Install an Arma 3 mod set with the correct client and server roles

An Arma 3 mod can affect the server, the player or both. Downloading a Workshop item does not choose its role or make it compatible with a mission. Build a small documented mod set before adding it to a community's active service.

1. Read the authors' requirements

Record each original Workshop ID, required dependency and supported game branch. Check the mission's requirements separately; an apparently optional content mod may be needed by its mission assets. Follow the author's instructions for client and server use rather than classifying a mod from its name.

Keep a known working client preset or equivalent written inventory. A shared preset is useful only when it refers to the intended items and versions. Do not distribute private paid content or account credentials with it.

2. Back up mission and persistence data

Keep the active startup configuration, server configuration, profiles, mission PBOs and any framework persistence together. Export an external database through its normal backup procedure if the mission uses one. Confirm which files actually belong to this instance before copying a directory from another service.

Use a separate test deployment for a framework or large mod pack. It must not write to the production database or integrations. A mission that creates a fresh campaign state is not a safe diagnostic against your existing persistence.

3. Stage complete mod directories

Use the original publisher's release or the supported Workshop deployment workflow. Keep the PBOs and their supplied signatures together. Mod directories often begin with `@`, but the important check is that the actual launch path matches the directory you deployed. Linux path spelling and case need particular care.

Do not merge files from unrelated mods into one anonymous folder. Preserve a manifest mapping each directory to its source ID and release. Ask support how the template stages Workshop files if only a numeric cache directory is visible.

4. Select the correct launch lists

`-mod` loads ordinary mods, while `-serverMod` identifies server-side content that is not broadcast to clients. Both accept semicolon-separated folder lists. This is illustrative startup syntax, not a complete command to replace your existing launch configuration:

```
-mod=@RequiredFramework;@RequiredContent
-serverMod=@ServerOnlyTool
```

Shell quoting and escaping vary by launcher and operating system. Preserve the existing template's handling of semicolons. Putting a client-required mod in the server-only list does not make its client dependency disappear.

5. Verify trusted signatures and gameplay

A publisher's public `.bikey` is used to verify signed addons; `.bisign` files accompany the addon data. Trust only keys obtained from the intended publisher. Review the instance's configured key locations rather than turning signature checks off to accept an unknown package. Never request or share a publisher's private signing key.

Start the test server, inspect its RPT output and join with the intended ordinary client preset. Test a playable mission, representative mod features and a normal save/restart. If it fails, preserve the first dependency or signature error and restore the complete test checkpoint. Do not strip production mods and save over campaign data as a shortcut.

Related CloudExa guides

- Configure an Arma 3 server and test a first mission connection
- Back up game files before making changes

Official references

- Bohemia Interactive: mod and serverMod startup parameters
- Bohemia Interactive: addon files, public keys and signatures
- Bohemia Interactive: Launcher and mod management

Revision #1
Created 2026-11-10 11:05:47 UTC by CloudExa Hosting Admin
Updated 2026-11-10 11:05:47 UTC by CloudExa Hosting Admin