



THE UNIVERSITY OF
CHICAGO



MPCS 51033 • AUTUMN 2017 • SESSION 7

BACKENDS FOR MOBILE APPLICATIONS

CLASS NEWS

CLASS NEWS

- Office hours Thursday from 10-11:30 in Young 308

CLASS NEWS

- Week 7 - CloudKit
 - Assignment 5 assigned
- Week 8 - CloudKit Server Extensions; Swift on the Server
 - Case Study assigned
 - Finish server component of Assignment 5
- Week 9 - Case Studies in Class (Assignment 5 Due)
- Week 10 - Cloud Roundup (Realm, Serverless, etc.)

CLASS NEWS

- Case Studies
 - Find an area of interest in mobile/cloud computing and present to class
 - Ideas:
 - New service (serverless, azure, aws lambda)
 - Technique (sharding :), mysql with app engine)
 - Case study (Snapchat on App Engine, what does XX company use)
 - Deeper dive on topic covered (eg. Google Vision API, etc.)
 - 30 minutes

CLASS NEWS

- Week 7 - CloudKit
 - Assignment 5 assigned
- Week 8 - CloudKit Server Extensions; Swift on the Server
 - Case Study assigned
 - Finish server component of Assignment 5
- Week 9 - Case Studies in Class (Assignment 5 Due)
- Week 10 - Cloud Roundup (Realm, Serverless, etc.)

MAKE-UP CLASS
MATERIALS AFTER
PRESENTATIONS

CLASS NEWS

- Final Projects
 - Opportunity for you to apply what you learned in class to a project you care about
 - Can use old iOS project or extend assignment
 - Talk me about your ideas early

CLOUDKIT ☁

CLOUDKIT

- CloudKit
QuickStart

Table of Contents

- Introduction
- Enabling CloudKit in Your App
 - About Containers and Databases
 - Setup
 - Enable iCloud and Select CloudKit
 - Access CloudKit Dashboard
- Share Containers Between Apps
 - Add Containers to an App
 - Create Custom Containers
 - Verify Your Steps
- Create an iCloud Account for Development
- Recap
- Creating a Database Schema by Saving Records
 - About Designing Your Schema
 - Separate Data into Record Types
 - Decide on Names for Your Records
 - Create Records Programmatically
 - Save Records
 - Enter iCloud Credentials Before Running Your App
 - Alert the User to Enter iCloud Credentials
 - Run Your App
- Verify Your Steps
 - Recap
- Using CloudKit Dashboard to Manage Databases
 - About the Development and Production Environments
 - Select Your Container
 - Reset the Development Environment
 - Create and Delete Record Types
 - Add, Modify, and Delete Records
 - Search Records
 - Sort Records
 - Recap
- Fetching Records
 - Fetch Records by Identifier
 - Fetch and Modify Records
 - Query for Records Using Predicates
 - Recap
- Using Asset and Location Fields
- Adding Reference Fields

About This Document

This document gets you started creating a CloudKit app that stores structured app and user data in iCloud. Using CloudKit, instances of your app—launched by different users on different devices—have access to the records stored in the app's database. Use CloudKit if you have model objects that you want to persist and share between multiple apps running on multiple devices. These model objects are stored as records in the database and can be provided by you or authored by the user.



You'll learn how to:

- Enable CloudKit in your Xcode project and create a schema programmatically or with CloudKit Dashboard
- Fetch records and subscribe to changes in your code
- Use field types that are optimized for large data files and location data
- Subscribe to record changes to improve performance
- Test your CloudKit app on multiple devices before uploading it to the App Store, Mac App Store, or Apple TV App Store.
- Deploy the schema to production and keep it current with each release of your app

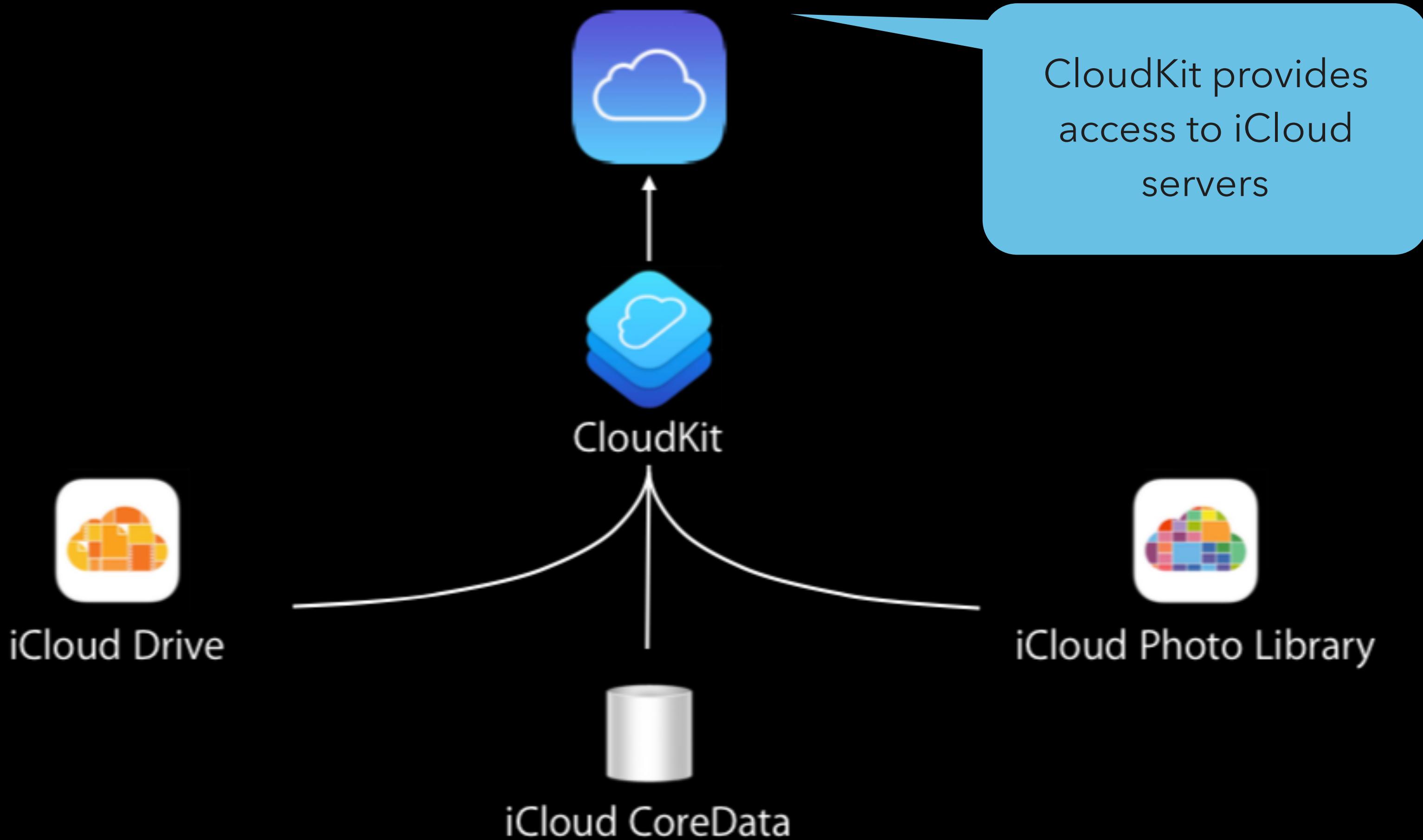
See [Glossary](#) for the definition of database terms used in this book.

See Also

The following WWDC sessions provide more CloudKit architecture and API details:

- [WWDC 2014: Introducing CloudKit](#) introduces the basic architecture and APIs used to save and fetch records.
- [WWDC 2014: Advanced CloudKit](#) covers topics such as private data, custom record zones, ensuring data integrity, and effectively modeling your data.
- [WWDC 2015: CloudKit Tips and Tricks](#) explore some of its lesser-known features and best practices for subscriptions and queries.
- [WWDC 2016: What's New with CloudKit](#) covers the new sharing APIs that lets you share private data between iCloud users.
- [WWDC 2016: CloudKit Best Practices](#) best practices from the CloudKit engineering team about how to take advantage of the APIs and push notifications in order to provide your users with the best experience.

