STM32Cube for Agentic AI Installation Guide

Release 1.0.0

Nov 03, 2026

Installation

1 Install STM32Cube for Agentic AI **3**
1.1 Installation methods . . . 3
1.2 Licenses . . . 3
1.3 Contexts of this release . . . 3

2 AI agent installation **4**
2.1 Typical installation prompt . . . 4
2.2 Expected interaction . . . 5
2.3 Custom installation prompt . . . 6
2.4 Troubleshooting . . . 7

3 Manual installation **9**
3.1 STM32CubeCLI installation . . . 9
3.2 Typical installation . . . 9
3.3 Setup commands . . . 10
3.3.1 VS Code extension setup and repair . . . 11
3.4 Finish installation . . . 11
3.5 Customization . . . 12
3.5.1 Custom installation . . . 12
3.5.2 Existing project . . . 13
3.5.3 Updates . . . 13
3.5.4 Commands and options . . . 13
3.6 Troubleshooting . . . 13

4 Target, prerequisites, and project folder **15**
4.1 Supported IDEs and agentic systems . . . 15
4.2 Prerequisites . . . 15
4.3 Project and workspace folder . . . 15

1 Install STM32Cube for Agentic AI

The STM32Cube for Agentic AI Toolkit (beta) provides STM32Cube Agentic Contexts to AI coding agents. Use `cube agentic-ai setup` for the typical route: it installs all contexts compatible with the selected target and deploys them into a project folder. The setup command supports six editor-and-agent combinations: see *Target, prerequisites, and project folder*.

Important

This STM32Cube for Agentic AI Toolkit does not itself perform AI inference or make autonomous decisions

1.1 Installation methods

Before either path, review *Target, prerequisites, and project folder* for supported targets, prerequisites, and project/workspace folder guidance. See Choose your installation for a decision diagram covering typical and custom installations, Python, and VS Code profiles.

Each method offers a typical installation with all compatible contexts and a custom installation with interactive context and service selection. Both support target selection. To update an existing installation, see *Updates*.

For host requirements, beta limitations, and the current context inventory, see the STM32Cube for Agentic AI documentation. For command details, see the installer user manual.

1.2 Licenses

STM32CubeCLI is distributed under the SLA0048 software license agreement. The STM32Cube for Agentic AI bundles are distributed under the SLA0044 software license agreement. Tool dependencies, such as the STM32CubeMX2 application and STM32CubeProgrammer, have their own license agreements. Installing a bundle means accepting its license agreement.

`cube bundle install` asks for license acceptance. `setup` installs the contexts and their tool dependencies without license prompts: running it, or sending the agent prompt, is the acceptance, and its result lists the licenses in `data.licenses`. To read a license before installation, run:

```
cube bundle license --online cube-agentic-ai-installer
```

To see what `setup` would install without installing it, run `cube agentic-ai setup --dry-run`.

1.3 Contexts of this release

The installer catalog defines the contexts available to the installation. The catalog can change independently of this page. See the current context inventory and versions for the latest list. The installer is the central point for catalog and context updates.

Note

This toolkit is a beta and receives regular context and guidance updates. Review the current limitations in the user documentation.

2 AI agent installation

Paste a prompt into an AI coding agent that can run terminal commands. The agent gathers the license, target, project-folder, profile, and consent choices before it runs commands. Review *Target, prerequisites, and project folder* first for target, prerequisite, and project-folder guidance.

