BugIt
by Taskivator

Complete step-by-step manual

BugIt

QA Bug-Filing Agent for VS Code

First-Time Setup & Everyday Use: a detailed, click-by-click guide written for people who have never done anything like this before. No technical background needed. Follow it in order and you'll be filing clean bug tickets today.

Runs in: VS Code on Windows (primary), macOS & Linux

© 2026 Taskivator. All Rights Reserved.

WHAT'S INSIDE

Do **Part 1** once, in order. Everything after that you can dip into whenever you need it. The **Troubleshooting** and **FAQ** near the end answer the most common questions, so please check them before emailing support.

Part 1 · Setup (do this once)

- Step 0: Choose your AI assistant
- Step 1: Install VS Code
- Step 2: Unzip your BugIt download
- Step 3: Open BugIt in VS Code
- Step 4: Pick the BugIt agent in chat
- Step 5: Activate your license
- Step 6: Accept the terms (once)
- Step 7: Run "Begin setup"
- Step 8: Connect your tracker
- Step 9: Save your sign-ins
- Step 10: Confirm you're ready

Part 2 · Everyday use

- Commands at a glance

Part 3 · Make it yours

Part 4 · Updates, backups & safety

Part 5 · Troubleshooting & FAQ

- Frequently asked questions
- Team licenses: questions

Part 6 · License & support

The one rule that keeps you safe

BugIt **never** files anything without showing you a preview and getting your typed `FILE IT` first, as the whole message. A plain "yes" is not enough. You are always in control. Want to practise with zero risk? Type `dry run` and it writes the whole report **without** filing.

First-time setup vs. everyday use

You only follow all of **Part 1** once. After that, opening BugIt is quick, and updates are one command (`Update`). See Part 4.

PART 1 · SETUP: DO THIS ONCE

Roughly 15 minutes. Do the steps in order; each one builds on the last. Don't skip ahead.

0 Choose your AI assistant

> **This guide follows the GitHub Copilot path**
>
> BugIt is the "driver" and the AI is the "engine" (it is what actually writes the reports). These steps set you up with GitHub Copilot inside VS Code, which is the quickest route if you are starting from nothing. BugIt also works with the Claude extension, with any other assistant that can run a command, and on its own with your own AI key. If you already use one of those, skip to Step 2 and open the BugIt folder in that tool instead of VS Code.

Before you begin: you need the latest VS Code and an assistant that can run a command, such as GitHub Copilot Chat or the Claude extension.

What it is: GitHub Copilot is an AI assistant from GitHub (owned by Microsoft). It has a free tier for light use and a paid tier for heavy use. If you choose it, you'll install it in Step 1 and sign in.

1. You'll need a **GitHub account**. If you don't have one, create a free one at `github.com/signup`.
2. Copilot's free tier is enough to try BugIt. If you file bugs all day, the paid "Copilot Pro" plan is worth it, and you can upgrade anytime at `github.com/features/copilot`.
3. You'll actually turn Copilot on inside VS Code in Step 1, so there's no need to do anything else here yet.

1 Install VS Code

VS Code is a free Microsoft app. It's the "window" BugIt lives in.

1. Open your web browser and go to `code.visualstudio.com`.
2. Click the big blue **Download** button (it detects your system automatically).
3. Open the downloaded file and install it: **accept all the default options**, just keep clicking Next/Install.
4. Open VS Code once it's installed.

Turn on GitHub Copilot inside VS Code

1. On the far left of VS Code, click the **Extensions** icon (it looks like four small squares).
2. In the search box, type **GitHub Copilot**.
3. Click **Install** on "GitHub Copilot" (and, if offered, "GitHub Copilot Chat").
4. A prompt will ask you to **Sign in to GitHub**. Click it and sign in with your GitHub account in the browser window that opens.

> **You'll know it worked when:** a small Copilot icon appears at the bottom of VS Code, and the Chat panel opens (the speech-bubble icon on the left, or press `Ctrl+Alt+I`).

2 Unzip your BugIt download

Your purchase came with a .zip file (for example `bugit-qa-agent.zip`). You need to unzip it; running it zipped won't work.

1. Find the .zip file (usually in your **Downloads** folder).
2. **Right-click** it → choose **Extract All...** (Windows) or double-click it (Mac).
3. Choose a simple location like your **Documents** folder, then click Extract.
4. You'll now have a **BugIt** folder with lots of files inside. Remember where it is.

> **Don't put it in OneDrive or a synced folder**
> Cloud-sync folders can cause file conflicts. Your Documents or Desktop is perfect. Also don't rename the files inside; just leave the folder as-is.

3 Open BugIt in VS Code

1. In VS Code, click **File → Open Folder** (top-left menu).
2. Navigate to where you unzipped BugIt (e.g. Documents).
3. Click the **BugIt** folder once to select it, then click **Select Folder**.
4. If VS Code asks "Do you trust the authors of the files in this folder?", click **Yes, I trust the authors**.

> ✅ **You'll know it worked when:** you see the BugIt files (folders like `tools`, `.github`, and files like `README.md`) listed down the left side of VS Code.

