**Strapi Data Persistence and Local Databases**

Strapi stores content entries and user/role data in its database (not in source code). By default, it uses a local SQLite file (.tmp/data.db) for development. If we clone a Strapi project without this database file, the new instance starts with an empty database—losing all content and roles. Since SQLite files are typically in .ignore, GitHub clones won’t include this data. Strapi’s docs and community confirm this behaviour.  
  
**How Strapi Stores Data**

**Content** (collections and singles) are stored in database tables. **Administration roles (RBAC) and user roles (Users & Permissions plugin)** are also stored in database. Strapi sets up default roles (e.g. *Super Admin*, *Editor*, *Author* for the admin panel and *Public*, *authenticated* for front-end users) on a fresh install, but thereafter any custom roles or permissions are saved in the DB. None of these appear in source code. **Content-type definitions (schemas)**: These are part of the code (/src/api/<type>/content-types/…), so they are tracked in Git. Changing a content type can trigger database migrations. However, the actual content is **not** in code.

**Best Practices to Export and Import Data.**

**Use Strapi’s Data Export/Import CLI:**

Starting in Strapi v4.6+, a built-in CLI strapi export and strapi import can snapshot your entire database (including content types, entries, roles, permissions, etc.) to a file.

**Data export from local strapi instance:**

Command: npm run strapi export -- *--file* my-strapi-export

Without encryption: npm run strapi export -- *--file* my-strapi-export --no-encrypt

(Admin users and API tokens and Media from 3rd party provider are not exported)

**Data import:**

strapi import delete all the existing data, including the database and uploads directory, before importing the backup file.

Command: npm run strapi export -- *--file* my-strapi-export.tar.gz.enc

(will need encrypt key if required)

**Data transfer:**

The default transfer command transfers your content (entities and relations), files (assets), project configuration, and schemas. The command allows you to transfer data:

* from a local Strapi instance to a remote Strapi instance
* from a remote Strapi instance to a local Strapi instance

**Generate a transfer token**

A salt transfer token should be defined in the [**admin panel configuration**](https://docs.strapi.io/cms/configurations/admin-panel) file.

The strapi transfer command requires a transfer token issued by the destination instance.

npm run strapi transfer -- *--to* destinationURL (Push data to remote)

npm run strapi transfer -- *--from* remoteURL (Pull data from remote)

(Note: The URL should include the /**admin** part. Add the transfer token when prompted to do so.)

*The transfer will also delete the existing database contents.*

**Include only specified data types during transfer**

**Example: Only transfer files**

npm run strapi transfer -- --to https://example.com/admin --only files

(--only allows us to transfer only the listed items)

**Exclude data types during transfer**[**​**](https://docs.strapi.io/cms/data-management/transfer#exclude-data-types-during-transfer)

**Example: exclude files from transfer**

npm run strapi transfer -- --to https://example.com/admin --exclude files

(The --exclude option allows us to exclude content, files, and the project configuration)

Key Differences between Import/Export and Data Transfer

**Import/Export (Built-in Feature)**

**What it does**

**Export**: Generates a .tar or .zip file containing:

* + **Content schema** (content-type structures, components, relations).
  + **Content data** (entries for collections/single types).
  + **Configuration** (core settings, roles, permissions).

**Import**: Restores the exported bundle into another Strapi instance.

**Use Cases**

* Migrate an **entire project** (schema + data) between environments (e.g., dev → prod).
* Backup/restore a Strapi instance.

**Limitations**

* Not designed for partial transfers (e.g., single content types).
* Requires identical Strapi versions for compatibility.

**Data Transfer (Custom/API-Based)**

**What it does**:

Moves **specific data** (not schema/config) via Strapi **REST/GraphQL APIs**, **Database tools**(e.g., direct SQL/NoSQL exports) and **Custom scripts** (e.g., using axios to sync data between instances).

**Use Case:**

* Sync **content only** (e.g., update products from a CSV).
* Partial transfers (e.g., migrate only "Blog Posts" content).
* Integrations with external systems (e.g., CMS → mobile app).