2.1 Typical installation prompt

This prompt asks which editor and AI coding agent to use. GitHub Copilot in VS Code is the default, not a requirement. It uses `setup` to install the contexts and deploy them into a project folder under the current working directory.

```
Install STM32Cube for Agentic AI for my selected editor and AI coding agent.

Before running any command, ask me these questions in one message and wait
for my answers:
1. Licenses: do I accept the SLA0044 license agreement of the STM32Cube for
   Agentic AI bundles and the license agreements of the tools that they
   install? The commands accept them on my behalf. If STM32CubeCLI must be
   installed, ask whether I accept the SLA0048 license agreement before
   running its installer. Stop if I decline any license.
2. Target: choose one supported combination:
   - GitHub Copilot in VS Code (default)
   - GitHub Copilot CLI: --ide none
   - Claude Code in VS Code: --agentic-system claude-code
   - Claude Code CLI: --ide none --agentic-system claude-code
   - Cursor editor: --ide cursor --agentic-system cursor
   - Cursor CLI: --ide none --agentic-system cursor
   Host extensions required by a selected editor are separate
   prerequisites. The toolkit installer does not install or manage them.
   Do not treat them as context extensions or claim that setup installs
   them.
3. Project folder: what project folder should receive the contexts? Use a
   project or workspace folder, not my home folder.
4. VS Code profile: for a VS Code target, which exact profile should
   receive the extensions? Do not ask this for another target.
5. Consent: do I accept installing bundles and their declared dependencies
   on my machine and deploying files into the selected project folder?
If I do not accept the licenses or the installation, stop.

After my answers, run these steps without asking me anything else. Use the
terminal tool you already have. Stop at the first failure, show me the full
error, and propose a fix.

1. If "cube --version" fails, install STM32CubeCLI with the command for the
   shell of your terminal tool:
   - PowerShell: irm https://developer.st.com/bundles/install/install.ps1 | iex
   - bash or zsh: curl -fsSL https://developer.st.com/bundles/install/install.sh | bash -
     then: export PATH="$HOME/.local/stm32cube/bin:$PATH"
2. From a working directory outside any deployed project's bundle lock,
   run "cube bundle install --yes cube-agentic-ai-installer", then
   "cube bundle update cube-agentic-ai-installer" to use the latest
   available installer even if an older version was already installed.
```

(continues on next page)

(continued from previous page)

```
3. Build the target options from my answer. For any VS Code target, add
   "--install-vscode-extensions". If I chose a named profile, also add
   "--vscode-profile <exact profile name>". Otherwise use the Default
   profile. Run "cube agentic-ai setup --project-folder <project-folder>
   --python venv <target options> <VS Code extension options>". Do not pass
   an installer or context version. setup can take several minutes: allow a
   long timeout or run it in the background and wait. If it stops before
   deploying, run it again. Once it has deployed, run
   "cube agentic-ai doctor <project-folder> --fix" instead.
4. Check that the JSON result has "ok": true and
   "data.status": "ALL_DEPLOYED". ALL_DEPLOYED confirms deployment, not
   dependency readiness. Also inspect "data.readiness",
   "data.contexts[].requirements[]", and "data.nextSteps". If action
   remains, run "cube agentic-ai doctor <project-folder> --fix". For VS Code
   extension installation, add "--vscode-profile <exact existing profile
   name>". Complete manual steps and check again with doctor. Do not rerun
   setup on the deployed project. Show me "data.licenses", "data.notices",
   "data.unavailableContexts", and "data.nextSteps".
5. VS Code only. Setup installs the declared extensions because the command
   includes "--install-vscode-extensions". If "data.vscodeCli.onPath" is
   false, do not run "code" or ask me to install extensions manually. Tell
   me that extension installation could not run because "code" is not on
   PATH. Otherwise, for each entry of "data.vscodeProfiles", run only its
   "verifyCommand". If the profile is not found, stop and ask me to choose
   an existing profile. Inspect "data.vscodeInstall" and "data.nextSteps"
   for the installation result. Do not run "installCommands" yourself.
6. Apply "data.hostInstructionUpdate" as "data.nextSteps" says. For VS Code,
   run "data.openCommand" if present and tell me to continue in the new
   VS Code window.
7. Open the deployed project with the selected editor or start the selected
   CLI agent from that project folder. Approve its project MCP servers as
   required by that agent. Ask it to list the STM32 services deployed in
   this project, then make one read-only request using a deployed context.
   Ask it to cite its source or evidence. Do not change files or operate a
   device during this first-use check.

Never ask me for a password or a token in the chat.
```

`setup` can target GitHub Copilot, Claude Code or Cursor, in an editor or in a terminal (see *Target, prerequisites, and project folder*), a project folder not yet deployed by this installer, or a selection of contexts: answer question 2 with your choice, or add your choices to the prompt, for example `folder C:\work\demo, only the Embedded Software and STM32CubeMX2 contexts`. The agent adds the matching options, which the *customization* section lists. For an already deployed project, use `project-add` or `project-remove` followed by `doctor --fix` instead of setup.