4 Pick the BugIt agent in chat

1. Open the Copilot **Chat** panel: click the speech-bubble icon on the left, or press `Ctrl+Alt+I` (Windows) / `Cmd+Alt+I` (Mac).
2. At the top of the chat box there's a small **mode dropdown** (it may say "Ask" or "Agent").
3. Click it and choose **BugIt QA Agent** from the list.

> **Don't see "BugIt QA Agent" in the list?**
> Make sure you opened the BugIt folder (Step 3), not a different folder; the agent loads from inside it. Close and reopen the folder if needed.

5 Activate your license

A BugIt account is all you need; there is no license key to copy or paste. You approve this device in your browser, and the `Activate` command finishes the rest.

1. In the chat box, type `hi` and press Enter.
2. Type `Activate`. BugIt opens the **BugIt Portal** in your browser. Sign in to your own account, choose your Solo or Team entitlement, and approve this device. Then return to VS Code and BugIt finishes authorizing automatically. Your password stays in the browser and is never entered into VS Code.

> **You'll know it worked when:** BugIt confirms you are activated and names the account you signed in as.

> **Keep your account private**
>
> Your account and its entitlement are tied to you. Don't share, post, or resell them; shared access can be revoked. Team members each sign in with their own account; there is no shared key and no shared login. Moving to a new computer later is fine (see FAQ).

6 Accept the terms (once)

The first time, BugIt shows a short summary and asks you to accept before it does anything else.

1. Read the short summary it shows (you review each ticket before it's filed; your project's safety is your responsibility; the safeguards are aids, not guarantees, and since they're followed by whichever AI model is answering you in Copilot, a lighter/free model may occasionally skip a step or need a command repeated, where a stronger model wouldn't).
2. Choose **I agree** from the two options BugIt shows.

> **You'll know it worked when:** BugIt confirms and moves on to setup. You're only asked this once; it remembers.

7 Run "Begin setup"

Now BugIt configures itself by interviewing you in plain language. You never edit a settings file by hand.

```
YOU TYPE
Begin setup
```

It will ask short questions and turn on only what you pick. The most important one is your tracker.

8 Connect your tracker

Setup recorded *which* tracker you use. This step is what lets BugIt actually file into it. You create an API token in your own tracker account, then save it on this computer with one command. BugIt files straight to your tracker's own REST API using that token, so nothing has to stay running and filing never uses an MCP server.

8a · Create a token in your own tracker account

YOUR TRACKER	WHERE TO CREATE IT
Jira Cloud	`id.atlassian.com/manage-profile/security/api-tokens`
Azure DevOps	`dev.azure.com/YOUR-ORG/_usersSettings/tokens`
GitHub Issues	`github.com/settings/personal-access-tokens/new`
GitLab Issues	`gitlab.com/-/user_settings/personal_access_tokens`
Bugzilla	in your own Bugzilla: Preferences → API Keys
YouTrack	in your own YouTrack: avatar → Profile → Account Security → Authentication
Linear	`linear.app/settings/account/security`
Shortcut	`app.shortcut.com/settings/account/api-tokens`
ClickUp	`app.clickup.com/settings/apps`
Asana	`app.asana.com/0/my-apps`
Trello	`trello.com/apps/admin` (an API key AND a token you authorise with it)

> **Copy your token the moment it's shown**
>
> Most services show a new token only once. Copy it immediately and keep it somewhere safe until it's stored in your computer's secure store (Step 9). Pick the longest expiry your tracker offers so you don't have to redo this soon.

8b · Save it on this computer

Ask your assistant to connect your tracker and it runs this for you. To do it yourself, open a terminal inside VS Code and run the command for your tracker, using `ado`, `github`, `gitlab` and so on in place of `jira`.

YOU RUN (IN THE TERMINAL)

```
python tools/connect.py jira
```

8c · Paste the token into the masked prompt

The command asks for the token in a masked local prompt: it is never shown on screen and never reaches the chat, a ticket or `config.json`. Paste it there and nowhere else. BugIt then signs in, lists the projects the token can really see, reads the issue types you are allowed to create, and **refuses to save a credential that cannot file**. Setup finishing therefore means filing works.

8d · Check it's connected

To confirm at any time, run the check in the terminal, or just ask in the chat:

YOU RUN (IN THE TERMINAL)

```
python tools/connect.py jira --check
```

YOU TYPE

```
Check status
```

You'll know it worked when: BugIt lists your tracker as connected / healthy.

You only connect once

Your API token is never readable on disk. On Windows it is encrypted with your own Windows account's key and kept inside your BugIt folder, so a copy of that folder is useless on any other machine or to any other user; on macOS and Linux it goes into the Keychain or the libsecret service and the file holds only a reference. Closing VS Code changes nothing: reopen the folder and BugIt is ready to file.

9 Save your sign-ins

Once you save your token (Step 8c), your computer's own protection keeps it, so you never have to look your token up again: on Windows that is an encrypted record only your Windows account can open, and on macOS and Linux it is the Keychain or the Secret Service. `config.json` holds only orgs and URLs, and credential values never touch the chat log or a plain file.