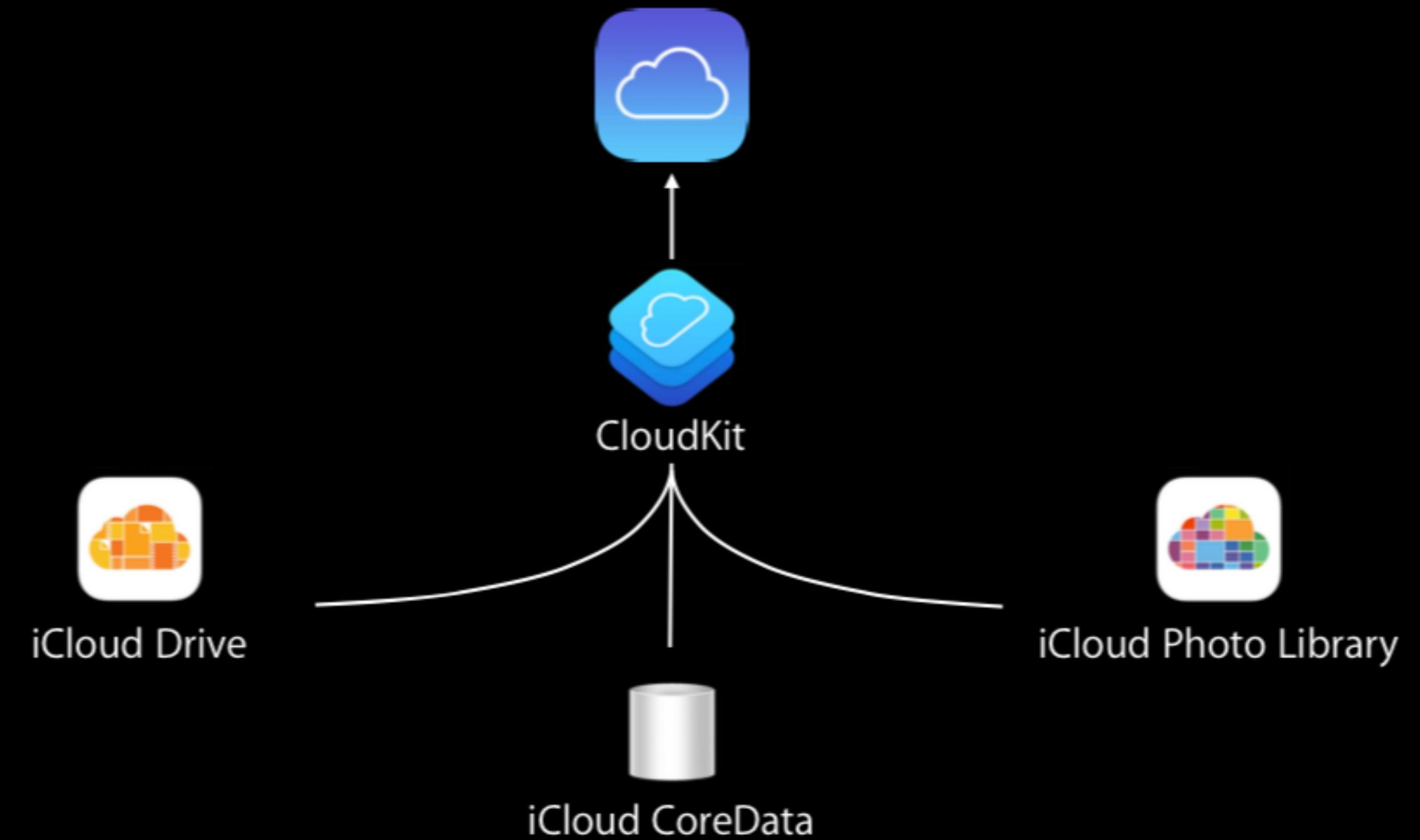
CLOUDKIT



CLOUDKIT

- Access to iCloud servers
- Supported on macOS, tvOS, iOS, watchOS and web (JS API)

#1 consideration
on choosing over
other services



CLOUDKIT

- Basically Free
- Public scales with users to PB
- Private db is charged against users quota
 - Permission can make anything private

Getting Started with CloudKit for free.

CloudKit provides a generous amount of free public storage and data transfer to help you get started. Sign in to the [CloudKit Dashboard](#) to view your quota and project usage.

10 GB Asset storage	100 MB Database storage	2 GB Data transfer	40 Requests per second
-------------------------------	-----------------------------------	------------------------------	----------------------------------

CLOUDKIT



Capacity scales with your users.

The amount of public storage and data transfer allocated to your apps will grow with every new active user—all for free with very high limits. Calculate the amount of storage you'll gain as your number of active users grows.

10,000,000

Active Users

- Nice problem to have

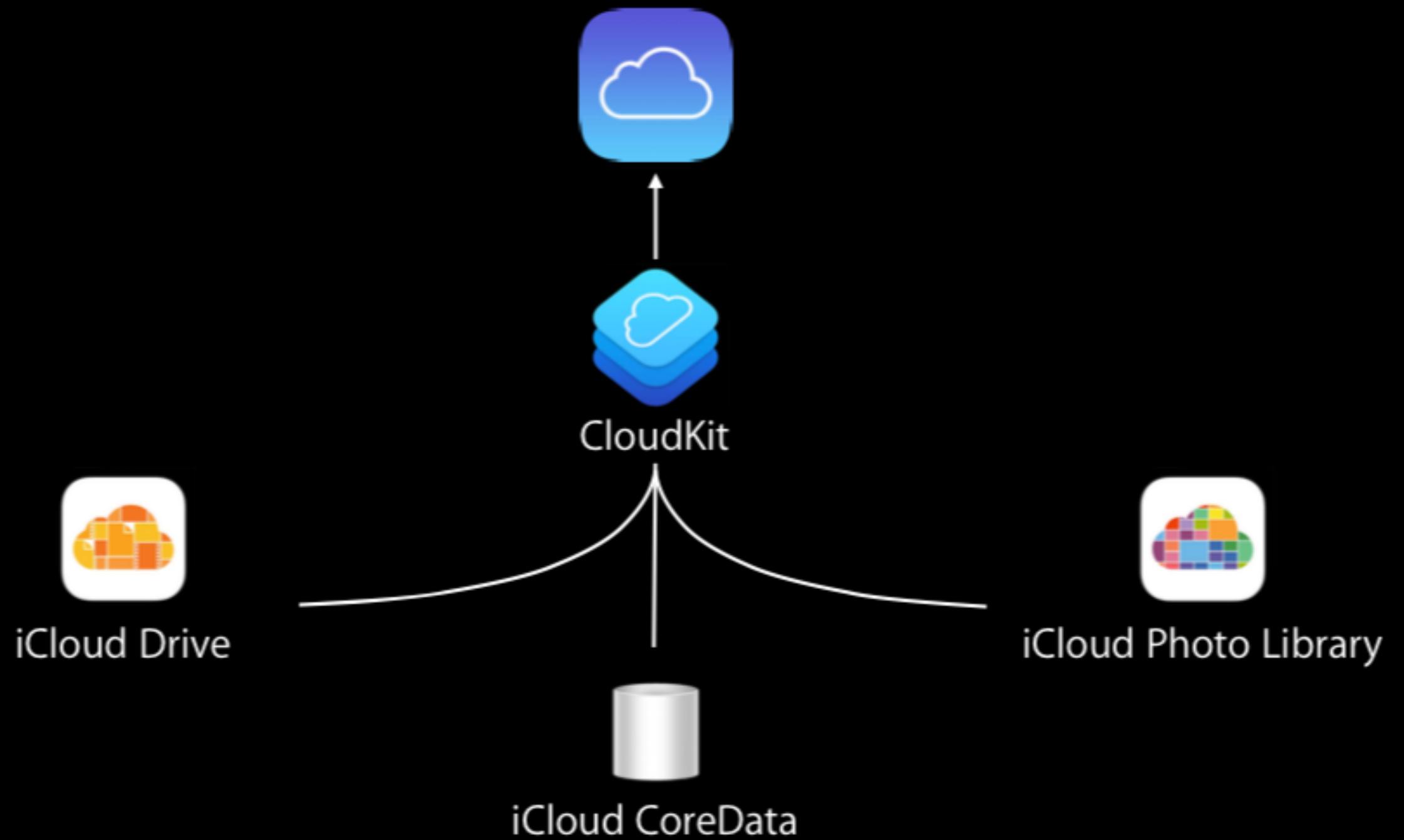
1 PB Asset storage	10 TB Database storage	200 TB Data transfer	400 Requests per sec.
Based on: 100 MB per user	Based on: 1 MB per user	Based on: 20 MB per user	Based on: 4 per 100,000 users

\$0

Total Cost

CLOUDKIT

- Uses iCloud accounts
 - Logged in accounts used to identify user
 - Not logged in users can have read only anonymous access



