# Supabase

This guide discusses the concepts behind using Prisma ORM and Supabase, explains the commonalities and differences between Supabase and other database providers, and leads you through the process for configuring your application to integrate with Supabase.

## What is Supabase?

Supabase is a PostgreSQL hosting service and open source Firebase alternative providing all the backend features you need to build a product. Unlike Firebase, Supabase is backed by PostgreSQL which can be accessed directly using Prisma ORM.

To learn more about Supabase, you can check out their architecture here and features here

## Commonalities with other database providers

Many aspects of using Prisma ORM with Supabase are just like using Prisma ORM with any other relational database. You can still:

- model your database with the Prisma Schema Language
- use Prisma ORM's existing `postgresql` database connector in your schema, along with the connection string Supabase provides you
- use Introspection for existing projects if you already have a database schema in Supabase
- use `db push` to push changes in your schema to Supabase
- use Prisma Client in your application to talk to the database server at Supabase

## Specific considerations

If you'd like to use the connection pooling feature available with Supabase, you will need to use the connection pooling connection string available via your Supabase database settings with `?pgbouncer=true` appended to the end of your `DATABASE_URL` environment variable:

.env

```
# Connect to Supabase via connection pooling with Supavisor.
DATABASE_URL="postgres://postgres.[your-supabase-project]:[password]@aws-0-eu-central-1.pooler.supabase.com:6543/postgres?pg
```

If you would like to use the Prisma CLI in order to perform other actions on your database (e.g. migrations) you will need to add a `DIRECT_URL` environment variable to use in the `datasource.directUrl` property so that the CLI can bypass Supavisor:

.env

```
# Connect to Supabase via connection pooling with Supavisor.
DATABASE_URL="postgres://postgres.[your-supabase-project]:[password]@aws-0-eu-central-1.pooler.supabase.com:6543/postgres?pg
# Direct connection to the database. Used for migrations.
+ DIRECT_URL="postgres://postgres:[password]@db.[your-supabase-project].supabase.co:5432/postgres"
```

You can then update your `schema.prisma` to use the new direct URL:

**schema.prisma**

```
datasource db {
  provider  = "postgresql"
  url       = env("DATABASE_URL")
+ directUrl = env("DIRECT_URL")
}
```

More information about the `directUrl` field can be found here.

> We strongly recommend using connection pooling with Supavisor in addition to `DIRECT_URL`. You will gain the great developer experience of the Prisma CLI while also allowing for connections to be pooled regardless of your deployment strategy. While this is not strictly necessary for every app, serverless solutions will inevitably require connection pooling.

## Getting started with Supabase

If you're interested in learning more, Supabase has a great guide for connecting a database provided by Supabase to your Prisma project available here.

If you're running into issues integrating with Supabase, check out these specific troubleshooting tips or Prisma's GitHub Discussions for more help.

Pulse EARLY ACCESS

Pricing

RESOURCES

Docs

Get started

prisma-examples ↗

Prisma ORM in your stack

Ecosystem

Tutorials

Playground ↗

Customer stories

Data guide

Data Platform status ↗

VS Code extension ↗

CONTACT US

Community

Support

Enterprise

COMPANY

About

Blog

Data DX ↗

Careers

Ask AI

Events

Causes ↗

OSS Friends

Terms & Privacy ↗

Service Level Agreement ↗