To see which trackers already have a saved credential, type:

YOU TYPE

```
Check saved credentials
```

It reports which trackers have a saved credential and it **never shows credential values**. If one is ever removed or expires, run the same command again (Step 8b) and the new token replaces the old one.

10 Confirm you're ready

Ask BugIt to double-check everything before you file for real:

YOU TYPE

```
Check readiness
```

It lists anything still missing (usually just "connect your tracker" or "activate"). When it's all green, you'll see "✅ Setup complete."

That's setup done, forever

You won't repeat Part 1. From now on, opening BugIt is: open the folder → type `hi`. To change any choice later, just type `Begin setup` and say what to change.

No GitHub Copilot? There's a terminal wizard

If you can't use Copilot, open the built-in Terminal (Terminal → New Terminal) and run `python tools/setup_wizard.py`. It asks a few questions and writes your settings without the AI. You can also run BugIt standalone with your own AI key; see the FAQ.

PART 2 · EVERYDAY USE

Once set up, you work entirely from the chat. Talk naturally, or use the exact examples below. Type `help` anytime for the full list.

Write a bug report

BugIt's main job. Describe the bug in one plain sentence; it writes the full ticket, auto-fills the version and category, checks for duplicates, shows you a preview, and files it only after you reply with `FILE IT` as your whole message. “yes”, “go ahead” and “looks good” are refused.

YOU TYPE

```
Write a bug report: the app crashes every time I tap Save on iPhone
```

It drafts a title, steps to reproduce, expected vs. actual result, severity, and platform, and it records the severity you confirm in the ticket text. Where your tracker and your project offer a native priority or severity field, BugIt sets that too. GitHub, GitLab and Trello have no such field. Want changes before filing? Just say so, for example "make the severity high" or "add that it started in the latest build."

Quick bugs (fast filing)

For common shapes (crash, error, missing, layout, slow, stuck, data, visual, audio, blocker), the quick form skips some questions and fills category and priority for you, and skips the duplicate search to save the round trip. Use it when you already know the issue is new; the preview says so, and a standard report still runs the full duplicate check.

YOU TYPE (EXAMPLES)

```
Quick bug: crash on startup after update
Quick bug: layout button overlaps text on the settings screen
Quick bug: blocker can't continue past the payment step
Quick bug: slow the dashboard takes 20 seconds to load
```

Crash bug reports

If you connected a crash tool (like Sentry), BugIt does everything above plus matches your report to the crash and pulls the stack trace for you.

YOU TYPE

```
Write a crash bug report: game crashes when opening the map
```

Verify & close comments

After you test a fix, BugIt writes a tidy comment to post on the ticket. It writes (and optionally posts) the comment; it does not change the ticket's status. You still close/resolve/reopen the ticket yourself in your tracker.

YOU TYPE	WHAT YOU GET
`Close #123`	Asks which scenario (fix confirmed, closed per request, as-designed, reopen...) then writes that comment.
`Write a verification comment for the build you tested on Windows`	A "fix confirmed" comment with the build details filled in.
`Write a reopen comment for the build you tested on Windows`	An "issue still happens" comment (then you reopen the ticket yourself).

Translation

Translate reports or terms between your configured languages, using your team's exact wording (from your glossary; see Part 3).

YOU TYPE (EXAMPLES)

```
Translate to ja: the save button does nothing on the profile screen
Translate this bug report to English: [paste text]
```

Look things up (specs & glossary)

If you connected specs or filled your glossary, ask BugIt questions about your project.

```
YOU TYPE (EXAMPLES)
What is [your term]?
Walk me through the checkout flow
What's new in specs?
```

Duplicate detection

This happens automatically on a standard report, before anything is written: BugIt searches your tracker for matching tickets (even ones worded a bit differently) and warns you before anything is written, so you decide. It is a check rather than a guarantee: the search reads a bounded number of tickets, so a duplicate it does not find is one you can still file. Quick Bug deliberately skips the search to save the round trip, and says so in its preview, so you always know which of the two you got.

Power features (if you turned them on)

TYPE THIS	WHAT IT DOES
`Triage digest`	Instant standup: open bugs by area/severity, oldest flagged.
`Export bug to file`	Saves the report as Markdown/CSV/JSON instead of filing (great before your tracker is set up).
`Undo last filing`	Shows the last write in the audit log. BugIt cannot delete or close a ticket; after a fresh preview and `FILE IT` it can remove only a comment or attachment it created.
(automatic)	SLA flags on urgent bugs · category-specific fields.

The full command list

```
YOU TYPE
help
```

...shows every command inside the agent, anytime.

> **Practise safely anytime**
>
> Type `dry run` and BugIt writes the entire report **without** filing anything, perfect for learning or testing.

Commands at a glance

You can always describe what you want in your own words. These are the exact phrases BugIt recognizes, if you prefer them.

