Supabase

CS 346 Application Development

Local vs. Cloud Hosting

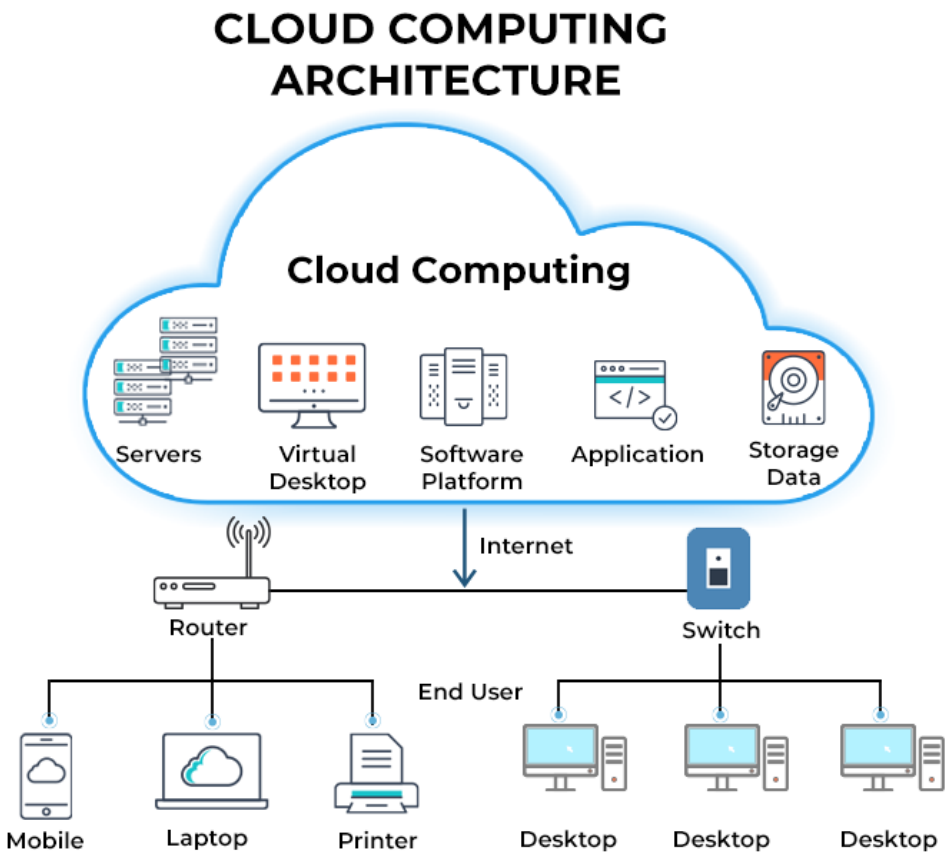
Rosenberg & Mateos (2014) define cloud computing as

"a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services)"

-- https://en.wikipedia.org/wiki/Cloud_computing.

Although you *can* install and host your own database, it's very common to rely on a host provider i.e., a third-party to manage it for you. This provides:

- **Flexibility**: you can deploy new hardware with a button-click.
- **Scalability**: you can grow your hardware with demand e.g., scale up during peak season, scale down when demand shrinks.
- **Security**: robust security can be challenging to manage in-house.



Spiceworks 2024.

Cloud providers offer a range of Services:

- A **range of preconfigured services** including application hosting, and data storage.
- **Pooled computing resources** available to any subscribing users.
- **Virtualized computing resources** to maximize hardware utilization.
- **Elastic scaling** up or down according to need.
- **Automated creation** of new virtual machines or deletion of existing ones.
- Resource usage **billed only as used.**

Options include:

- Microsoft Azure
- Google Cloud Platform (GCP)
- Amazon Web Services (AWS)

Useful Services (for this course)

Access Control

- *Authentication*: Verifying the identity of a person or system (e.g., username/password).
- *Authorization*: Determining which resources a user can access, and what capabilities are available e.g., they can read table X but cannot write to it.

Database

- A robust records database that you can configure/manage e.g., SQL/Relational DB.
- They often also have a file storage “database” for binary files e.g., images, images, video.

Notifications

- Mechanism to receive updates from the platform when data changes.
- This can replace the need to write your own service to manage notifications to client applications e.g., Supabase offers database notifications.

Types of authentication

HTTP

- **Basic** - use Base64 encoding for username/password. Requires HTTPS.
- Digest - applies a hash function to the username/password to encrypt them.
- Bearer - an authentication scheme that involves security tokens called bearer tokens. The Bearer authentication scheme is used as part of OAuth or JWT, but you can also provide custom logic for authorizing bearer tokens.