For copyable `setup` commands for all six targets, see *Setup commands* in the *manual installation*.

2.2 Expected interaction

- **The agent asks for licenses, target, project folder, VS Code profile when** relevant, and consent before it runs a command. For VS Code targets, setup installs declared extensions in the selected existing profile.
- The AI coding agent can request an approval for each terminal command. These approvals depend on the permission settings of the agent, not on the installer.
- `setup` writes progress lines while it installs, and prints one JSON result at the end.
- **After setup, continue in the selected editor or start the selected CLI agent**

from the project folder. Approve project MCP servers as required by that agent, then verify the deployed services with a read-only request. The exact approval steps depend on the target. See the supported target details.

For a VS Code target, keep the project folder as the top-level workspace folder. See *Project and workspace folder*.

2.3 Custom installation prompt

Use this prompt to see the installation in two steps and to choose the contexts and services one by one. The first step installs all contexts on the machine (`machine-bootstrap`). The second step deploys the ones you choose into a project (`project-deploy-installed`) and asks about each context and each service.

```
Install STM32Cube for Agentic AI step by step, with my choice of contexts
and services.

Before running any command, ask me these questions in one message and wait
for my answers:
1. Licenses: do I accept the SLA0044 license agreement of the STM32Cube for
   Agentic AI bundles and the license agreements of the tools that they
   install? The commands accept them on my behalf. If STM32CubeCLI must be
   installed, ask whether I accept the SLA0048 license agreement before
   running its installer. Stop if I decline any license.
2. Target: choose one supported combination:
   - GitHub Copilot in VS Code (default)
   - GitHub Copilot CLI: --ide none
   - Claude Code in VS Code: --agentic-system claude-code
   - Claude Code CLI: --ide none --agentic-system claude-code
   - Cursor editor: --ide cursor --agentic-system cursor
   - Cursor CLI: --ide none --agentic-system cursor
   Host extensions required by a selected editor are separate
   prerequisites. The toolkit installer does not install or manage them.
   Do not treat them as context extensions or claim that setup installs
   them.
3. VS Code profile: for the VS Code target, do I use the Default profile or
   another profile? If another, what is its exact name? Do not ask this for
   another IDE target.
4. Project folder: what project folder should receive the contexts? Use a
   project or workspace folder, not my home folder.
5. Consent: do I accept that you install software on my machine, in my user
   account and without administrator rights, and create the project folder
   if it does not exist?
If I do not accept the licenses or the installation, stop.

After my answers, run these steps. Use the terminal tool you already have.
Stop at the first failure, show me the full error, and propose a fix.

1. If "cube --version" fails, install STM32CubeCLI with the command for the
   shell of your terminal tool:
   - PowerShell: irm https://developer.st.com/bundles/install/install.ps1 | iex
   - bash or zsh: curl -fsSL https://developer.st.com/bundles/install/install.sh | bash -
     then: export PATH="$HOME/.local/stm32cube/bin:$PATH"
2. From a working directory outside any deployed project's bundle lock,
   run "cube bundle install --yes cube-agentic-ai-installer", then
   "cube bundle update cube-agentic-ai-installer" to use the latest
   available installer even if an older version was already installed.
3. Run "cube agentic-ai machine-bootstrap --upgrade-installed-in-range".
   It installs all contexts and their tools on the machine, and nothing in
```

(continues on next page)

