

# Supabase

---

CS 346 Application  
Development

# Local vs. Cloud Hosting

Rosenberg & Mateos (2014) define cloud computing as

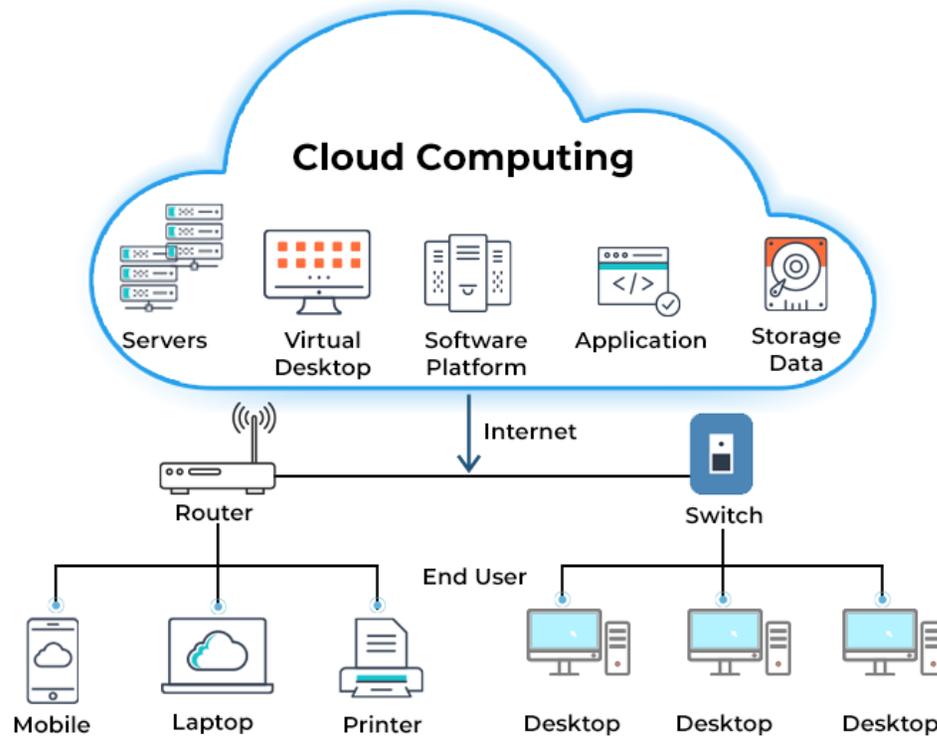
"a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services)"

-- [https://en.wikipedia.org/wiki/Cloud\\_computing](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.

## CLOUD COMPUTING ARCHITECTURE



[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

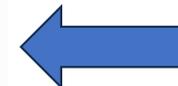
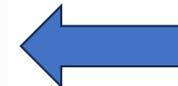
# Steps

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

# New Project

[www.supabase.com](http://www.supabase.com)  
> Start your project

Organization	jeff.avery.ca@gmail.com's Org <small>FREE</small>
Project name	courses
Database password	..... <small>Copy</small>
	This password is strong. <a href="#">Generate a password.</a>
Region	Americas
	Select the region closest to your users for the best performance.
Security	<input checked="" type="checkbox"/> <b>Enable Data API</b> Autogenerate a RESTful API for your public schema. Recommended if using a client library like <a href="#">supabase-js</a> .
	<input checked="" type="checkbox"/> <b>Enable automatic RLS</b> Create an event trigger that automatically enables Row Level Security on all new tables in the public schema.
<small>ADVANCED CONFIGURATION &gt;</small>	
	<small>Cancel</small> <b>Create new project</b>



# New Project

The screenshot shows the Supabase dashboard for a project named "courses". The top navigation bar includes the user "jeff.avery.ca@gma...", a "FREE" tier indicator, the project name "courses", the current branch "main", and a "PRODUCTION" environment tag. There are buttons for "Connect", "Feedback", and a search bar.

The main content area is divided into two columns. The left column contains project details:

- courses** (NANO)
- URL: <https://babyzifjipoaffwvnnjpv.supabase.co>
- STATUS**: Healthy
- LAST MIGRATION**: No migrations
- LAST BACKUP**: No backups
- RECENT BRANCH**: No branches

The right column features a "click here to setup access" link with a blue arrow pointing to the URL. Below this is a "Primary Database" card showing "East US (Ohio)" with an American flag icon and "us-east-2 • t4g.nano".

# New Project

Use the supabaseKey for access in your code.

Connect to your project  
Get the connection strings and environment variables for your app.

Connection String App Frameworks **Mobile Frameworks** ORMs API Keys MCP

Framework  Android Kotlin ▼ Using  supabase-kt ▼ [Android Kotlin guide](#)

Add the following files below to your application

 MainActivity.kt  TodoItem.kt

```
val supabase = createSupabaseClient(
    supabaseUrl = "https://babyzifjipoaffwnnjpv.supabase.co",
    supabaseKey = "sb_publishable_So2BME1LtuiPZroQRtKM9g_7I-lz7ce"
) {
    install(Postgrest)
}

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
    }
}
```

[Copy](#)



# Creating Tables

The screenshot shows a database management interface for updating a table named 'Course'. The interface is divided into several sections:

- Table Editor (Sidebar):** Located on the left, it shows the current schema as 'schema public' and a list of tables including 'Course'. A blue box labeled 'schema' highlights the 'schema public' dropdown.
- Update table Course:** The main header of the interface.
- Name:** A text input field containing 'Course'.
- Description:** A text input field containing 'Course Information'.
- Columns:** A table defining the table's structure. It includes a 'Columns' header, a 'About data types' link, and a table with columns: Name, Type, Default Value, and Primary.

Name	Type	Default Value	Primary
id	# int8	NULL	<input checked="" type="checkbox"/>
created_at	timestamp	now()	<input type="checkbox"/>
subject_code	T varchar	NULL	<input type="checkbox"/>

At the bottom right, there are 'Cancel' and 'Save' buttons.

# Adding Users

The screenshot shows a web interface for managing authentication. On the left is a sidebar with a menu. The 'auth' menu item is highlighted with a blue box. The main content area is titled 'Users' and contains a search bar, a sorting dropdown, and an 'Add user' button. A blue arrow points to the 'Add user' button with the text 'add project members explicitly!'. Below the button is a table header with columns 'UID', 'Display name', and 'Email'. The table body is empty, displaying a message: 'No users in your project' and 'There are currently no users who signed up to your project'.

UID	Display name	Email
No users in your project There are currently no users who signed up to your project		

# 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?.signUpWithEmail {
        this.email = email
        this.password = password
    }
}

suspend fun loginUser(email: String, password: String) {
    supabase?.auth?.signInWithEmail {
        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>

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



Code  
demos!