Guide contents

Choose the scope of your first mod

Install and create a project

Learn the editor on a tiny change

Package and test locally

Release carefully

A complete first-project loop

Sources / References

Documentation: In progress **Quick start: In progress**

Offline copy from ModDocs. Source links remain clickable.

Baldur's Gate 3 modding

Larian's official Baldur's Gate 3 Toolkit is the clearest starting point for a new mod project. It has a project browser, editor, resource tools, local packaging, and a publishing route. The official modding documentation is the version-specific reference.

> **Quick start:** install the Toolkit using the current installation guide, create an empty project, make one small visible change, publish a local `.pak`, and test it in game before considering an online release.

Choose the scope of your first mod

Goal	Begin with	Check before publishing
Change or add a game resource	Toolkit resource manager	Overrides, dependencies, and game behavior
Add a custom visual asset	Toolkit asset import workflow	Mesh, texture, material, and in-game appearance

Goal	Begin with	Check before publishing
Edit a level	Toolkit level editor	Navigation, encounters, loading, and saves
Change story or gameplay logic	Relevant Toolkit scripting/system guide	Trigger path and compatibility with other content

Choose a small change you can reach within minutes of launching a test save. A complex quest or level is much harder to debug before you understand the project's resources and publishing flow.

Install and create a project

The Toolkit installation guide lists the current account, key, branch, and download steps. Follow those steps as written for your storefront and game build; do not use a random older Toolkit package. Once installed, launch it and point it to your BG3 `Data` directory when prompted.

From the Toolkit Browser, create a new project and give it a clear name. The create-a-new-mod guide walks through the initial project and Source Asset Data Path. Keep a record of your project location and the game build it targets. Open the default project and confirm you can navigate the editor before changing anything.

Learn the editor on a tiny change

The editor navigation guide introduces the main windows and how resources relate. Locate one safe resource to alter, make a deliberately obvious change, and save the project. If you intend to replace an existing resource, understand whether you are overriding a shared asset; an override can affect more places than the one scene where you noticed it. Keep a copy of the untouched project or use version control so you can compare the effect of each edit.

For custom assets, start with the official asset specifications and import path linked from the documentation home. Test the asset in the editor and in game. A successfully imported mesh can still have missing materials, wrong scale, or a bad reference in the actual level.

Package and test locally

The Toolkit's **Publish Local** action creates a `.pak` without uploading it. The publishing guide explains that route and the later mod.io path. Start with the local package, enable it through the game's supported mod workflow, then check your change in a new or appropriate test save. Make sure the package contains the resources you edited and that the result is visible without your editor project open.

If the mod works only while the Toolkit is open, suspect packaging or missing dependencies. If the project cannot open, revisit its game data path and required Toolkit version. If the game loads but the change is absent, check the resource override, project save, mod enablement, and the exact save or scene you are testing.

Release carefully

Give the mod a clear description, supported game version, dependencies, and installation notes. Test the local `.pak` before publishing through the official path. Keep a source backup and a record of what you changed; the Toolkit's import of an existing packed `.pak` is limited and should not be treated as a full project backup.

A complete first-project loop

Begin with the official project-creation walkthrough. Create a new project with a unique name and correct Source Asset Data Path, then make the smallest change supported by the resource type you choose. Save, close, and reopen the project before packaging; this catches unsaved or misplaced assets. Use **Project Settings** to fill in the required name, description, and thumbnail and declare dependencies. The official local-publishing instructions say **Publish Local** makes a `.pak` without mod.io authentication.

Move that `.pak` into a clean game test setup and enable it using the supported in-game mod flow. Check the altered resource in the exact save, location, or character context where it should appear. If the editor shows the change but the local `.pak` does not, compare the packed project with its resource dependencies and the game load order; avoid publishing online until the local artifact works. Finally, disable the

`.pak` and confirm the change disappears. That last check distinguishes the mod's effect from a game setting or another installed mod.

This loop completes project creation and delivery; the resource edit itself is specific to the chosen mod. Use the Toolkit's resource and editor guides for the exact fields. A universal item/quest/level recipe would be misleading because those assets have different validation paths.

Still being expanded: this page provides an end-to-end route through project creation and local packaging, but not yet tested recipes for each resource, quest, or level type. Use the official Toolkit docs for current editor commands and exact asset specifications.

Sources / References

External project and documentation links cited in this guide. Check the upstream release and version before following version-sensitive steps.

- modding documentation
- installation guide
- create-a-new-mod guide
- editor navigation guide
- publishing guide
- Toolkit's import of an existing packed `.pak`