OAuth

- OAuth is an open standard for securing access to APIs. The `oauth` provider in Ktor supports authentication using external providers e.g., Google, Facebook.

Session

- Sessions provide a mechanism to persist data between different HTTP requests. Typical use cases include storing a logged-in user's ID.

Supabase

Supabase is an alternative to large-scale platforms like AWS or GCP. We recommend it for this course.

Features

- Authentication (basic, oauth)
- Postgres SQL Database
- Notifications API
- A “free” price tier for small projects!

It *also* works well with Kotlin. See Supabase Kotlin SDK documentation.

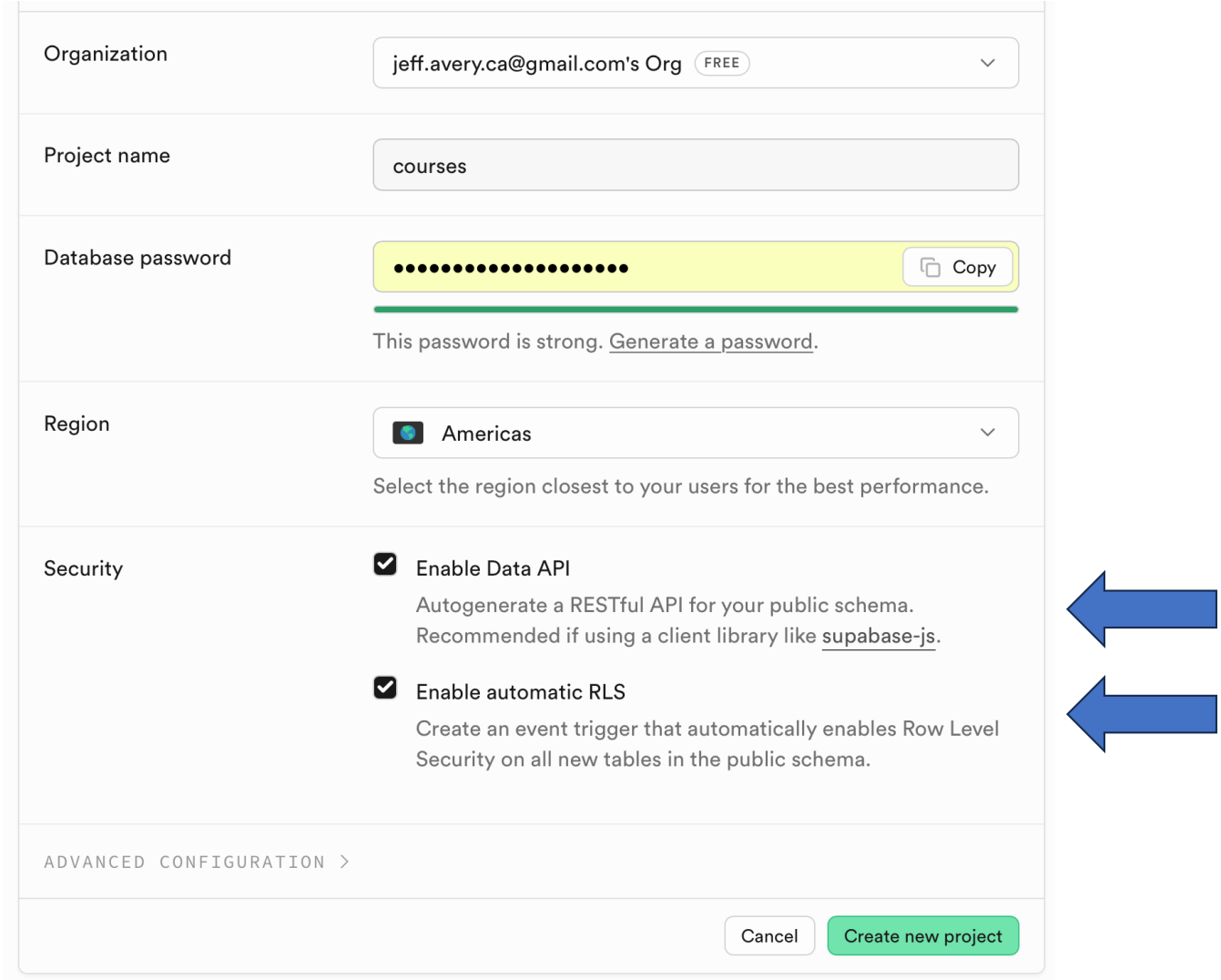
Setup

Steps

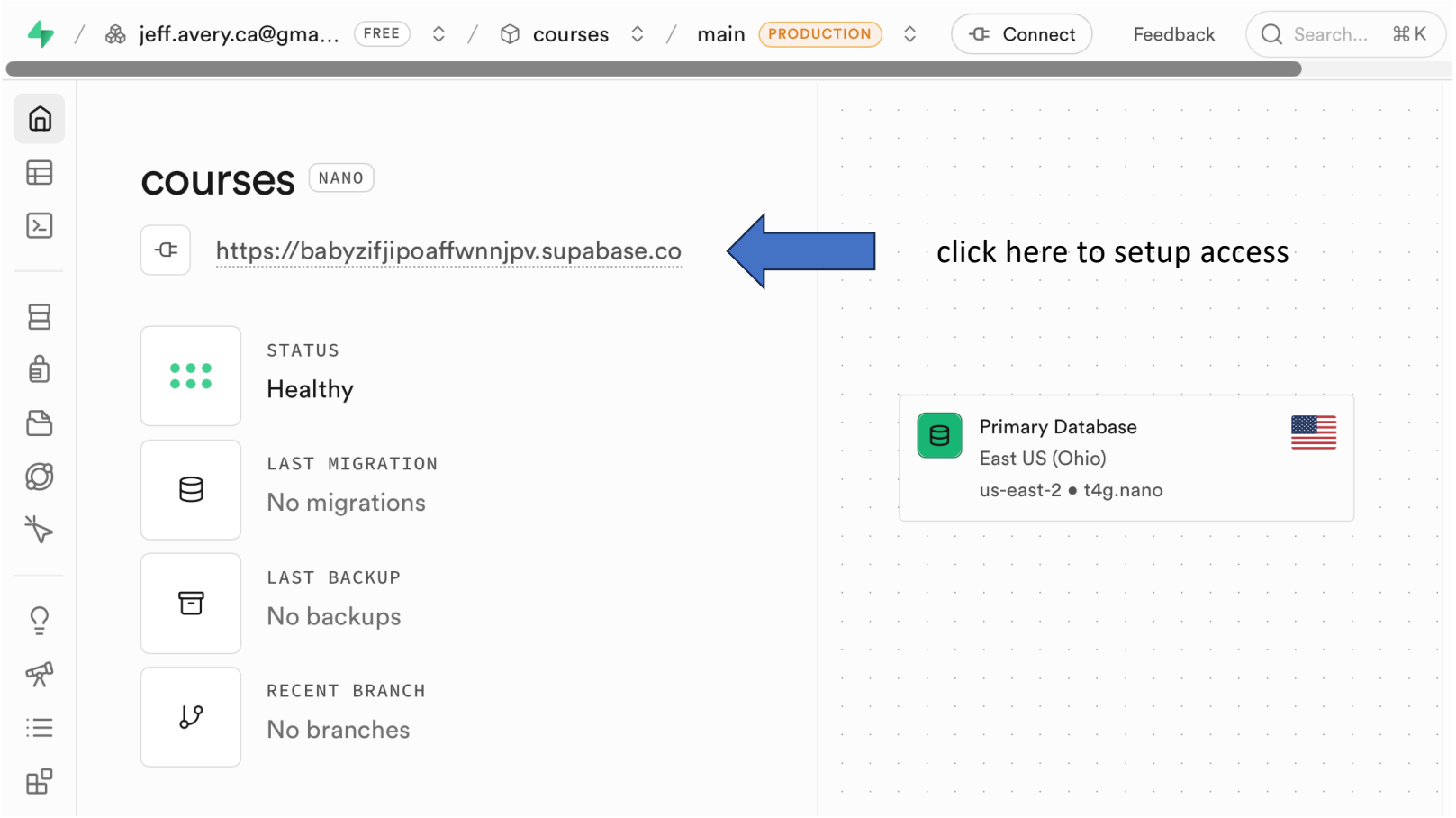
- Create a user account.
- Creating a project in Supabase.
- Adding your teammates to the project.