(continued from previous page)

```
   any project.
4. Create the project folder if it does not exist: "mkdir -p" in bash or
   zsh, "New-Item -ItemType Directory" in PowerShell. If it exists and is not
   the project I named, ask me before using it.
5. Run "cube agentic-ai project-deploy-installed <folder>
   --mode interactive --services-mode interactive <target options>". Target
   options: none for GitHub Copilot in VS Code, "--ide none" for GitHub
   Copilot CLI or app, "--agentic-system claude-code" for Claude Code in
   VS Code, "--ide none --agentic-system claude-code" for Claude Code CLI,
   "--ide cursor --agentic-system cursor" for the Cursor editor,
   "--ide none --agentic-system cursor" for Cursor CLI, "--canonical" for
   another agentic system. The installer asks, for
   each context and each service, whether to deploy it. Relay each question
   to me, and type my answer in the terminal. Do not answer for me. Required
   items are deployed without a question.
6. Check that the results of steps 3 and 5 have "ok": true and
   "data.status": "ALL_DEPLOYED". Run "cube agentic-ai doctor <folder> --fix":
   the deployment does not install what the contexts need, such as Python
   packages or the STM32CubeMX2 application. For every VS Code target,
   including Claude Code and GitHub Copilot, add
   "--vscode-profile <exact existing profile name>". For the Default
   profile, explicitly use '--vscode-profile "Default"'. Without this
   option, doctor does not install declared extensions. Show me
   "data.readiness", "data.nextSteps", and "data.vscodeInstall" when present.
7. VS Code only. If "data.vscodeProfile" is absent, skip the extension
   commands. Otherwise, if "code --version" fails, stop and tell me extension
   installation could not run because "code" is not on PATH. Do not ask me
   to install extensions manually. Check "data.vscodeInstall": claim
   success only when its "status" is "PROCESSED" and none of its
   "extensions[]" entries has "outcome": "FAILED". If the profile was not
   found, stop and ask me to choose an existing profile. For another repair
   failure or a missing result, stop, show the reported details and
   "data.nextSteps", and propose a fix. After successful repair, run
   "data.vscodeProfile.verifyCommand". Do not repeat its "installCommands".
   Run "data.vscodeProfile.openCommand" only after the profile is verified.
   Extension installation does not create ".vscode/extensions.json" in
   this custom route. Do not use that file's presence or absence as proof
   that extensions are installed. For another target, tell me how to open
   the folder with it.
8. Use the selected agent with the project folder open. Approve its project
   MCP servers as required by that agent. Ask it to list the STM32 services
   deployed in this project, then make one read-only request using a
   deployed context. Ask it to cite its source or evidence. Do not change
   files or operate a device during this first-use check.

Never ask me for a password or a token in the chat.
```

For the same choices in a terminal, see *Custom installation* in the manual installation. To update an existing installation, see the installer user manual. For target-specific MCP approval steps, see the supported target details.

2.4 Troubleshooting

Agent asks for a password or a token

Type credentials directly in the terminal, never in the chat. Then tell the agent to continue.

A step fails or times out

The agent stops and shows the error. After the fix, ask the agent to run the failed step again. `machine-bootstrap` and `doctor --fix` can run again safely. `setup` runs only in a project folder that has never been deployed: once it has deployed, finish with `doctor <project folder> --fix`.

The agent says that code is not on the PATH

The VS Code extensions are not necessarily missing: the agent could not check them. Add `code` to the PATH, or install the extensions from the Extensions view of VS Code. See the VS Code command line reference.

For other issues, see the troubleshooting section of the *manual installation*.

3 Manual installation

Run the commands of the typical installation in PowerShell, bash, or zsh. First read *Target, prerequisites, and project folder* for target selection, prerequisites, and the project/workspace folder requirements.

3.1 STM32CubeCLI installation

Skip this step if `cube --version` prints a version.

PowerShell

bash or zsh

```
irm https://developer.st.com/bundles/install/install.ps1 | iex
```

For Linux, macOS, and Windows with Git Bash, WSL, or MSYS2:

```
curl -fsSL https://developer.st.com/bundles/install/install.sh | bash -
```

The script installs STM32CubeCLI for the current user in `~/.local/stm32cube` and adds `~/.local/stm32cube/bin` to `PATH`. It does not require administrator rights. Open a new terminal, then check the installation:

```
cube --version
```

STM32CubeCLI is distributed under the SLA0048 software license agreement. See *Licenses*.

