# Contributing

Prisma consists of a mono-repo for all TypeScript code.

To setup and build the packages, follow these steps:

```bash

git clone https://github.com/prisma/prisma.git

cd prisma/src

npm i -g pnpm@5.1.7

pnpm i --ignore-scripts

pnpm run setup

```

Note for Windows: Use the latest version of [Git Bash](https://gitforwindows.org/)

### Developing Prisma Client JS

2. `cd src/packages/client`

3. `ts-node fixtures/generate.ts ./fixtures/blog/ --skip-transpile`

4. `cd fixtures/blog`

5. `prisma migrate save --name init --experimental && prisma migrate up --experimental`

6. `ts-node main.ts`

### Working on code generation

If you have your local blog fixture running, you can now do changes to `TSClient.ts` and re-execute `npx ts-node fixtures/generate.ts ./fixtures/blog/`.

When doing changes and working on a fixture use `yarn build && rm -rf fixtures/blog/node\_modules/ && ts-node fixtures/generate.ts fixtures/blog`

### Working with the runtime

If you want to use the local runtime in the blog fixture, run

```sh

ts-node fixtures/generate.ts ./fixtures/blog/ --local-runtime

```

Changes to `query.ts` will then be reflected when running `fixtures/blog/main.ts`

### Developing Prisma Migrate

1. `cd src/packages/migrate/fixtures/blog`

2. `ts-node ../../src/bin.ts up`

### Developing `prisma init` Command

1. `cd src/packages/introspection`

2. `mkdir test && cd test`

3. `ts-node ../src/bin.ts`

### Developing `@prisma/cli` CLI

1. `cd src/packages/prisma2`

2. `mkdir test && cd test`

3. `ts-node ../src/bin.ts generate`

### How to update all binaries

```bash

# In the root directory

pnpm run download

```

### Running the CI system locally

```bash

cd src/.buildkite/test

docker-compose up -d

docker-compose logs -f app

```

### Git Commit Messages

We structure our messages like this:

```

<type>(<package>): <subject>

<BLANK LINE>

<body>

```

Example

```

feature(client): new awesome feature

Closes #111

```

List of types:

- feat: A new feature

- fix: A bug fix

- docs: Documentation only changes

- style: Changes that do not affect the meaning of the code (white-space, formatting, missing semi-colons, etc)

- refactor: A code change that neither fixes a bug nor adds a feature

- perf: A code change that improves performance

- test: Adding missing or correcting existing tests

- chore: Changes to the build process or auxiliary tools and libraries such as documentation generation

List of packages:

- cli

- client

- engine-core

- fetch-engine

- generator-helper

- get-platform

- ink-components

- migrate

- sdk

- introspection