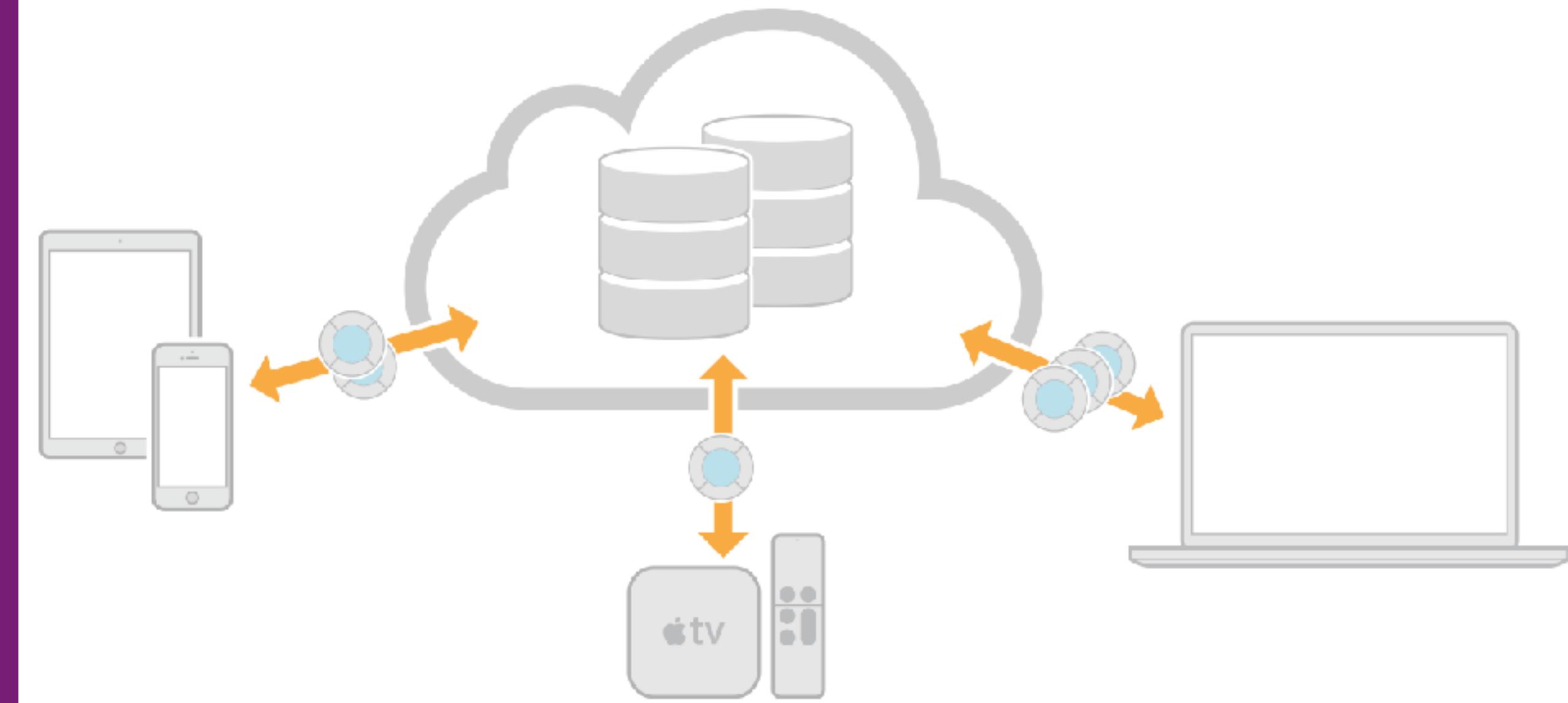
CLOUDKIT

- Databases
 - Public
 - Private
 - Shared



CLOUDKIT

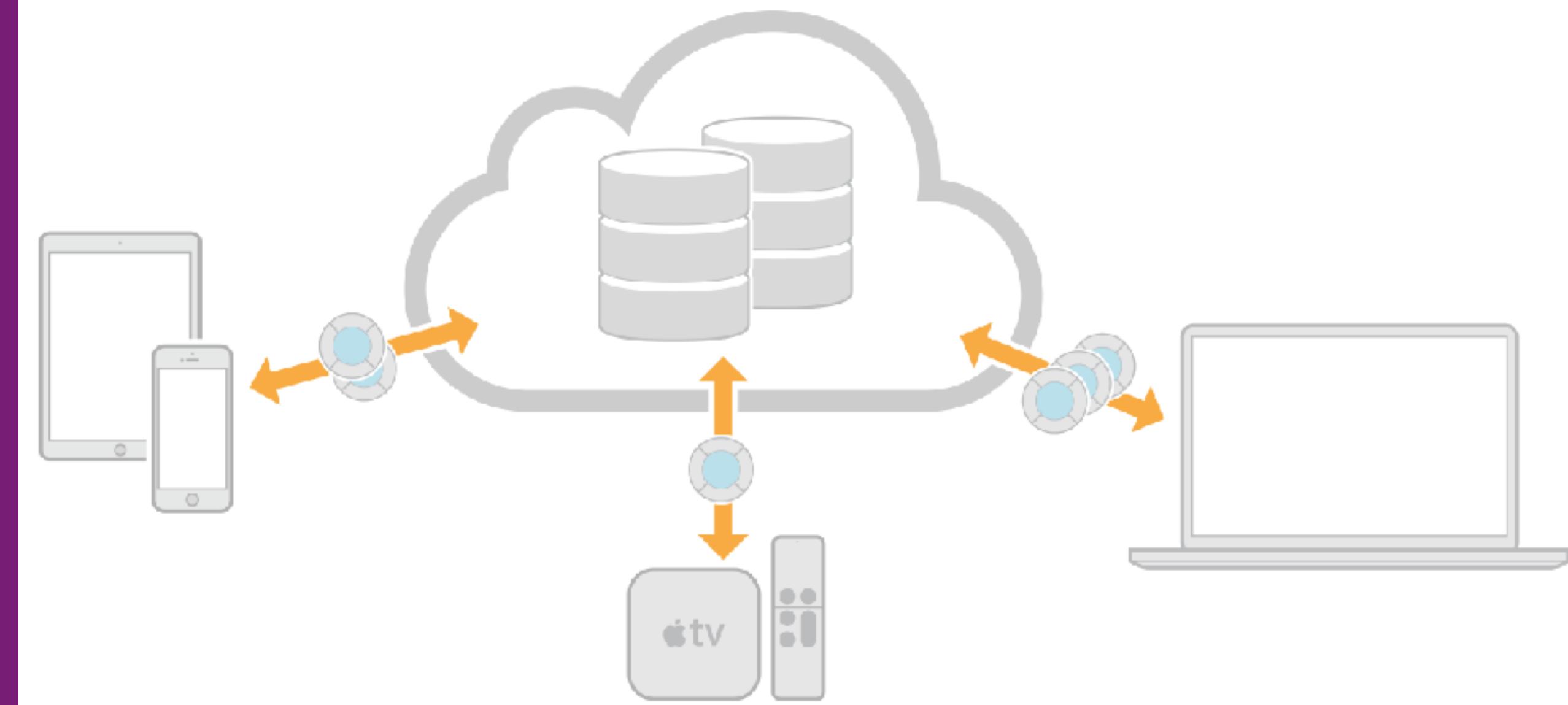
- Structured and bulk data
 - Data with types
 - Blobs



CLOUDKIT

CORE OBJECT

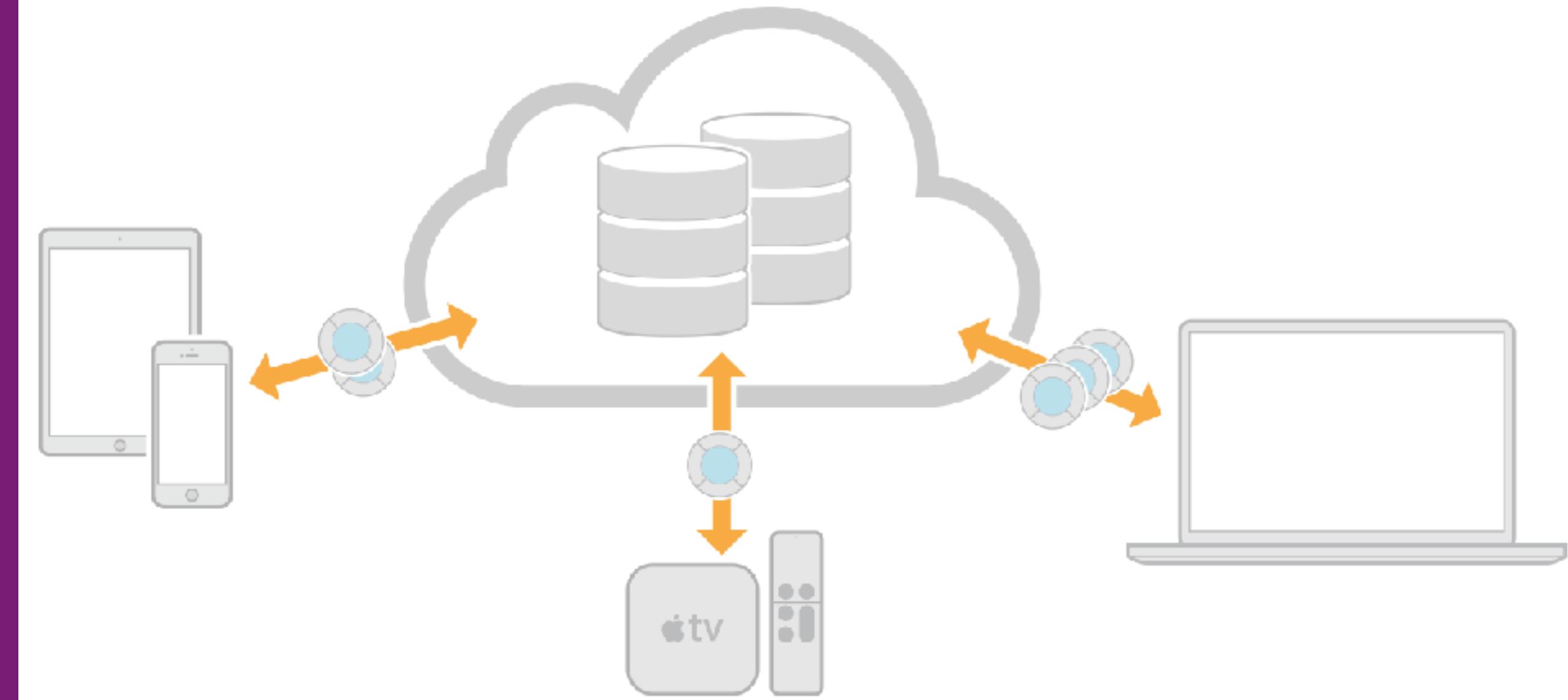
- Containers
- Databases
- Records
- Record Zones
- Record Identifiers
- Shares
- References
- Assets