New Project

www.supabase.com
> Start your project

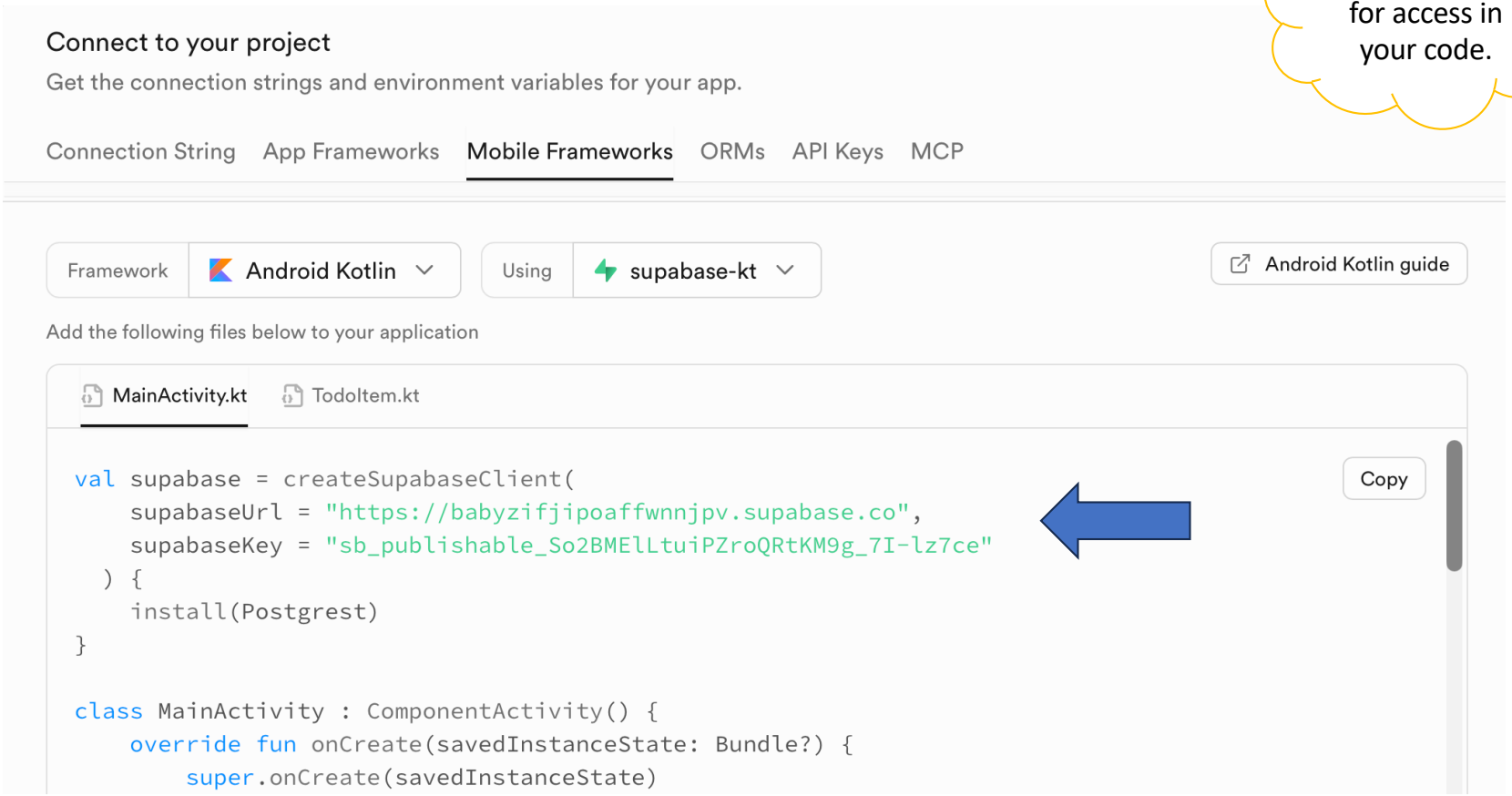


New Project

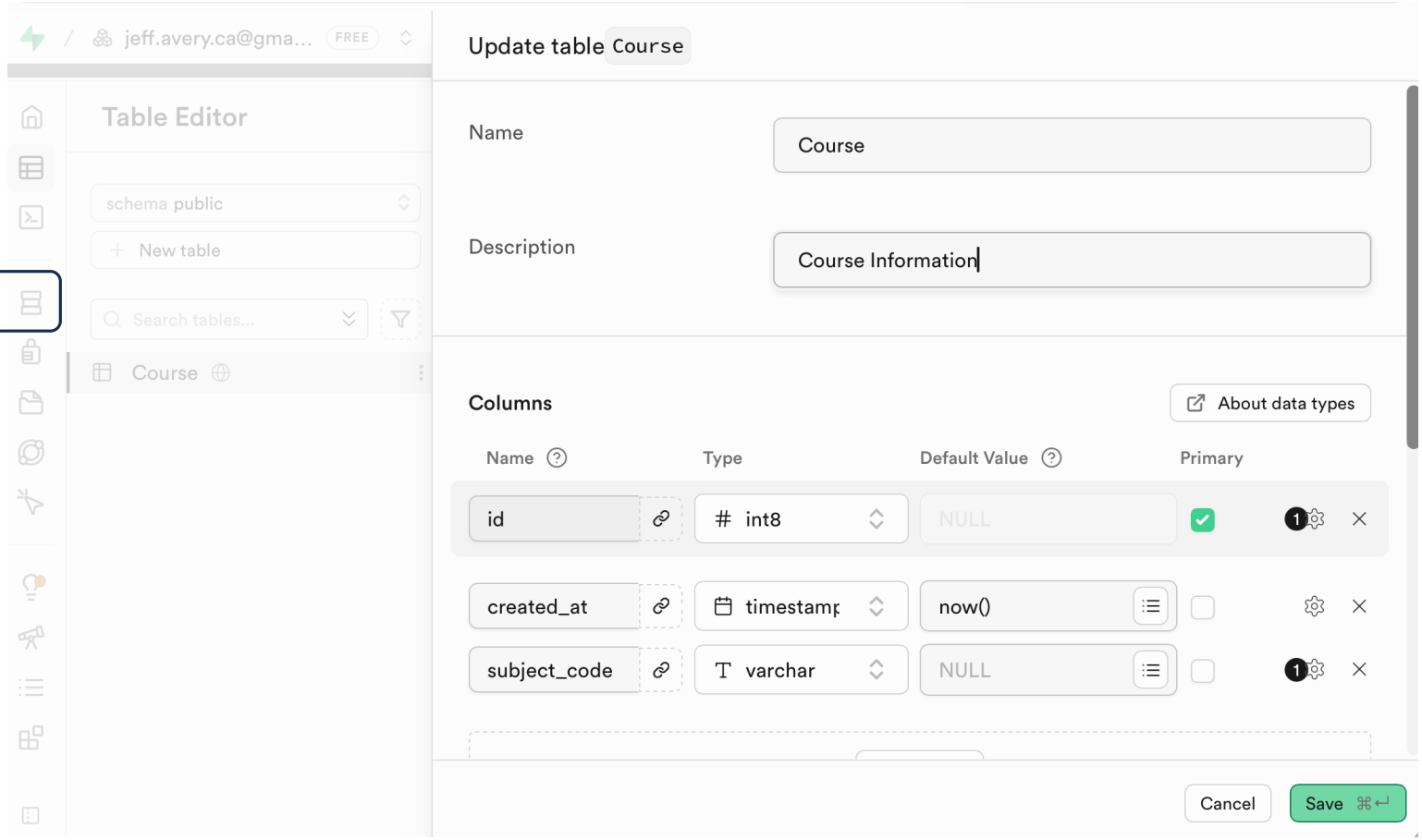

jeff.avery.ca@gma...
FREE
courses
main
PRODUCTION
Connect
Feedback
Search...
⌘K
courses
NANO
https://babyzifjipoaffwnnjpv.supabase.co
click here to setup access
STATUS
Healthy
LAST MIGRATION
No migrations
LAST BACKUP
No backups
RECENT BRANCH
No branches
Primary Database
East US (Ohio)
us-east-2 • t4g.nano

