Check QBCore resources, database readiness and boot order

A working QBCore server needs compatible core, character, inventory, database and supporting resources. Uploading `qb-core` by itself will not produce a complete roleplay server. Start by identifying which official recipe or release your installation follows, then compare the installed resources to that reference before changing anything.

Preserve the live installation

Take a resource and configuration backup together with a matching database export. Record custom resources and their supported QBCore versions. A fresh txAdmin deployment can import SQL and replace configuration, so use a separate test directory and database for experiments. Do not rerun a recipe against existing player data to resolve an isolated missing export.

The official upstream repository URLs currently resolve to the `qbcore-fivem` organisation; older links may use `qbcore-framework`. Follow the official project's redirects and recipe references rather than trusting a download purely because its filename says QBCore. Record actual commits or releases; mixed snapshots can disagree about exports and item definitions.

Make dependency readiness visible

Use your recipe's full configuration as the reference. For diagnosis, the essential relationship is database settings first, the database adapter ready, then core, then dependent resources. A deliberately simplified example is:

```
# Supply your private connection setting before this block.
ensure oxmysql
```

```
ensure qb-core
ensure [qb]
```

This does not replace the recipe's standalone, voice, map and default resources. Review those groups too. Ensure the adapter is present once under its required resource name and examine its first startup message. In a managed container the database can be on another endpoint; do not substitute `localhost` for the address supplied with your database.

The official recipe checked for this guide pins an oxmysql download rather than treating any fork's newest build as interchangeable. Follow the dependency reference for your chosen recipe version, and test upgrades separately. Do not assume a modern core will support every older `qb-` resource unchanged.

Trace errors from the beginning

A missing `GetCoreObject` export can result from a failed or incompatible core, a renamed folder or an outdated add-on. Check those causes before editing calls globally. Unknown item or job errors need comparison with the core's shared definitions and the add-on's instructions. A missing SQL table points to schema compatibility, not necessarily a broken network connection.

Keep framework administration separate from txAdmin roles. Do not copy owner-level command grants from a sample into ordinary player permissions. Fix the particular resource's access requirements without making every staff member a god or granting all console commands.

Verify characters and persistence

After a planned restart, join with a test account and complete character selection, spawn and a normal inventory action. Disconnect cleanly and reconnect to confirm persistence. Test an existing character after any approved migration; successful new-character creation can hide problems with older rows. Check the client F8 log as well as the server console when a loading screen hangs.

Send support the first error, core and add-on versions, database adapter release and the exact action that fails. Redact connection strings and identifiers. Restore files and database together if an approved change requires rollback, rather than mixing a previous schema with new resource files.

Related guides

- Configure the database adapter

- Test pma-voice proximity and radio

Further reading

- QBCore official core repository
- QBCore official txAdmin recipe
- QBCore recipe configuration
- QBCore recipe dependency downloads

Looking for another game? Browse game server setup and troubleshooting guides.

Revision #2
Created 2026-11-09 09:21:20 UTC by CloudExa Hosting Admin
Updated 2026-11-09 09:21:21 UTC by CloudExa Hosting Admin