CLOUDKIT

- Transport layer, not local persistence
- You still need to figure out what to do with the data once you have it on your device
 - Cache
 - Mirror

#2 consideration
on choosing over
other services



CLOUDKIT

- CloudKit's place in mobile backends -
The good
 - Convenient
 - Free
 - Zero authentication
 - Push notifications
 - Proven (Apple actually uses it)



CLOUDKIT

- CloudKit's place in mobile backends -
The bad
 - No local storage
 - Platform lock
(account & device)
 - No native server logic



CLOUDKIT

- CloudKit's place in mobile backends - The bad
 - No local storage
 - Platform lock (account & device)
 - No native server logic

We will discuss how these may be less of an issue than they seem



ENABLING CLOUDKIT IN YOUR APPLICATION

ENABLING CLOUDKIT IN YOUR APPLICATION

The screenshot shows the Xcode interface with the project "CloudyWithAChanceOfErrors" selected. The General tab is active in the settings pane. A yellow callout bubble with the text "NEED PAID DEVELOPER ACCOUNT" points to the "Automatically manage signing" checkbox under the Signing section.

CloudyWithAChanceOfErrors

CloudyWithAChanceOfErrors

CloudyWithAChanceOfErrors.entitlements

AppDelegate.swift

ViewController.swift

CloudKitManager.swift

UserRecordViewController.swift

README.md

Main.storyboard

Assets.xcassets

LaunchScreen.storyboard

Info.plist

CloudPlayground.playground

Products

Frameworks

Cloudy...fErros > iPhone 7

Finished running CloudyWithAChanceOfErros on iPhone 7 1 ! 1

General Capabilities Resource Tags Info Build Settings Build Phases Build Rules

Identity

Display Name: CloudyWithAChanceOfErros

Bundle Identifier: cloud.uchicago.CloudyWithAChanceOfErros

Version: 1.0

Build: 1

Signed for Distribution

Automatically manage signing

Xcode will create and update profiles, app IDs, and certificates.

Team: University of Chicago (Department of Comput...)

Provisioning Profile: Xcode Managed Profile ⓘ

Signing Certificate: iPhone Developer: Andrew Binkowski (8T47C82J...)

Deployment Info

ENABLING CLOUDKIT IN YOUR APPLICATION

The screenshot shows the Xcode project navigator on the left and the Capabilities tab of the CloudyWithAChanceOfErrors target in the center. A yellow callout box labeled "NEW CAPABILITY" points to the "CloudKit" section. Another yellow callout box labeled "CUSTOM CONTAINERS CAN BE SHARED BETWEEN APPS" points to the "Containers" section where a custom container is selected.

NEW CAPABILITY

CUSTOM CONTAINERS CAN BE SHARED BETWEEN APPS

CloudyWithAChanceOfErrors

CloudKit

Services:

- Key-value storage
- iCloud Documents
- CloudKit

Containers:

- Use default container
- Specify custom containers

iCloud.cloud.uchicago.CloudyWithAChanceOfErrors iCloud.\$

CloudKit Dashboard

Steps:

- ✓ Add the iCloud feature to your App ID.
- ✓ Add iCloud Containers to your App ID
- ✓ Add the iCloud entitlement to your entitlements file
- ✓ Link CloudKit.framework

ENABLING CLOUDKIT IN YOUR APPLICATION

The screenshot shows the CloudKit developer portal interface. At the top, there's a navigation bar with a cloud icon, a search bar labeled "Search for Container", and a user dropdown for "ANDREW BINKOWSKI". Below the header, the user's name "Andrew Binkowski" is displayed, along with their team information: "Individual Team - 9U8C5A3E8R". A "VIEW TEAM" button is also present. The main content area shows a message: "No containers. You do not have access to any CloudKit containers for this team. Learn how to automatically generate one with Xcode or create one in the Developer Portal." A yellow callout bubble points from this message towards the first container card. The bottom section lists several CloudKit containers, each represented by a blue card with a grid icon and a unique identifier and status: "iCloud.cloud.uchicago.CloudyWithAChanceOfErros IN DEVELOPMENT", "iCloud.com.alicechicago.InstaWatch IN DEVELOPMENT", "iCloud.com.bennetth.loopyQ2 IN DEVELOPMENT", "iCloud.com.isabellee.CatGameDevelopers-grade IN DEVELOPMENT", "iCloud.com.isabellee.CatJumpGame-grade IN DEVELOPMENT", and "iCloud.com.isabellee.spotSpotSpots-graded IN DEVELOPMENT".

DEFAULT CONTAINER IS CREATED ON ICLOUD

Andrew Binkowski
Individual Team - 9U8C5A3E8R
VIEW TEAM

No containers.
You do not have access to any CloudKit containers for this team.
Learn how to automatically generate one with Xcode or create one in the [Developer Portal](#).

University of Chicago (Department of Computer Science)
Educational Institution Team -

iCloud.cloud.uchicago.CloudyWithAChanceOfErros
IN DEVELOPMENT

iCloud.com.alicechicago.InstaWatch
IN DEVELOPMENT

iCloud.com.bennetth.loopyQ2
IN DEVELOPMENT

iCloud.com.isabellee.CatGameDevelopers-grade
IN DEVELOPMENT

iCloud.com.isabellee.CatJumpGame-grade
IN DEVELOPMENT

iCloud.com.isabellee.spotSpotSpots-graded
IN DEVELOPMENT

ENABLING CLOUDKIT IN YOUR APPLICATION

- Dashboard is only way to access permissions
- Data can be created, manipulated
- View analytic information about your data, users and operations

Apple Inc.

iCloud.cloud.uchicago.CloudyWithAChanceOfErros

Choose an area in an environment to view:

Development

 **Data >**
Manage records, record types, indexes, subscriptions, and security roles in your public, private, and shared databases.

 **Logs >**
View real-time and historical logs of server activity, showing database operations, push notifications, and other activity in this environment.

 **Telemetry >**
View graphs of server-side performance and utilization across database, sharing, and push events in this environment.

 **Public Database Usage >**
View graphs of public database usage including active users, requests per second, asset transfer, and database storage.

 **API Access >**
Manage API tokens and server-to-server keys that allow web service calls for this environment.

Production

 **Data >**
Manage records, record types, indexes, subscriptions, and security roles in your public, private, and shared databases.

 **Logs >**
View real-time and historical logs of server activity, showing database operations, push notifications, and other activity in this environment.

 **Telemetry >**
View graphs of server-side performance and utilization across database, sharing, and push events in this environment.

 **Public Database Usage >**
View graphs of public database usage including active users, requests per second, asset transfer, and database storage.

 **API Access >**
Manage API tokens and server-to-server keys that allow web service calls for this environment.

Reset... Environment Settings... Deploy to Production...

Display a menu

ENABLING CLOUDKIT IN YOUR APPLICATION

- "Flip a switch" to go to production mode

Apple Inc.

iCloud.cloud.uchicago.CloudyWithAChanceOfErros

iCloud.cloud.uchicago.CloudyWithAChanceOfErros

Choose an area in an environment to view:

Development

- Data >** Manage records, record types, indexes, subscriptions, and security roles in your public, private, and shared databases.
- Logs >** View real-time and historical logs of server activity, showing database operations, push notifications, and other activity in this environment.
- Telemetry >** View graphs of server-side performance and utilization across database, sharing, and push events in this environment.
- Public Database Usage >** View graphs of public database usage including active users, requests per second, asset transfer, and database storage.
- API Access >** Manage API tokens and server-to-server keys that allow web service calls for this environment.