3.2 Typical installation

Install the installer, then use `setup` to install and deploy all contexts compatible with your selected editor and AI coding agent. GitHub Copilot in VS Code is the default target. Add the target options shown in *Setup commands* to choose another supported editor or CLI agent. Run the commands from a working directory and choose a project folder beneath it. The commands use the latest installer available to STM32CubeCLI. They do not pin an installer or context version.

Run installer installation and updates outside a deployed project's bundle lock so the project does not select an older installer version.

PowerShell

bash or zsh

```
cube bundle install --yes cube-agentic-ai-installer
cube bundle update cube-agentic-ai-installer

# Optional: preview setup without making changes
cube agentic-ai setup --project-folder .\my-agentic-ai-project --python venv --dry-run
# Default VS Code target: install declared extension dependencies
cube agentic-ai setup --project-folder .\my-agentic-ai-project --python venv --install-vscode-
↪extensions
```

```
cube bundle install --yes cube-agentic-ai-installer
cube bundle update cube-agentic-ai-installer

# Optional: preview setup without making changes
cube agentic-ai setup --project-folder ./my-agentic-ai-project --python venv --dry-run
```

(continues on next page)

(continued from previous page)

```
# Default VS Code target: install declared extension dependencies
cube agentic-ai setup --project-folder ./my-agentic-ai-project --python venv --install-vscode-
↪extensions
```

For other editor and CLI targets, use the matching options in *Setup commands*.

The commands perform two stages:

Installer installation. `cube bundle install` installs the installer, which provides the `cube agentic-ai` command. Accept the license agreement when prompted. `cube bundle update` updates an existing installer.

Project setup. `setup` installs compatible STM32Cube Agentic Contexts and their tool dependencies on the machine. It creates `my-agentic-ai-project` under the working directory, deploys the contexts and their services into it, and checks the result. It asks no questions. Running it accepts the license agreements listed in `data.licenses`. Installation progress is printed and can take several minutes. If interrupted before deployment, run setup again to resume. After deployment, run `doctor <project folder> --fix` instead.

`setup` prints one JSON result. Check `ok`, `data.status`, `data.readiness`, and `data.nextSteps`.

Important

`ALL_DEPLOYED` confirms deployment, not dependency readiness. Setup can return `ALL_DEPLOYED` with `ACTION_REQUIRED` or `CHECK_MANUALLY`, for example when a declared requirement needs attention. Follow `data.nextSteps` and run `cube agentic-ai doctor <project-folder> --fix` for resolvable requirements. Complete remaining manual prerequisites, then rerun doctor to check readiness. Do not rerun setup on the deployed project. For VS Code extension setup or repair, see *VS Code extension setup and repair*.

3.3 Setup commands

After installing or updating the installer above, choose one target below and run only its command instead of the typical `setup` command. Each example uses a separate project folder. The same syntax works in PowerShell, bash, or zsh.

VS Code + Copilot

Copilot CLI

VS Code + Claude Code

Claude Code CLI

Cursor editor

Cursor CLI

```
cube agentic-ai setup --project-folder ./vscode_copilot_project/ --install-vscode-extensions --
↪python venv
```

```
cube agentic-ai setup --project-folder ./copilot_cli_project/ --python venv --ide none
```

```
cube agentic-ai setup --project-folder ./vscode_claude_code_project/ --python venv --ide vscode -
↪-agentic-system claude-code --install-vscode-extensions
```

```
cube agentic-ai setup --project-folder ./claude_code_cli_project/ --python venv --ide none --
↪agentic-system claude-code
```

```
cube agentic-ai setup --project-folder ./cursor_ide_project/ --python venv --ide cursor --
↪agentic-system cursor
```

```
cube agentic-ai setup --project-folder ./cursor_cli_project/ --python venv --ide none --agentic-
↪system cursor
```

For deployment layouts and target-specific behavior, see *Target, prerequisites, and project folder*.

