Hono Monorepo Project Structure

Multiple Workers sharing packages. Ideal for edge microservices and platform APIs.

#hono #typescript #monorepo #cloudflare #workers #turborepo

PNG PDF Copy Prompt

Project Directory

```
myproject/
  apps/  Individual Work…
    api/  Main API Worker
      src/
        index.ts
        routes/
      wrangler.toml
      package.json
    auth/  Auth Worker
      src/
        index.ts
      wrangler.toml
      package.json
    webhook/  Webhook handler…
      src/
        index.ts
      wrangler.toml
      package.json
  packages/  Shared code
    shared/  Common utilities
      src/
        index.ts
        middleware/  Shared middlewa…
        schemas/  Shared Zod sche…
        types/
      package.json
      tsconfig.json
    db/  Database package
      src/
        index.ts
        schema.ts  Drizzle schema
      package.json
  turbo.json  Turborepo config
  package.json  Workspace root
  pnpm-workspace.yaml
  tsconfig.json  Base TypeScript…
  .gitignore
```

Why This Structure?

Multiple Hono Workers sharing code via internal packages. Each Worker deploys independently to its own route or subdomain. Turborepo caches builds. Share middleware, schemas, and database code without duplication.

Key Directories

`apps/` - Each folder is a deployable Cloudflare Worker

`packages/shared/` - Middleware, types, and utilities used by all Workers

`packages/db/` - Drizzle schema and database utilities

`turbo.json` - Build pipeline and caching configuration

Worker Using Shared Packages

```
// apps/api/src/index.ts
import { Hono } from 'hono'
import { logger, errorHandler } from '@myproject/shared'
import { db } from '@myproject/db'

const app = new Hono()
app.use('*', logger())
app.onError(errorHandler)

app.get('/users', async (c) => {
  const users = await db.query.users.findMany()
  return c.json(users)
})

export default app
```

Getting Started

1. `pnpm create turbo@latest`
2. Add Hono to each app in `apps/`
3. Create shared packages in `packages/`
4. `pnpm install`
5. `pnpm dev` to run all Workers

When To Use This

- Platform with multiple APIs or subdomains
- Microservices on Cloudflare Workers
- Shared business logic across services
- Teams owning separate Workers
- Gradual migration from monolith

Trade-offs

`Complexity` - Monorepo tooling adds cognitive overhead

`Build coordination` - Package changes affect downstream Workers

`Local dev` - Running multiple Workers locally needs setup

Best Practices

- Use `@myproject/` namespace for internal packages
- Keep Workers focused—split by domain, not size
- Share Zod schemas between Workers for RPC validation
- Use Cloudflare Service Bindings for Worker-to-Worker calls
- Pin shared package versions during active development

Naming Conventions

`Apps` - Named by domain: `api`, `auth`, `webhook`

`Packages` - Named by purpose: `shared`, `db`, `email`

`Internal imports` - `@myproject/{package}` namespace