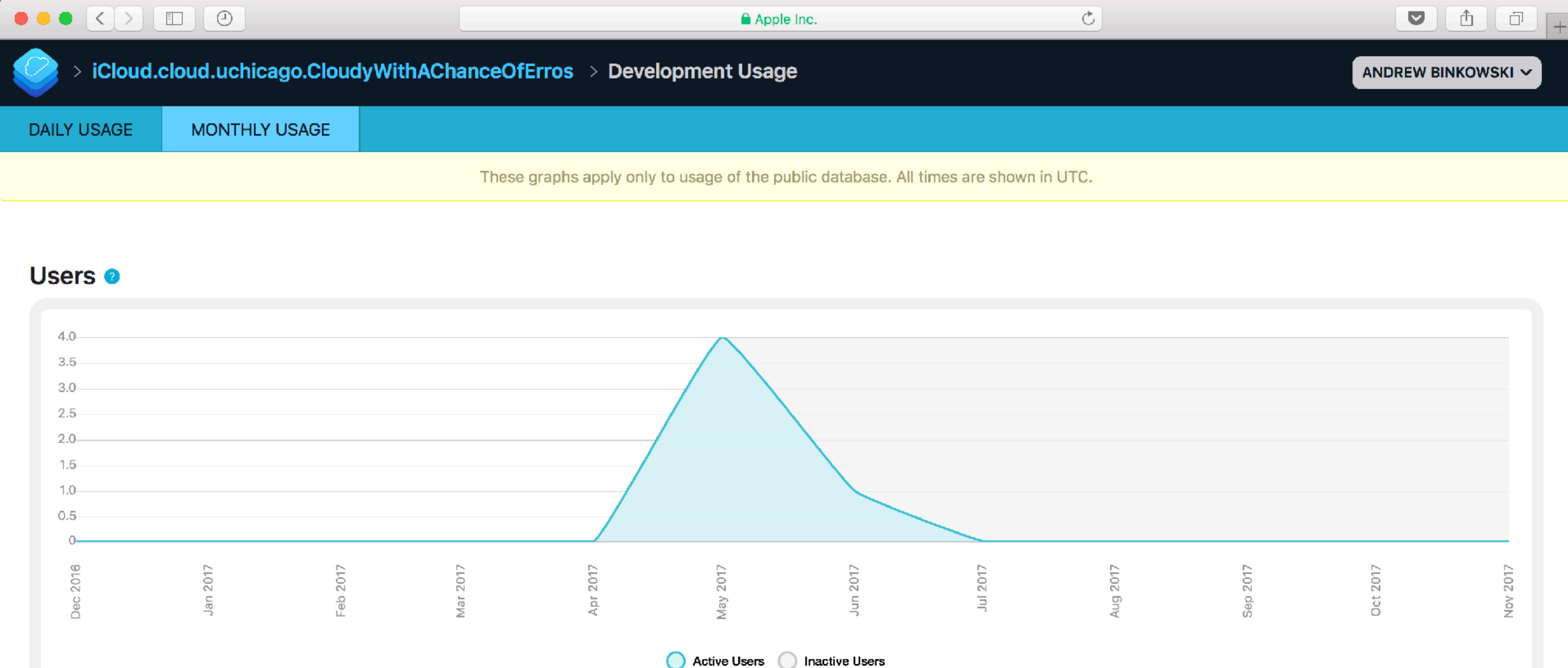
Reset... **Environment Settings...** **Deploy to Production...**

Production

- Data >** Manage records, record types, indexes, subscriptions, and security roles in your public, private, and shared databases.
- Logs >** View real-time and historical logs of server activity, showing database operations, push notifications, and other activity in this environment.
- Telemetry >** View graphs of server-side performance and utilization across database, sharing, and push events in this environment.
- Public Database Usage >** View graphs of public database usage including active users, requests per second, asset transfer, and database storage.
- API Access >** Manage API tokens and server-to-server keys that allow web service calls for this environment.

Display a menu

ENABLING CLOUDKIT IN YOUR APPLICATION



ENABLING CLOUDKIT IN YOUR APPLICATION

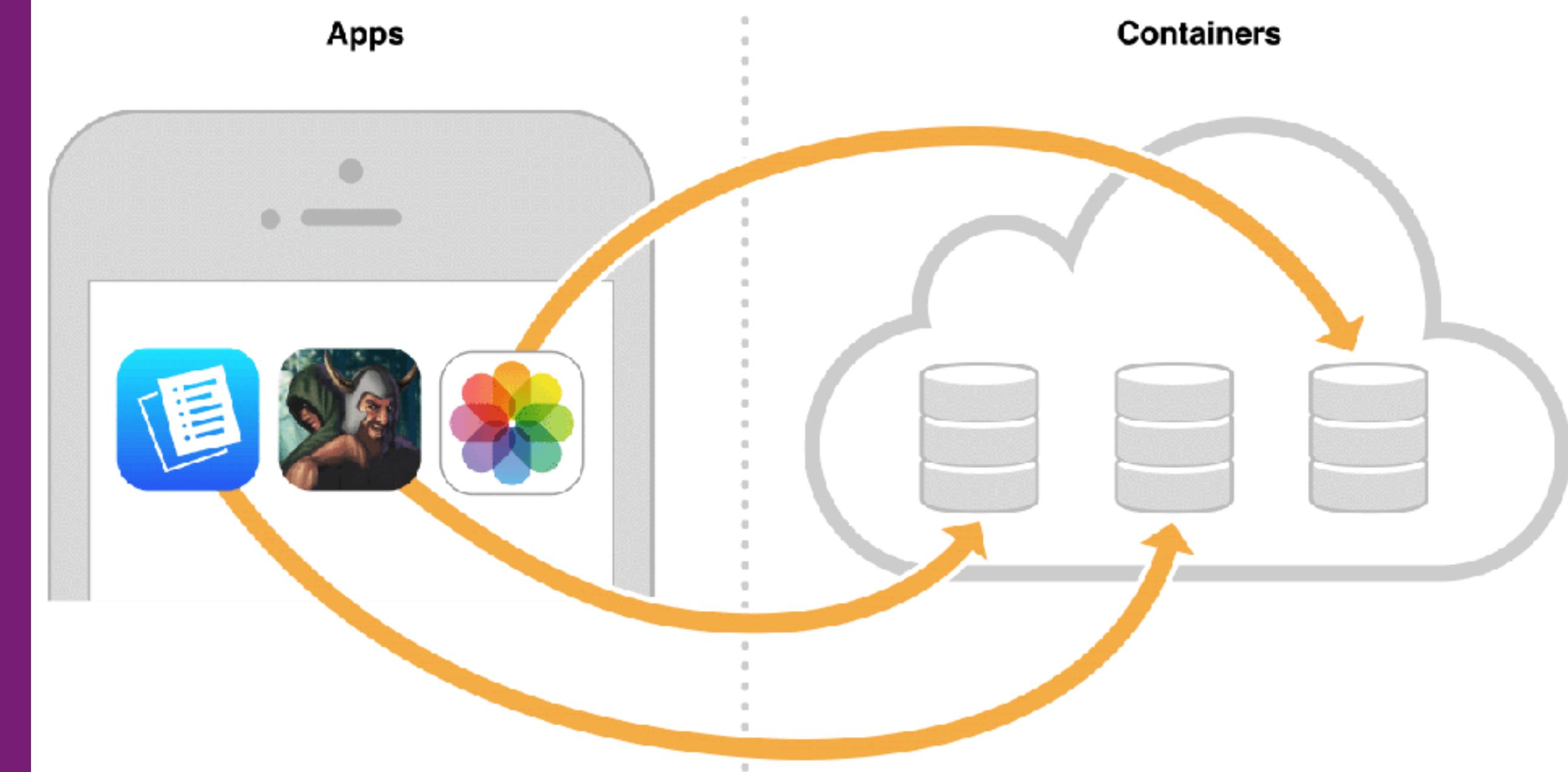
A screenshot of the CloudKit dashboard interface. At the top, there's a navigation bar with icons for window control, a lock icon indicating security, and user authentication. Below the navigation bar, the URL is shown as `iCloud.cloud.uchicago.CloudyWithAChanceOfErros`. To the right of the URL is a dropdown menu for the user "ANDREW BINKOWSKI". The main content area has tabs for "DATABASE" and "PUSH NOTIFICATIONS", with "PUSH NOTIFICATIONS" currently selected. Below the tabs are three dropdown filters: "Last 30 Days", "All Private Databases", and "All Operations". The main visual element is a line chart titled "Requests" with a question mark icon. The Y-axis ranges from 0.5 to 3.0 with increments of 0.5. The X-axis represents time, though no specific labels are provided. The chart shows a sharp, nearly vertical increase in requests starting from the bottom of the visible range up to 3.0, indicating a recent spike in activity.