SAY THIS	WHAT HAPPENS
Filing a bug	

SAY THIS	WHAT HAPPENS
`Quick bug: [type] [subject]`	File fast. Skips the duplicate search on purpose, and says so in the preview.
`dry run`	Write the whole report and file nothing. The safe way to practise.
`FILE IT`	The only thing that files. Typed as your whole message, once, for the draft you just read.
`Export bug to file`	Save the report as a file instead of sending it to your tracker.
`Undo last filing`	Show the last write, and remove a comment or attachment BugIt added. It cannot close a ticket. Needs the audit log switched on.
`Triage digest`	A summary of what you have filed recently.
Your project's knowledge	
`What is [term]?`	Look a term up in your glossary.
`What's new in specs?`	Recent changes in the spec folder you pointed BugIt at.
`Look up crash [ID]`	Search your crash tool and return links only. No report is written.
`Look up #[ticket]`	Search your tracker and return links only. No report is written.
`Seed glossary: [text]`	Suggest glossary terms from a block of text. You approve each one.
`Learn from my recent tickets`	Pick up your house style from bugs already in your tracker.
Setup and health	
`Begin setup`	Full first-run setup, or change one choice later.
`Check status`	Are my tools connected?
`Check readiness`	What is left before I can file?
`Add a custom MCP server`	Connect a tool that is not built in.
`Fix servers`	Diagnose and repair MCP connections.
`Check saved credentials`	Names only. BugIt never shows a saved credential's value.
`Talk to me in [language]`	Change the language BugIt chats in on this machine. Separate from the language your reports are written in.
`help`	The short version of this table, in the chat.
License, updates and backups	

SAY THIS	WHAT HAPPENS
`Activate`	Authorize this machine in your browser. No key and no password are ever pasted into the editor.
`License status`	Your plan, your entitlement, and your offline state.
`What version am I on?`	Your version, and whether an update is waiting.
`Update`	Install the latest signed version in place. Your settings are kept, and backed up first.
`Back up my settings`	Save config, glossary, house style and MCP setup locally. Credentials are not included.
`List backups`	See the snapshots you can go back to.
`Restore my settings`	Roll back to one of them.
`Create support bundle`	A sanitized diagnostics file to attach to a support email. No credentials, no tickets, no paths.
Team	
`Authorize this Team device`	Approve this machine on your own BugIt account, in your browser.
`Manage my devices`	Opens the BugIt Portal, where a device is signed out and its seat freed.
`Manage my Team`	Opens the BugIt Portal, where people, roles, seats and licenses are managed.

PART 3 · MAKE IT YOURS: ADD YOUR PROJECT'S KNOWLEDGE

Out of the box BugIt is a blank-slate QA agent. Here's how to teach it your project after purchase. No coding, mostly just talking to it in chat. Everything you add stays on your computer.

1 · Your team's words (the glossary)

This makes translations and category-guessing accurate for your project.

1. In the file list, open `.github/glossary/terms.template.md`.
2. Fill in the table with your team's words: feature names, screen names, item names, abbreviations, and (if you use two languages) their translations.
3. Save the file (`Ctrl+S`). That's it; BugIt uses these exact terms from now on.

Shortcut

Turn on "glossary auto-seed" during `Begin setup` and BugIt suggests terms from your existing tickets, and you just approve each one.

2 · Your bug style (house style)

Want tickets to match your team's exact format: title style, severity labels, required fields? You don't describe it; you **show** it:

1. In the chat, say: "match my team's style."
2. Paste **one screenshot** of an existing ticket from your tracker.
3. The AI model already in your chat reads it under its own provider's terms, BugIt redacts common personal data from the text it extracts, as a best effort, and your format is saved and followed on every future ticket.

3 · Your categories, platforms & fields

Just tell BugIt in plain language and it updates your settings for you:

```
YOU TYPE (EXAMPLES)
My categories are Gameplay, UI, Audio, and Networking
We test on Windows and Xbox
```

4 · Your own tools (custom connections)

Use a tool that isn't in the built-in list? Connect it just like a built-in:

```
YOU TYPE
Add a custom MCP server
```

Give it a short name and its address (must be `https://`); BugIt connects and health-checks it for you.

5 · Just correct it as you go

The easiest way of all: when BugIt drafts a ticket, fix anything you don't like: a label, the wording, the severity. With "learn from your edits" turned on (recommended), it remembers your preference and applies it next time. Your project knowledge builds up naturally on your machine, and none of it is ever sent to Taskivator.

Everything you add remains yours

Your glossary, house style, categories, and learned preferences are stored on your computer and backed up before every update. Relevant context may be sent to the AI model and tools you connect, when BugIt carries out your request. None of it is ever sent to Taskivator, and a share bundle leaves your machine only when you choose to send it.

6 · Sharing your setup with your team (Team license)

Once you've set up your glossary, house style, and tracker choices the way you want, share that setup with the rest of your team instead of repeating steps 1-3 on every machine:

YOU RUN (ONCE, AFTER YOUR SETUP IS THE WAY YOU LIKE IT)

```
python tools/share.py export
```

That writes `config.share.zip`: your tracker/crash/comms choices, glossary, and house style, bundled together with no passwords or tokens inside. Send it to your team.

EACH TEAMMATE RUNS FIRST, TO PREVIEW

```
python tools/share.py import config.share.zip
```

This shows what would change and writes nothing.

AFTER REVIEWING, TO APPLY THE SETTINGS

```
python tools/share.py import config.share.zip --apply
```

This applies your tracker, crash, and comms settings, with no re-typing categories. To also pick up your glossary and house style, review them, then choose `python tools/share.py import config.share.zip --apply --apply-instructions` instead. Their own license and device settings are untouched, and no credentials travel in the bundle.