3.3.1 VS Code extension setup and repair

This section applies only to VS Code targets. GitHub Copilot in VS Code requires the GitHub Copilot Chat extension (`GitHub.copilot-chat`), and Claude Code in VS Code requires Anthropic's Claude Code extension. The installer does not install or manage these host extensions. Install the appropriate host extension before using its agent. See the GitHub Copilot Chat extension Marketplace page and the Claude Code for VS Code guide.

The typical setup command uses the Default VS Code profile and includes `--install-vscode-extensions` so `setup` runs `code` to install declared context extensions. Remove the flag to install them manually. To use an existing named profile, add `--vscode-profile "<exact profile name>"`. The toolkit does not install or manage the host extensions required by Copilot Chat or Claude Code.

For an already deployed project, `doctor --fix` installs declared extensions only when you pass an explicit `--vscode-profile`. Without that option, `doctor --fix` still resolves other declared requirements and reports extension commands for manual installation in the Default profile. For the Default profile, if the result includes `data.vscodeProfile`, run its `verifyCommand` and then its `installCommands`. Reload VS Code after installing the extensions. To authorize extension repair in an existing named profile, run:

```
cube agentic-ai doctor <project-folder> --fix --vscode-profile "<exact existing profile name>"
```

3.4 Finish installation

Use this checkpoint after either typical or custom installation, including installation by an AI agent. Replace `<project-folder>` with the folder you deployed.

1. Check the command result: `ok` must be `true`. For deployment, `data.status: ALL_DEPLOYED` confirms that files were deployed, not that all dependencies are ready. Follow `data.readiness` and `data.nextSteps`. Use `cube agentic-ai doctor <project-folder> --fix` for resolvable requirements. For target-specific extension installation, profile, and repair instructions, follow *Target, prerequisites, and project folder*.
2. Complete reported manual prerequisites and apply any returned `data.hostInstructionUpdate`.
3. Open the deployed folder with the selected editor or start the selected CLI agent from that folder. Use `data.openCommand` when present. Follow the selected host's instructions for opening or reloading the project. Keep the project as the top-level workspace folder.
4. Approve the project MCP servers in the selected agent. The installer cannot grant this permission. Follow the agent-specific MCP approval steps.
5. Ask the agent to list the services deployed in this project, then make one read-only request using a deployed context:

   ```
   List the STM32 services deployed in this project.
   ```

   To assess dependency status and readiness without changing the project, run `cube agentic-ai doctor <project-folder>` without `--fix`. This check is read-only. A host integration check can remain `CHECK_MANUALLY`. Verify it in the selected editor or agent.

The first-use check is complete when the selected service returns a sourced result and you have resolved or explicitly verified the reported manual checks. Service discovery alone does not prove that an MCP server can start. For startup failures, inspect the agent's server status and logs. `doctor` does not diagnose arbitrary MCP or agent-host failures.

Next, follow Try it out for a development task. For Build and DAP Debug, also follow the context's project-discovery and configuration requirements before building or debugging.

3.5 Customization

`setup` options choose the target, the folder and the contexts. Add them to the `setup` command:

Option	Effect
`--project-folder "<path>"`	**Project top folder, created when missing. Use the top-level folder of** an existing folder only if it has not already been deployed by the installer. A folder with `.cube-ai/services-lock.json` is refused.
`--contexts <bundle,...>`	Install only these contexts, for example `cube-agentic-ai-esw, cube-agentic-ai-mx2-skill`.
`--python venv` or `--python "<path>"`	Where the Python packages of the contexts go: a dedicated environment, or an existing Python.
`--vscode-profile "<name>"`	VS Code profile for the extensions. Repeat the option for several profiles.
`--install-vscode-extensions`	Let `setup` run `code` to install the declared extensions. VS Code target only.
`--upgrade-installed-in-range`	Upgrade eligible machine bundles before deploying into a folder not yet deployed by this installer.
`--dry-run`	Report what `setup` would do, and change nothing.