CONTAINER

CLOUDKIT OBJECTS

CONTAINER

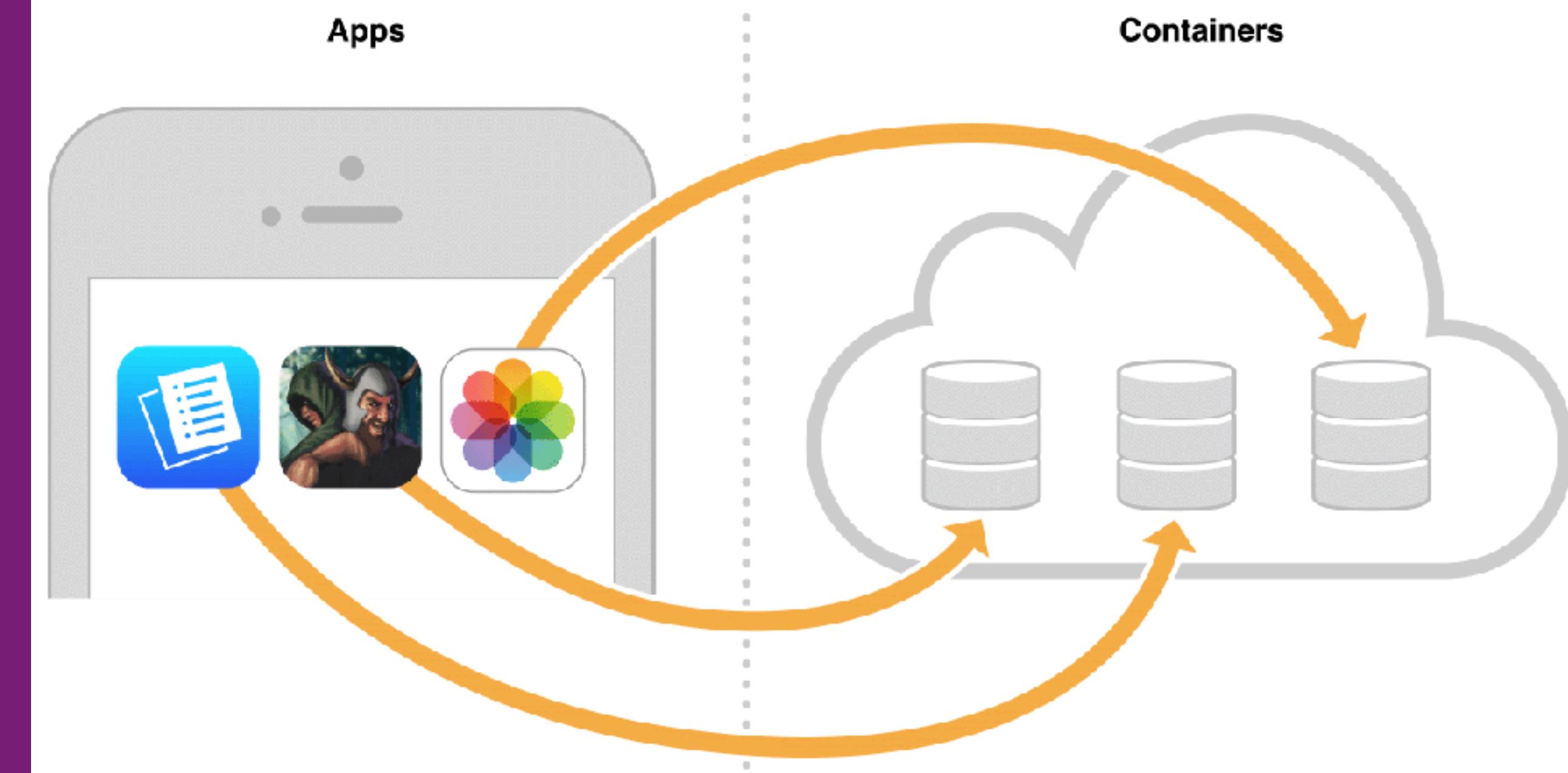
- CKContainer
 - One container per application
 - Data segregation
 - User encapsulation
 - Managed by the developer via portal



CLOUDKIT OBJECTS

CONTAINER

- CKContainer can be shared between apps from the same developer
- You can create additional custom containers for your app



CLOUDKIT CONTAINER

```
import CloudKit
```

IT'S JUST THAT EASY

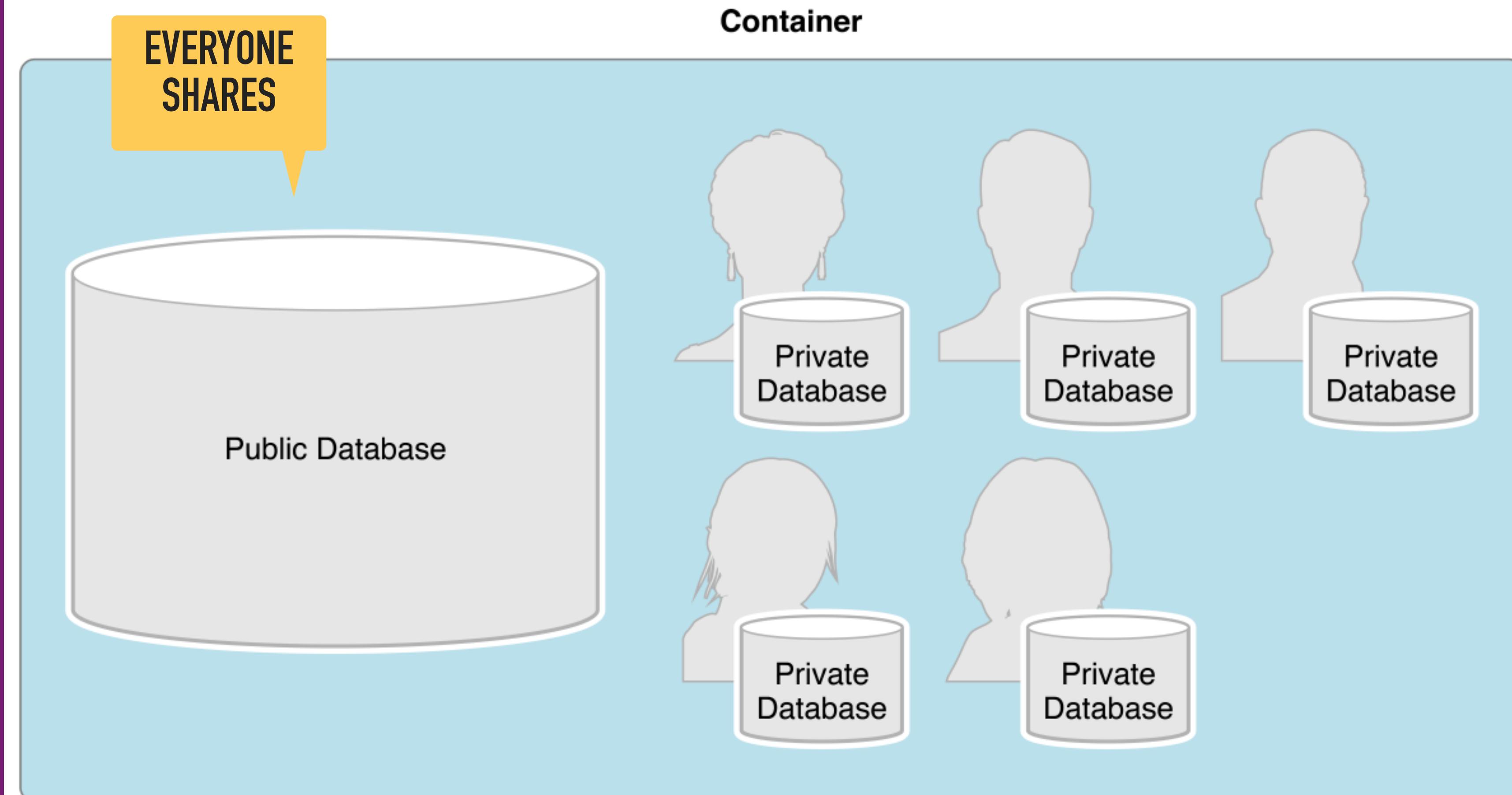
```
let container: CKContainer = CKContainer.default()
```