> **If the import changes where data is sent**
>
> If a shared setup would add a new custom connection or notification address, the preview shows exactly what changed before anything is applied. Applying also snapshots the teammate's current setup first, so `Restore my settings` undoes it if it's not what they expected.

PART 4 · UPDATES, BACKUPS & SAFETY

Getting updates (one command)

When a newer version is available, BugIt mentions it once when you say `hi`. Software updates are included while your BugIt license is active. To install it:

YOU TYPE

```
Update
```

1. BugIt takes a fresh backup of your files first.
2. It downloads the new version and checks it's genuine and untampered (it's cryptographically signed, so a fake or altered update is refused).
3. It installs it without touching your settings, glossary, house style, license, or tokens.

4. It asks you to reload VS Code (`Ctrl+Shift+P` → type "Reload Window" → Enter). Start a fresh chat and you're on the new version.

> **No folder deleting, ever**
>
> Unlike the first install, updates are in-place. You never delete or re-download anything, and your setup is always preserved and backed up.

What's safe to hand-edit, and what isn't

Safe, and kept through every update: `config.json`, `.github/instructions/house-style.instructions.md` (see Part 3; this is the supported way to customize behavior), `.github/glossary/terms.template.md`, `.github/instructions/learned.instructions.md`, and `.vscode/mcp.json`.

> **Not safe: don't hand-edit these**
>
> The agent file (`.github/agents/bugit-qa-agent.agent.md`), any other file under `.github/instructions/`, and anything in `tools/` are the product itself, not your settings. An update **silently overwrites** them, and, unlike the files above, **there is no backup to restore them from**. BugIt checks for this at startup and again right before installing an update, and warns you if it finds a change, but the safest rule is simply not to edit them.

Backups & restore

YOU TYPE	WHAT IT DOES
`Back up my settings`	Saves a local snapshot of your config, glossary, house style, MCP endpoint settings, and learning. Your signed device entitlement and credential values are excluded.
`List backups`	Shows every saved snapshot with its date.
`Restore my settings`	Rolls back to a snapshot (and snapshots your current state first, so a restore is itself undoable).

> **Automatic before every update**
>
> BugIt always backs up before installing a new version, so an update can never lose your setup. If your settings look off after an update, `Restore my settings` puts them right.

Safety: what's protected

Nothing without your FILE IT
Every ticket and comment is previewed; anything irreversible needs you to reply with exactly `FILE IT`: your whole message, tied to the draft you just read, and usable once.

Dry-run = practice mode
Say `dry run` and every external ticket, comment, edit, attachment, removal and notification is suppressed (explicit local commands such as backup or export can still create local files) and your tracker is not contacted at all, reads included: no search, no duplicate check and no connection check, because each of those also reaches your board and unlocks your saved token. Setting `QA_AGENT_DRY_RUN=1` in your terminal is the switch the bundled commands enforce in code; saying "dry run" in chat asks the assistant to hold off, which is useful but is not the same guarantee.

No secrets in files
Tokens live in your OS vault, never in a plain file, never sent to us.

Runs on your machine
Your work goes only to the tools you connect and your own AI model.

Your data & privacy

The only thing BugIt sends to Taskivator is license/update data: your BugIt account sign-in (in the browser, on the Portal), a one-way hashed device ID (a scrambled code that can't identify you), a random installation identifier, your device label (hostname) and operating-system name, the BugIt version and the plan filter you pick, short-lived cryptographic challenge material, the one-time approval token from the Portal link your browser opened, a per-activation acknowledgement secret (its hash, and the value itself only if you withdraw this device's access), and the Solo or Team entitlement you approve for this device. There is no license key, and your password stays in the browser and is never entered into VS Code. Your bug reports, specs, glossary, settings and tokens are never sent to Taskivator. What you file does go where you send it: your own tracker, in your own account, using your own token, exactly as if you had typed it there yourself. If you use a hosted AI model, your draft reaches that provider too, the same way anything else you type into it does. No telemetry, no analytics, ever. Full details are in the `PRIVACY.md` file in your BugIt folder.

Risks & important cautions: please read

AI can be wrong
BugIt drafts reports with an AI model, which can misread details or invent something. Always read the preview before you reply `FILE IT`. You are approving what gets filed.

It files to your real tracker
Once you approve, a real ticket is created. Practising? Use `dry run` to write the report without filing.

Your project's safety is yours
How you use BugIt, and the safety of your projects, data, and trackers, are your responsibility. The safeguards reduce risk but don't replace your review.

What BugIt doesn't do

BugIt files to all eleven trackers it names, each with a credential you create in your own account, which BugIt checks against your chosen destination before saving it. What reaches a sortable priority field differs per tracker, and GitHub, GitLab and Trello have no such field, so there the severity is written into the ticket text. Confluence, Sentry and Notion are read only; an MCP server you set up with BugIt's help is needed only for those. Slack, Teams, Discord and email notifications are built in. It uses your AI (Copilot or your own OpenAI/ Anthropic key), not a bundled model; redaction is best-effort; and it runs in VS Code, in the Claude extension, and in a plain terminal. One license = one device (Solo) or 5 member seats, one device each (Team); a Team can hold more than one Team license, and their seats add up. Results and consistency also depend on which AI model is answering: a stronger model follows the safety steps more reliably than a very lightweight or free one.