For supported targets, options, project file layouts, and VS Code-only context availability, see *Target, prerequisites, and project folder*.

3.5.1 Custom installation

`setup` does the two steps below in one command. Run them separately to choose the contexts and the services one by one:

PowerShell

bash or zsh

```
cube bundle install --yes cube-agentic-ai-installer
cube bundle update cube-agentic-ai-installer
cube agentic-ai machine-bootstrap --upgrade-installed-in-range
New-Item -ItemType Directory -Path .\my-agentic-ai-project
Set-Location .\my-agentic-ai-project
cube agentic-ai project-deploy-installed . --mode interactive --services-mode interactive
cube agentic-ai doctor . --fix
```

```
cube bundle install --yes cube-agentic-ai-installer
cube bundle update cube-agentic-ai-installer
cube agentic-ai machine-bootstrap --upgrade-installed-in-range
mkdir -p ./my-agentic-ai-project
cd ./my-agentic-ai-project
cube agentic-ai project-deploy-installed . --mode interactive --services-mode interactive
cube agentic-ai doctor . --fix
```

These commands use the default VS Code target. If `doctor` returns `data.vscodeProfile`, run its verification and extension-install commands as described in *VS Code extension setup and repair*. For another supported target, add the options from *Target, prerequisites, and project folder* to `project-deploy-installed`. For custom deployment, `doctor` uses the saved Python choice or automatic detection. It has no `--python` option. To select a Python runtime explicitly, choose the typical route with `setup --python` before deployment. See the eSW Python troubleshooting for detection and environment details.

Continue with *Finish installation*.

3.5.2 Existing project

`setup --project-folder "<path>"` can deploy into a folder that already exists, provided it has not already been deployed by the installer (there is no `.cube-ai/services-lock.json`). Setup refuses a deployed project with `SETUP_PROJECT_ALREADY_DEPLOYED`. For an existing deployed project, use `project-add` or `project-remove` and then `doctor --fix`. See *Updates*. Build and DAP Debug workflows require the project folder to be the top-level folder opened as the VS Code workspace.

3.5.3 Updates

To install newer eligible context versions on the machine, first update the installer if needed, then run `machine-bootstrap`. This does not change the versions already deployed in a project:

```
cube bundle update cube-agentic-ai-installer
cube agentic-ai machine-bootstrap --upgrade-installed-in-range true
```

To update a deployed project, first make the desired context versions available on the machine. Then see which deployed contexts are behind with:

```
cube agentic-ai doctor <project folder>
```

For each context that is behind, read `data.nextSteps`: `doctor` prints the `project-remove`, `project-add` and `doctor --fix` commands for that project, including its recorded target and the deployed services. If the desired version is not installed yet, run `machine-bootstrap --upgrade-installed-in-range true` first, then run doctor again. Removing before adding avoids leaving services from the old version in the project.

For a project folder not yet deployed by this installer, `setup --upgrade-installed-in-range` can upgrade eligible machine bundles before deploying into that folder. It cannot update an already-deployed project.

For extension installation or repair in a named VS Code profile, see *VS Code extension setup and repair*.

The project housekeeping page of the installer user manual describes this procedure.

3.5.4 Commands and options

To assess dependency status and readiness without changing the project, run `cube agentic-ai doctor <project folder>` without `--fix`. To resolve the dependencies that the installer can manage, add `--fix`. To list the contexts deployed in a project, run `cube agentic-ai project-lock-show <project folder>`. To add one context later, run `cube agentic-ai project-add <bundle_name> <project folder> --services all`, then `doctor <project folder> --bundles <bundle_name> --fix`. Only `setup` and `doctor` install what a context needs. The commands `project-add`, `project-deploy` and `project-deploy-installed` do not.

To read the concise installer guide, run:

```
cube agentic-ai get-user-manual --format tldr
```

The installer user manual describes all commands and options.