DATABASE

CLOUDKIT OBJECTS

DATABASE

- CKDatabase
 - Public
 - Private
 - Shared



**EVERY USER HAS
THEIR OWN**

CLOUDKIT OBJECTS

DATABASE

```
/// Container
let container: CKContainer = CKContainer.default()

/// Databases
let publicDB: CKDatabase = CKContainer.default().publicCloudDatabase
let privateDB: CKDatabase = CKContainer.default().privateCloudDatabase
```

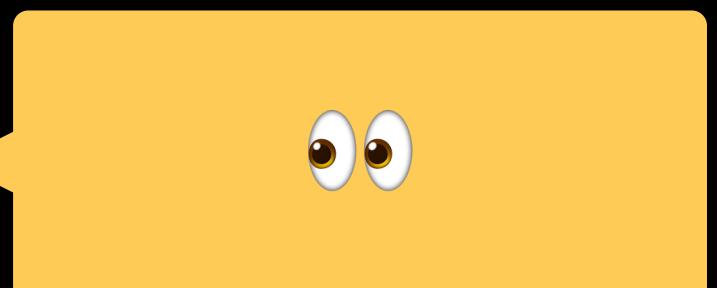
CLOUDKIT OBJECTS

DATABASE

	Public Database	Private Database
Data Type	Shared Data	Current User's Data
Account	Required for Writing	Required
Quota	Developer	User
Default Permissions	World Readable	User Readable
Editing Permissions	iCloud Dashboard Roles	N/A

CLOUDKIT OBJECTS

DATABASE

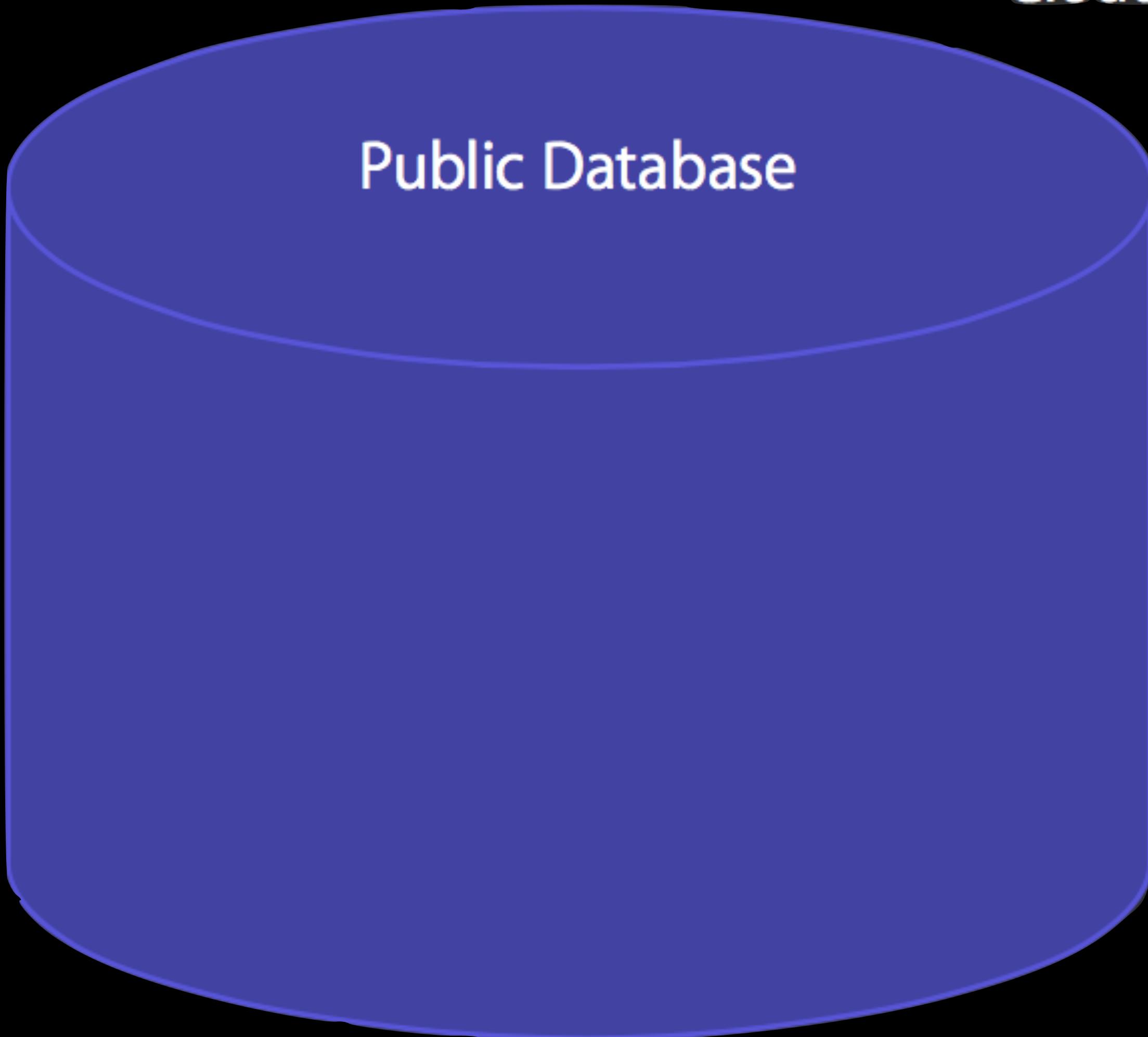
	Public Database	Private Database
Data Type	Shared Data	Current User's Data
Account	Required for Writing	Required
Quota	Developer	User
Default Permissions	World Readable	User Readable
Editing Permissions	iCloud Dashboard Roles	N/A 

CLOUDKIT OBJECTS

DATABASE

- Private databases
 - Atomic writes
 - Delta downloads
 - No cross-zone references

CloudKit Container



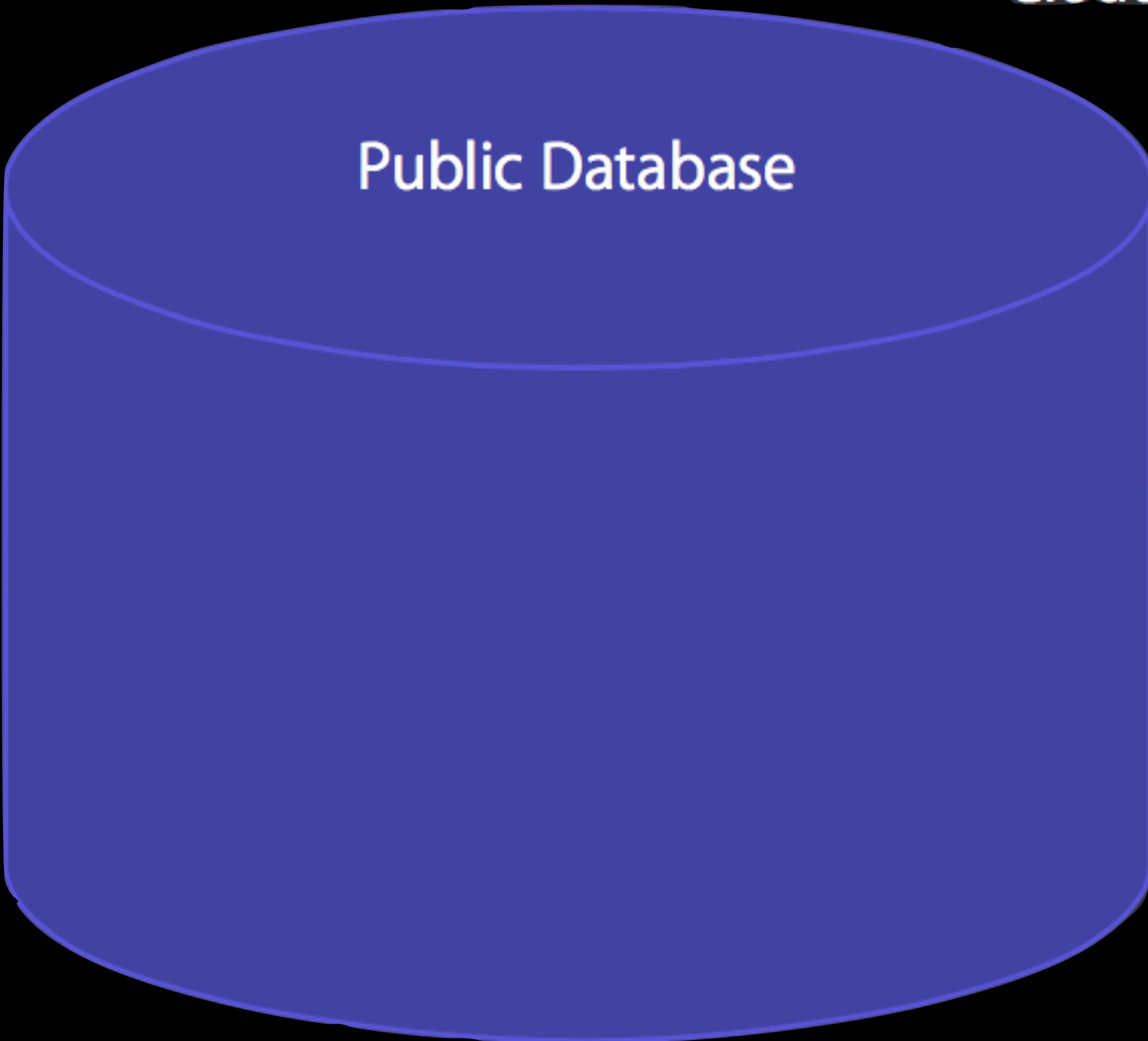
CLOUDKIT OBJECTS

DATABASE

- Public databases
 - No atomic writes
 - No delta downloads
 - Cross zone references

