Melos in Flutter

Handling multiple Flutter and Dart packages

Flutter Developer

@ Ahmed Hussein
01114484229

What is Melos?

Melos is a command-line **tool** designed to manage **monorepos** containing multiple Dart/Flutter packages. It helps automate tasks such as dependency management, versioning, package linking, and running scripts across multiple packages at the same time.

When to Use Melos?

- You're working on a **monorepo** with multiple Dart/Flutter packages.
- You need to **automate tasks** like testing, linting, or building across packages.
- You want to **streamline** versioning and publishing workflows.
- Your team collaborates on **multiple interdependent packages**.

Installing Melos

```
dart pub global activate melos
```

@ Ahmed Hussein
01114484229

Setting Up a Monorepo with Melos (Use Case)

Two user roles (Driver & Client) inside your **single Flutter app**:
Melos helps manage multiple packages/modules inside a single repository. This allows you to:
✅ Separate Driver an d Client logic into different packages.
✅ Share common utilities (e.g., authentication, API handling).
✅ Improve scalability for large applications.
✅ Enable independent updates for different features.

```
// ride_app/
// ├── core/ ========> # shared code between all packages
// │   ├── lib/
// │   │   ├── constants.dart
// │   │   ├── utils.dart
// │   │   └── shared_widgets.dart
// │   └── pubspec.yaml
// ├── auth/ ========> # authentication package for login and signup
// │   ├── lib/
// │   │   ├── login_screen.dart
// │   │   ├── signup_screen.dart
// │   │   └── auth_service.dart
// │   └── pubspec.yaml
// ├── client/ ========> # client package for client related features
// │   ├── lib/
// │   │   ├── home_screen.dart
// │   │   └── client_service.dart
// │   └── pubspec.yaml
// ├── driver/ ========> # driver package for driver related features
// │   ├── lib/
// │   │   ├── home_screen.dart
// │   │   └── driver_service.dart
// │   └── pubspec.yaml
// ├── app/    ========> # main app package that uses all other packages
// │   ├── lib/
// │   │   └── main.dart
// │   └── pubspec.yaml
// └── melos.yaml   ========> # melos configuration file for managing packages
```

Flutter Developer

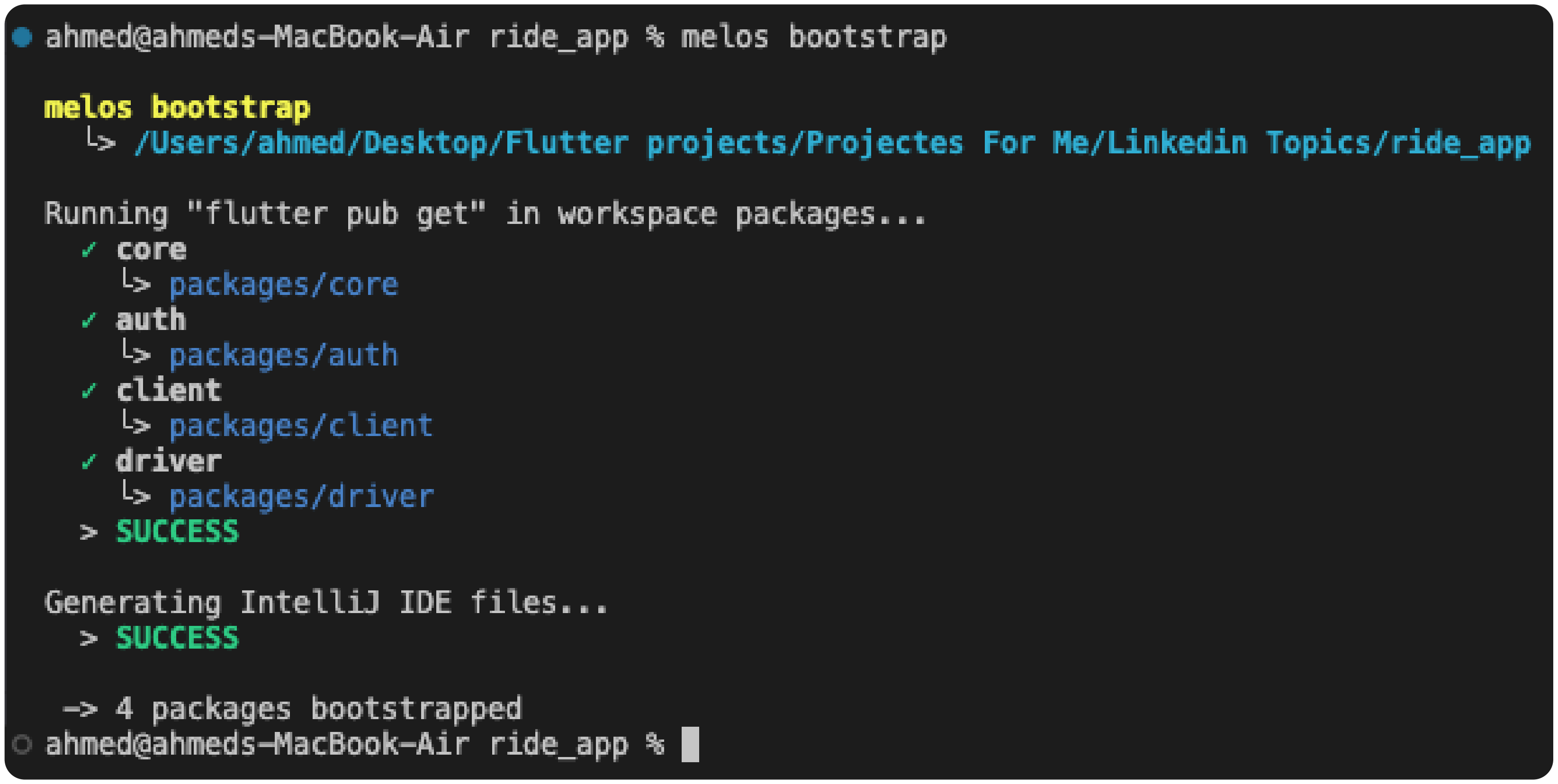
@ Ahmed Hussein
01114484229

Importance of melos.yaml

melos.yaml is essential for Melos to properly manage and link all packages together in a monorepo structure.

Melos Bootstrap

Now run this command: **melos bootstrap** to link the local dependencies between packages automatically.
You will be seeing the following output:



```
● ahmed@ahmeds-MacBook-Air ride_app % melos bootstrap

  melos bootstrap
    └> /Users/ahmed/Desktop/Flutter projects/Projectes For Me/Linkedin Topics/ride_app

  Running "flutter pub get" in workspace packages...
    ✓ core
      └> packages/core
    ✓ auth
      └> packages/auth
    ✓ client
      └> packages/client
    ✓ driver
      └> packages/driver
    > SUCCESS

  Generating IntelliJ IDE files...
    > SUCCESS

   -> 4 packages bootstrapped
○ ahmed@ahmeds-MacBook-Air ride_app % █
```

Ahmed Hussein
Flutter Developer

THANK YOU

Was This Helpful?

Like, Share and Follow
for more Content

Like Comment Share Save