PART 5 · TROUBLESHOOTING & FAQ

When something goes wrong, first describe it to the assistant in plain language. Most setup and everyday-use issues can be diagnosed directly in the editor.

Try this first: ask the AI to fix it

Whenever anything goes wrong (during setup or everyday use), your quickest fix is almost always to **ask the assistant right in the chat**. Describe what happened in plain words (for example, "the setup stopped halfway" or "my tracker won't connect") and ask it to fix it. The AI can diagnose and repair most problems on the spot. Please give this a try before emailing us; it resolves the large majority of issues in seconds.

Applying an AI fix: the "Keep" button

If the assistant fixes something by changing a file, VS Code shows a small Keep / Undo bar. If you asked for the fix and you're happy with it, click **Keep**, and the change is applied to your own copy only, on your computer. Click **Undo** to discard it. Nothing you Keep is shared with anyone else.

WHAT YOU SEE	WHAT TO DO
"Activate to start"	Type `Activate`. BugIt opens the BugIt Portal in your browser. Sign in, choose your Solo or Team entitlement, and approve this device. No account yet? Get one at the store.
The browser didn't open, or activation didn't finish	Type `Activate` again to reopen the BugIt Portal, then finish signing in and approving this device. Leave the command running until it says the authorization is saved on this computer. There is no key to paste.
"No tracker enabled"	Type `Begin setup` and pick your tracker.
Filing is refused	Run `python tools/connect.py jira --check`. It reports who the tracker thinks you are and what BugIt may create, and the `fixes` list names the actual remedy.
It keeps asking you to activate	Type `Activate` and finish signing in on the Portal in your browser. BugIt never displays saved credential values. (First time? See Step 8c.)

WHAT YOU SEE	WHAT TO DO
It won't file a bug	Type `Check readiness`; it lists exactly what's missing (usually the tracker is not connected yet, or the device is not activated).
Something looks broken after an update	If your settings changed, type `Restore my settings`. If BugIt's own files are damaged, type `Repair my BugIt install` and check the preview first. If the files are fine and the new version still misbehaves, email support@bugit.dev.
The agent seems "off script"	Type `Read and follow all your given instructions`, or start a fresh chat. This can happen more often on a lightweight/free AI model; a stronger model in Copilot's model picker is more consistent.
Answers feel generic / not project-specific	Give it more context, or show it one existing ticket during `Begin setup` so it learns your style. Rerun `Begin setup` to refine.
You want to change a setup choice	Type `Begin setup` and say what to change (e.g. "add a tracker" or "start over") and it reopens just that. If setup was simply interrupted partway, a bare `Begin setup` resumes where it left off instead of restarting.

Built-in health checks

`Check status` (are my tools connected?) · `Check readiness` (what's left before I can file?). For a deeper check, open Terminal → New Terminal and run `python tools/selftest.py`.

Frequently asked questions

Do I need to know how to code?
No. If you can install an app and type a sentence, you can use BugIt. It does the technical parts for you.

Will it file bugs without me knowing?
Never. Every ticket and comment is previewed first, and an irreversible action happens only when you reply with exactly `FILE IT`. It is matched as your whole message, so a sentence that merely contains it does nothing; it is tied to the exact draft you read, so a later change means BugIt asks again; and it is used once, so a repeated command cannot file twice. BugIt can never delete a ticket, change its status, or change its assignee. To practise, say `dry run`: every external ticket, comment, edit, attachment, removal and notification is suppressed (explicit local commands such as backup or export can still create local files) and the agent refuses tracker writes, though that refusal follows BugIt's instructions rather than a platform/OS lock, so use read-only credentials when evaluating.

Do I have to start a server every time?
No. Filing uses the API token you saved with `python tools/connect.py`, so there is nothing to start. An MCP server is only ever needed for the optional read-only integrations such as Confluence, Sentry and Notion.

Can I use it on two computers?
One device at a time on Solo. On a Team, each member has their own account and their own device, and the number of seats comes from the Team licenses the Team holds: every Team license adds 5. Moving to a

new device is fine: activate the new one, and the old device stops being able to renew straight away. Its slot frees up as soon as the old device reconnects (confirming it has given up access) or its current 72-hour offline lease runs out, so there is no fixed waiting period. Type `License status` to check.

What if the licensing server is down?
You keep working. After a successful online activation, BugIt runs on a cached lease for up to 72 hours offline, so a server outage (or just a weekend with no internet) won't interrupt you within that window. After that it re-verifies online.

Do I need GitHub Copilot?
No. BugIt runs in any assistant that can run a command: Copilot Chat, the Claude extension in VS Code, or a plain terminal. Copilot is the easiest way. If you don't have it, run `python tools/setup_wizard.py` in the Terminal to set up, and BugIt can run standalone with your own OpenAI/Anthropic key (`python standalone/run.py`).