3.6 Troubleshooting

cube: command not found

Open a new terminal. If the command is still not found, add `~/.local/stm32cube/bin` to `PATH`. On Windows, sign out and sign in again to refresh the environment.

code: command not found

The VS Code extensions are not necessarily missing: the terminal could not check them. On macOS, run `Shell Command: Install 'code' command in PATH` from the VS Code Command Palette. On Windows and Linux, reinstall VS Code or add its `bin` folder to the `PATH`. Or install the extensions from the Extensions view of VS Code. See the VS Code command line reference.

Installer not updated

Update the installed installer with `cube bundle update cube-agentic-ai-installer`, then use `cube agentic-ai show-catalog` to confirm the contexts available to that installer.

"status": "PARTIAL"

At least one bundle failed: `data.nextSteps` names it. Fix the cause, then add it with `project-add` and run `doctor <project folder> --fix`: `setup` refuses a project folder where it has already deployed contexts. For more detail, add diagnostic output:

```
cube agentic-ai --verbose standard setup
```

"readiness": "ACTION_REQUIRED"

A context needs something that `setup` could not do. Read `data.nextSteps` of the result, or run `cube agentic-ai doctor <project folder>`.

Download failure

Check the network connection and the proxy settings, then run the failed command again. Bundles that are already installed are not downloaded again.

4 Target, prerequisites, and project folder

Use this page to choose an agent target and project folder before starting either the *agent installation* or *manual installation*.

4.1 Supported IDEs and agentic systems

The installer supports these six editor-and-agent combinations. The default target is GitHub Copilot in VS Code. Pass the options listed in the table for the other combinations.

IDE and agentic system	Options	MCP configuration file	Build and DAP Debug
GitHub Copilot in VS Code (default)	None	`.vscode/mcp.json`	Yes
GitHub Copilot CLI or app	`--ide none`	`.mcp.json`	No
Claude Code in VS Code	`--agentic-system claude-code`	`.mcp.json`	Yes
Claude Code CLI	`--ide none --agentic-system claude-code`	`.mcp.json`	No
Cursor editor	`--ide cursor --agentic-system cursor`	`.cursor/mcp.json`	No
Cursor CLI	`--ide none --agentic-system cursor`	`.cursor/mcp.json`	No

The Build and STM32 DAP Debug contexts require the VS Code IDE target. They are not installed for Cursor or terminal-only targets. For deployment layout and agent approval steps, see the supported target details.

For another agentic system outside these six combinations, `--canonical` deploys a portable `.cube-ai/` layout for the agent to adapt. This is a generic layout, not an additional native installer target.

4.2 Prerequisites

STM32CubeCLI and the selected AI coding agent must be installed before setup. The toolkit never installs Python or the VS Code application. Python 3.10 or later is required by the STM32Cube Embedded Software Context. The Build and DAP Debug contexts require VS Code and its STM32CubeIDE extension. Debug also requires DebugMCP and a supported debugger or GDB server. These contexts and their VS Code extensions are not installed when another IDE or a CLI target is selected.

GitHub Copilot in VS Code requires the GitHub Copilot Chat extension (`GitHub.copilot-chat`), and Claude Code in VS Code requires Anthropic's Claude Code extension. The installer does not install or manage either extension. Install them separately from the GitHub Copilot Chat extension Marketplace page and the Claude Code for VS Code guide.

Review the full list of system and context prerequisites before installing. On Windows, PowerShell 7.3 or later is recommended. The user guide explains the PowerShell 5.1 Ctrl+U issue and workaround.

4.3 Project and workspace folder

Choose the top-level folder of the project or workspace. Do not deploy the contexts directly into the user home folder. The installer places selected context files and their project configuration in this folder, so each project can have its own à-la-carte selection. Only the context bundles are installed at machine scope. Project locks record the versions and services used by each project.

For the Build and DAP Debug contexts, open this folder as the VS Code workspace root. STM32CubeIDE for Visual Studio Code must discover the CMake project at that level before build or debug workflows can run.