Guide contents

Install the right editor

Create a plugin from a template

Keep testing separate from publishing

Troubleshoot the earliest failing stage

A complete first-project loop

What this guide does not finish yet

Sources / References

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

Offline copy from ModDocs. Source links remain clickable.

ARK: Survival Ascended modding

ARK: Survival Ascended uses the **ARK DevKit** to author mods. The current DevKit creates mods as plugins, provides templates for common content, and uploads builds for cloud cooking. These steps belong to ASA; do not mix them with older ARK: Survival Evolved tutorials. The official DevKit documentation is the reference for changing editor builds and publishing rules.

> **Quick start:** install the stable DevKit through Epic Games Launcher, create one small mod from a UGC template, change one asset in the plugin, verify it locally, upload for cloud cooking, then test the unpublished build through **My Mods** before publishing it broadly.

Install the right editor

The DevKit installation guide describes two distributions: a stable editor and a public beta. Stable is the safer starting point for a release because beta updates can arrive before the cloud-cooking service can process them. The guide also warns that the DevKit needs an SSD and a very large amount of free space. Check its **current** storage recommendation before downloading; this requirement may change as the tool is updated.

Install it through Epic Games Launcher and launch the DevKit itself. Keep a clean, separate project for experiments. Do not edit shipped game content directly as the distribution strategy; make a mod plugin so your changes can be enabled and tested independently.

Create a plugin from a template

In the DevKit, open **UGC** and create a new mod. Pick the closest template for your goal, such as a structure or creature, enter its details, and use **Create Mod**. The official creation page says the template creates the initial plugin files. In UGC's Enabled Mods area, ensure your new project is checked; find its files under **Plugins** in the Content Browser.

Start with one deliberate edit and save it. If the asset is a creature or structure, follow the matching official template/tutorial rather than copying values from an unrelated category. Keep track of which assets belong to your plugin and which are references to base-game content. Reopen the project to verify the changed asset still resolves before adding more dependencies.

Keep testing separate from publishing

The development loop has different stages:

Stage	Check
Edit in DevKit	Plugin is enabled and its assets compile/load in the editor
Local test	The game behavior matches the one intended change
Cloud cook	Upload is processed for the selected target platforms
Unpublished test	The cooked build installs and runs from **My Mods**
Public release	Description, compatibility, and actual build have been checked

ASA's cooking and publishing guide states that mods and maps are cooked through a cloud service, not cooked locally for release. Use **UGC** → **Share Mod**, choose the project and platform scope, upload, and monitor the cooking job on its project page.

A beta DevKit can temporarily get ahead of cloud cooking; if an upload fails for that reason, check editor/service compatibility before changing your mod content.

The testing guide describes **My Mods** for an author to install an unpublished version in the game. This matters because an editor preview is not proof that the distributed cooked package works. Test the exact build you plan to publish. If you use a private test server, follow the guide's unpublished-version rules and restrict it to testers with access; public player servers should use public releases.

Troubleshoot the earliest failing stage

If the project does not appear in Content Browser, check whether the plugin is enabled in UGC. If an asset looks wrong in the editor, inspect its own dependencies and references before uploading. If the upload or cook fails, read the cloud job's status; a DevKit/cloud version mismatch is different from an asset error. If a cooked mod installs but fails in game, reproduce it with only that mod enabled and compare the exact uploaded build with the local editor state. Keep a short version/change record so you can identify the first bad upload.

A complete first-project loop

Use the **stable** DevKit and choose one of the UGC menu's current templates. A structure template is a reasonable starting point if you want an object you can place and inspect; a creature template has a different asset and gameplay test. Enter a unique mod name, create it, enable its plugin in UGC, and find its assets under Plugins. Save one obvious change within that generated plugin. The creation guide describes this plugin boundary and template setup.

Check the modified asset in the editor, then use **UGC → Share Mod** to upload it for cloud cooking. Record the upload/build identifier and wait for the target platform's cook to finish. In the game, install the **unpublished** build from **My Mods**, following the private testing guide. Start a fresh single-player test with only that mod enabled and confirm the chosen object or creature actually differs from the template. Finally, disable the mod and verify the difference disappears. This tests the cooked artifact, not just an editor preview.

Do not infer that an editor-only success is cross-platform success. Choose PC-only or cross-platform in the sharing flow, then test the built platform versions you intend to claim. The DevKit docs explain how to give designated testers access to unpublished builds; keep those builds on test accounts or test servers until the public release is ready.

What this guide does not finish yet

This page covers setup, plugin creation, cloud cooking, and the unpublished test loop. It does not yet provide a tested creature, structure, map, or Blueprint implementation from start to finish. Those depend on the chosen template and current DevKit build. Use the official DevKit docs for the relevant asset-specific instructions and version notices.

Sources / References

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

- official DevKit documentation
- DevKit installation guide
- official creation page
- cooking and publishing guide
- testing guide