Is my data private?
Yes. The only thing BugIt sends to Taskivator is license/update data: your BugIt account sign-in (in the browser, on the Portal), a hashed device ID, a random installation identifier, your device label and operating-system name, the BugIt version and plan filter, short-lived challenge material, the one-time approval token from the Portal link your browser opened, a per-activation acknowledgement secret (its hash, and the value itself only if you withdraw this device's access), and the Solo or Team entitlement you approve for this device. No telemetry, ever. Your tickets themselves go only to the AI model and tracker you connect, never to us.

Can I edit BugIt's files?
You're meant to customize your parts: your glossary, house style, and settings (Part 3). Just don't disable the licensing/activation. If you hit a real bug in the tool itself, email support so it can be fixed for everyone in the next update.

Which bug trackers does it work with?
Eleven, and BugIt files to every one of them: Jira Cloud, Azure DevOps, GitHub Issues, GitLab Issues, Bugzilla, YouTrack, Linear, Shortcut, ClickUp, Asana and Trello. Each has guided setup, each writes with a credential you create in your own account, and each was tested against a real account rather than a stand-in. What differs is which extras land: BugIt does not upload files to GitHub, GitLab or Bugzilla and tells you before you try (GitHub offers no token-authenticated upload at all; GitLab and Bugzilla offer one BugIt has not certified), ClickUp can accept a file and cannot remove one, and GitHub, GitLab and Trello have no priority field, so there your severity is written into the ticket text. Pick yours during setup and switch anytime with `Begin setup`.

Will it catch duplicate bugs before I file?
Yes, on a standard report. Before anything is written, it searches your tracker for similar reports (even ones worded a little differently) and shows you the matches, so you can avoid raising the same bug twice. You still decide whether to go ahead. Quick Bug skips that search on purpose, to save the round trip, and its preview tells you so.

Can it write tickets in more than one language?
Yes. Add a second language during setup and every report is written in both, kept in separate blocks, using your glossary so product terms stay exact; it never improvises a translation.

Can I attach a screenshot?
Ask BugIt to attach them. It creates an evidence folder, opens it, and tells you the exact path; drop your files in there and say you are done. BugIt screens text files for credentials and stops rather than attach one, and it shows you a preview first. It cannot see inside an image, so check a screenshot yourself before you approve it. They are attached after their own `FILE IT`. A file pasted into the chat cannot be reached by BugIt, so it always offers you the folder instead.

Does it hide passwords and tokens in my reports?
When redaction is switched on, it redacts emails, tokens, and IP addresses in every draft before filing, as a best effort. You always see the full text in your preview first.

I filed a ticket by mistake. Can I undo it?
The preview and typed `FILE IT` stop most mistakes before anything is written. If you turned on the audit log, type `Undo last filing`: it shows what BugIt wrote and can take back a comment or attachment BugIt itself added. Closing or deleting the ticket is always your own action in your tracker, because no tracker offers a way for BugIt to do it.

Can it help me verify or close a bug?
Yes. Ask for a verification comment and it drafts a clear pass/fail note (in your languages) on the existing ticket, and you approve it before it posts.

Can I use it for more than one project?
Each BugIt folder is set up for one project's tracker. For another project, either re-run `Begin setup` to point it somewhere new, or keep a separate copy per project.

How do I get updates?
When a new version is out, BugIt mentions it as you start a chat. Type `update` and it installs in place: your glossary, house style, and settings are kept, and it backs them up first.

What happens when my license expires?
When your term ends, renew from your account; your device applies the renewed entitlement the next time it checks in online. The 72-hour offline lease above is separate; it only covers a temporary licensing-*server* outage, not an expired term. Type `License status` anytime to see where you stand.

If I renew or buy another license, do the days stack?
Days never stack. A renewal **replaces** your current entitlement; its term starts fresh from the day it takes effect, and unused days don't carry over, so renew near the end of your term rather than early. **Seats are different.** On a Team, a second Team license adds its 5 seats alongside the first, each for its own term, so two licenses give you 10 seats. See the Team questions below.

Team licenses: questions

Everything on this page happens in the **BugIt Portal** in your browser, on the **Team** page, not in the editor. If you bought a Solo license, you can skip this page.

What do I get with a Team license?
Five seats for one year, a 1-year license with no subscription and no auto-renew. Everyone on the Team signs in with their own BugIt account and activates BugIt on one device of their own: there is no shared key and no shared login. People, invitations, roles and seats are all managed on the Team page in the Portal.

How do I add someone?
Open the Team page and send an invitation to their email address; you can paste several addresses at once. Each invitation works only for the address it was sent to, so a forwarded link cannot be used by somebody else. A pending invitation holds a seat until it is accepted or cancelled, which is why a Team can be full before everybody has joined.

Who can do what?
The **owner** can do everything, including billing and handing the Team to someone else. An **admin** can invite and remove people and change the Team's shared defaults. A **member** uses BugIt and can leave whenever they like. The owner cannot be removed from the Team; ownership is transferred instead.

