships
- VS Code Access: User access workflow
- Certificate Generation: Let's Encrypt process
- GitHub Actions: Documentation build process

Extension and Customization

The project is designed for easy extension:

- Add new containers by creating subdirectories under `docker/`
- Extend GitHub Actions workflow for additional build steps
- Add new documentation by creating `.asciidoc` files
- Create new diagrams with `.plantuml` files
- Customize VS Code extensions via `extensions.json`