New Project

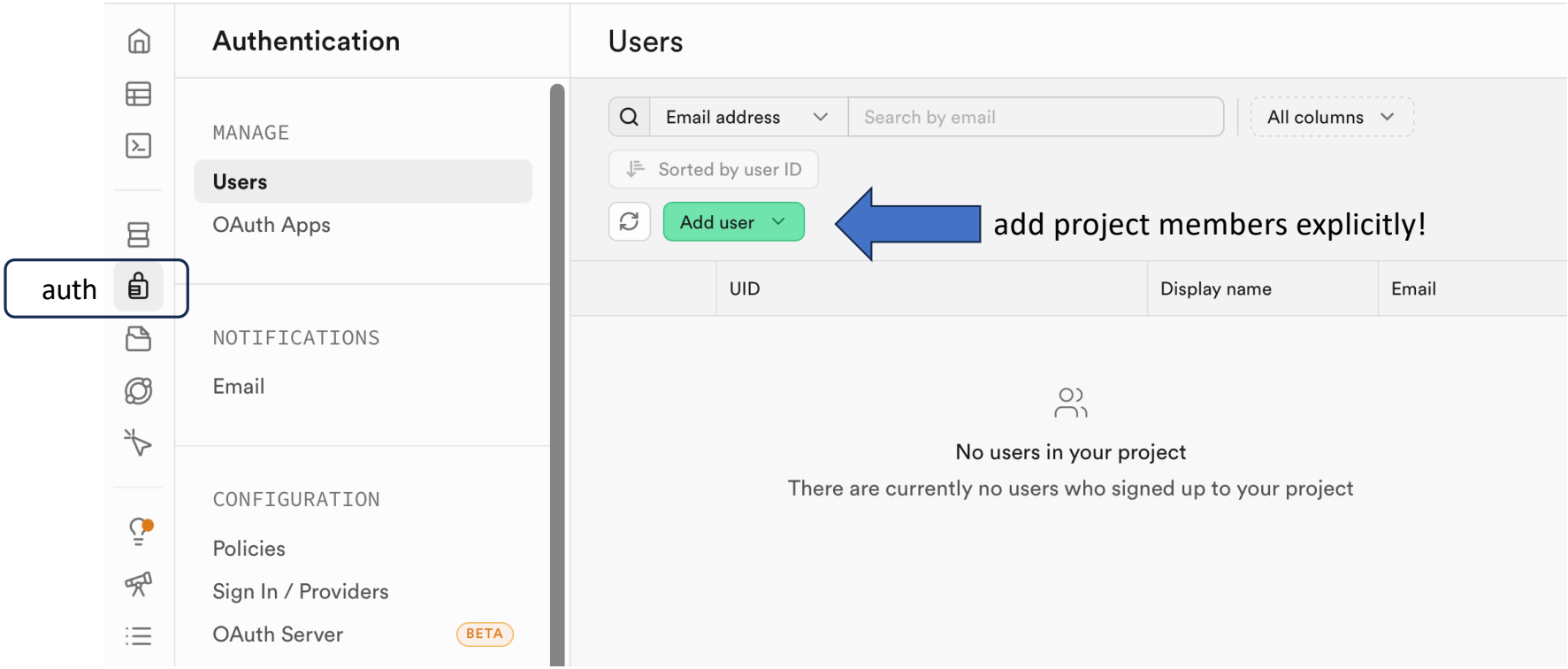
Use the supabaseKey for access in your code.



Creating Tables



Adding Users



Kotlin Queries

Taken from `mm-desktop` CloudStorage class.

Connect to a database

GitLab
demos/mm-desktop

```
class CloudStorage : IStorage {
  var supabase: SupabaseClient? = null
  var auth: Auth? = null

  init {
       supabase = createSupabaseClient(
              supabaseUrl = "https://bdifkfbvufwqdqfsdwee.supabase.co",
              supabaseKey = "sb_publishable_HaCdYtgF1YgeUWSqyIrTwg_EZ8VGdLb"
       ) {
              install(Postgrest)
              install(Auth)
       }
       auth = supabase?.auth
  }
}
```

This is the supabaseUrl and supabaseKey that was provided when you created the project.

Create and manage users

```
suspend fun createUser(email: String, password: String) {
      supabase?.auth?.signUpWith(Email) {
          this.email = email
          this.password = password
      }
  }

  suspend fun loginUser(email: String, password: String) {
      supabase?.auth?.signInWith(Email) {
          this.email = email
          this.password = password
      }
  }

  suspend fun logoutUser() {
      supabase?.auth?.signOut()
  }
```

Run a query

```
override suspend fun read(id: Int): Task? {
      return supabase?.from("tasks")
          ?.select(columns = Columns.list("id", ”priority", "title")) {
              filter {
                  Task::id eq id
              }
          }?.decodeSingle<Task>()
  }
```

This requires a Task data class, with a structure that matches the Tasks table in the database.

Next steps?

Using Supabase with Kotlin Android

https://supabase.com/docs/guides/getting-started/quickstarts/kotlin

- Official Kotlin/Android guide

Kotlin Client Library
https://supabase.com/docs/reference/kotlin/introduction

Code demos!

- Community supported
- Android + desktop, and includes library installation instructions