We need more than five people. Can I buy another Team license?
Yes. Buy a second Team license and add it to the same Team from the Team page. **The seats add up:** two licenses give you 10, three give you 15. There is no fixed maximum, because every seat comes from a license you own. Adding a license takes effect immediately.

If I add a second Team license, when does my Team expire?
Each license keeps its own end date and pays for its own 5 seats until then. Your Team keeps working while **any** of its licenses is live, so the Team's end date is the later of the two. When the earlier one ends, the Team loses the 5 seats that license was paying for; the Team itself does not close. A second license does not extend the first, so if one license has three months left and you add a fresh one-year license, you have 10 seats for three months and 5 seats for the nine months after that. Renew the earlier license to keep all 10. The Team page shows every license with its own end date, so you can always see which one runs out first.

What happens if we end up with more people than seats?
Before a license ends we write to the Team three times: 30 days, 7 days and 1 day ahead, so it is never a surprise. If a license does end while the Team is holding more seats than it pays for, pending invitations give up their seats first, and after that the people who joined most recently lose access. The owner is never removed, so there is always somebody who can renew. Nothing anyone has filed is deleted; only access to BugIt stops.

What if the person who owns our Team leaves?
The owner can nominate a **successor** on the Team page. If the owner then becomes unavailable, the successor asks to take over; everyone on the Team is told, and the change takes effect 7 days later unless the owner stops it first. An owner who is available can simply transfer the Team at any time, which asks them to confirm with two-factor authentication before anything changes.

Can the whole Team work the same way?
Yes. Set BugIt up the way you want it, run `python tools/share.py export`, and each teammate runs `import` on the file you send them (Part 3, section 6). The Team page also records the tracker, project, assignee and labels your Team has agreed on, so everyone can look them up in one place. **Never put a token or a password there:** everyone on the Team can read those settings. Real credentials are always saved per person with `python tools/connect.py`, on their own machine.

Somebody has left the Team. What happens to their seat?
Remove them on the Team page: their device is signed out and the seat is free straight away for your next invitation. A member can also leave by themselves. Either way their own BugIt account stays theirs, and only their access to your Team ends.

Where do receipts and renewal notices go?
To the owner's account address by default. On the Team page the owner can add a company name and a separate billing email, which is where the reminders and receipts for that Team then go. It does not change who owns the Team.

PART 6 · LICENSE & SUPPORT

License terms (plain-language summary)

The full, binding terms are in the `LICENSE` file in your BugIt folder, and that document governs. In short:

TOPIC	IN PLAIN WORDS
Licensed, not sold	You buy a license to use BugIt, not ownership. Taskivator keeps all rights.
Your seats	Solo = one device at a time. Team = 5 member seats per Team license, one device each, and more than one Team license can fund the same Team. Your account and entitlement are personal to you; don't share, resell, or publish them.
Revocation	Shared, refunded, charged-back, or misused access can be revoked.
Customize, but...	Edit your config, glossary, and templates freely, but don't disable or bypass the licensing/activation.
Your responsibility	How you use BugIt and your project's safety are yours. You review and approve every report before it's filed. The safeguards are aids, not guarantees.
"As is", liability	Provided as-is, no warranty. To the extent the law allows, liability is capped at what you paid, excluding indirect/consequential damages.
Term & law	A one-year license starting on the day you buy it (a 1-year license, no subscription, no auto-renew), governed by the laws of Japan. Refunds are handled through the store you bought from.
Days don't stack, seats do	A renewal replaces your current entitlement: its term starts fresh and unused days don't carry over, so renew near the end of your term. On a Team, a further Team license adds its seats for its own term; the Team runs until its last live license ends. When a term ends, renew from your account and your device re-authorizes on its next online check.

Two documents in your folder

`LICENSE` (full terms) and `PRIVACY.md` (privacy statement). By activating and using BugIt, you accept the LICENSE.

Getting help

Please check Troubleshooting and the FAQ above first; they cover almost everything. Then, in this order:

1 · Ask the AI to fix it
Your best first move: describe the problem in the chat and ask the assistant to fix it. If it changes a file and it looks right, click Keep to apply it (your PC only).

2 · Run a health check
`Check status` · `Check readiness` · or `python tools/selftest.py` in the Terminal.

3 · Restore
`Restore my settings` brings back your settings, glossary and house style; it does not undo an update. Damaged BugIt files: `Repair my BugIt install`.

4 · Reach a human
Only if the guide and the AI couldn't help: visit bugit.dev or email support@bugit.dev (support is handled in English). Setup trouble in your first 7 days? We'll get you running or help with a store refund.

Please keep confidential project information out of support emails

Every project is different, and much of your work may be confidential. If you do email us, please describe the BugIt problem in general terms only, and don't paste confidential code, bug details, specifications, screenshots, or data from your project. Taskivator is not responsible for any confidential information sent to us, and anything confidential we receive is deleted immediately. Wherever possible, ask the AI assistant to help first; it can resolve most issues right inside your editor.

Thank you for choosing BugIt.
File clean tickets, skip the duplicates, and get your time back, one bug at a time.

BugIt QA Agent · Setup & User Guide · © 2026 Taskivator. All Rights Reserved. · bugit.dev · The LICENSE and PRIVACY.md files included with your download are the governing documents.