Public Database

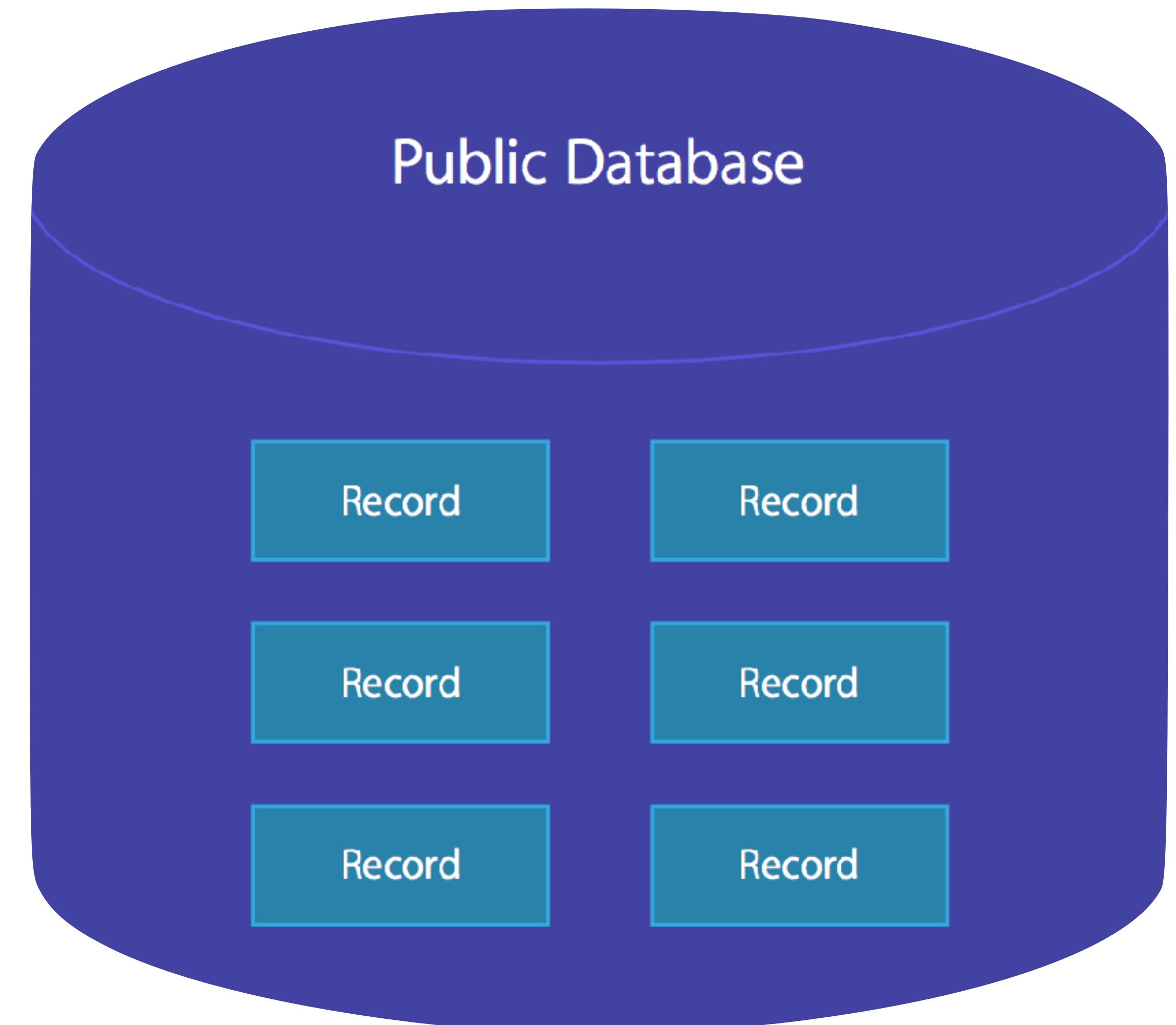
CloudKit Container



CLOUDKIT OBJECTS

DATABASE

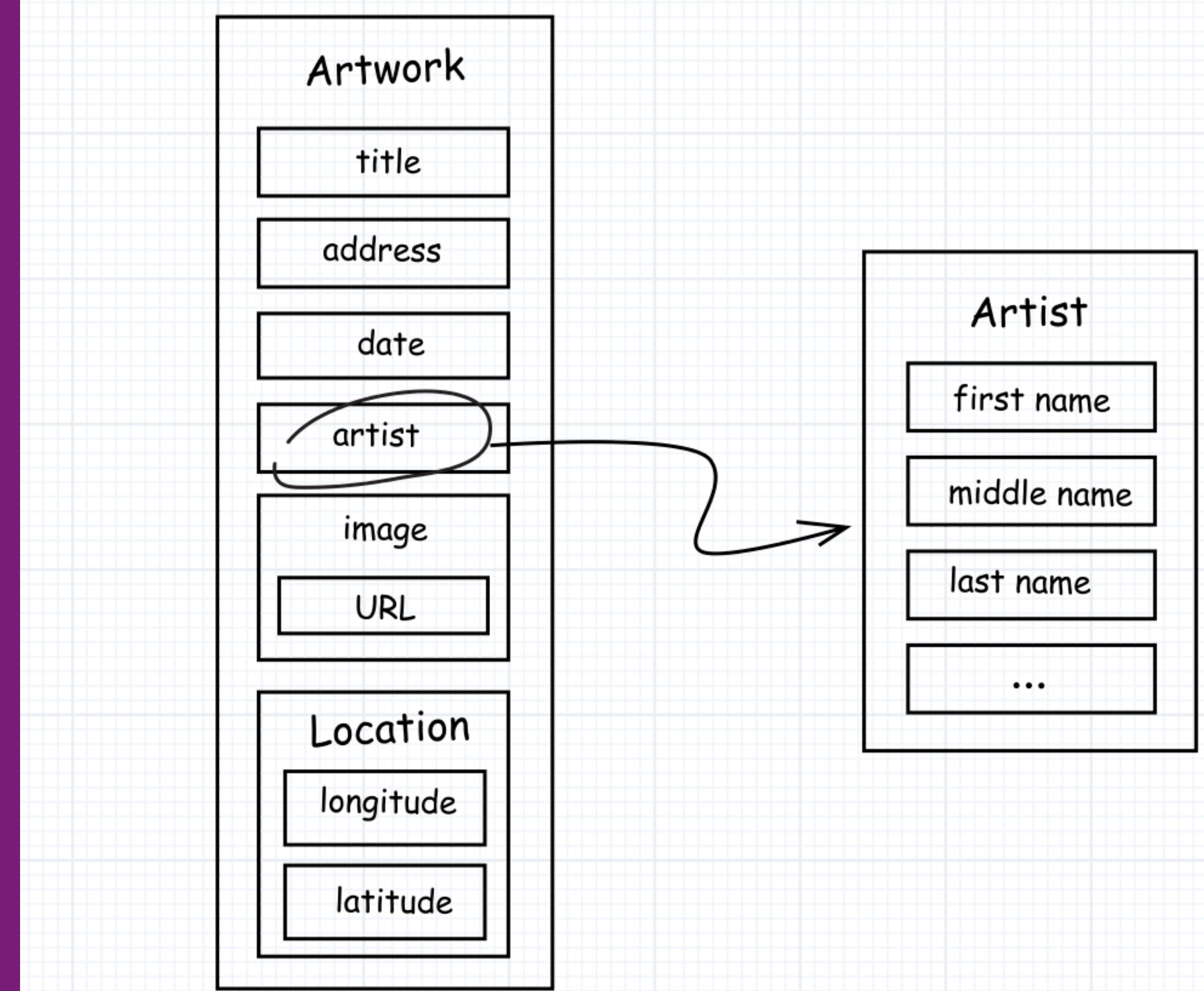
- Structured Data
 - Wraps key/value pairs
 - Types and references
 - Just-in-time schema
 - Metadata
 - Created, edited, etc.



CLOUDKIT OBJECTS

DATABASE

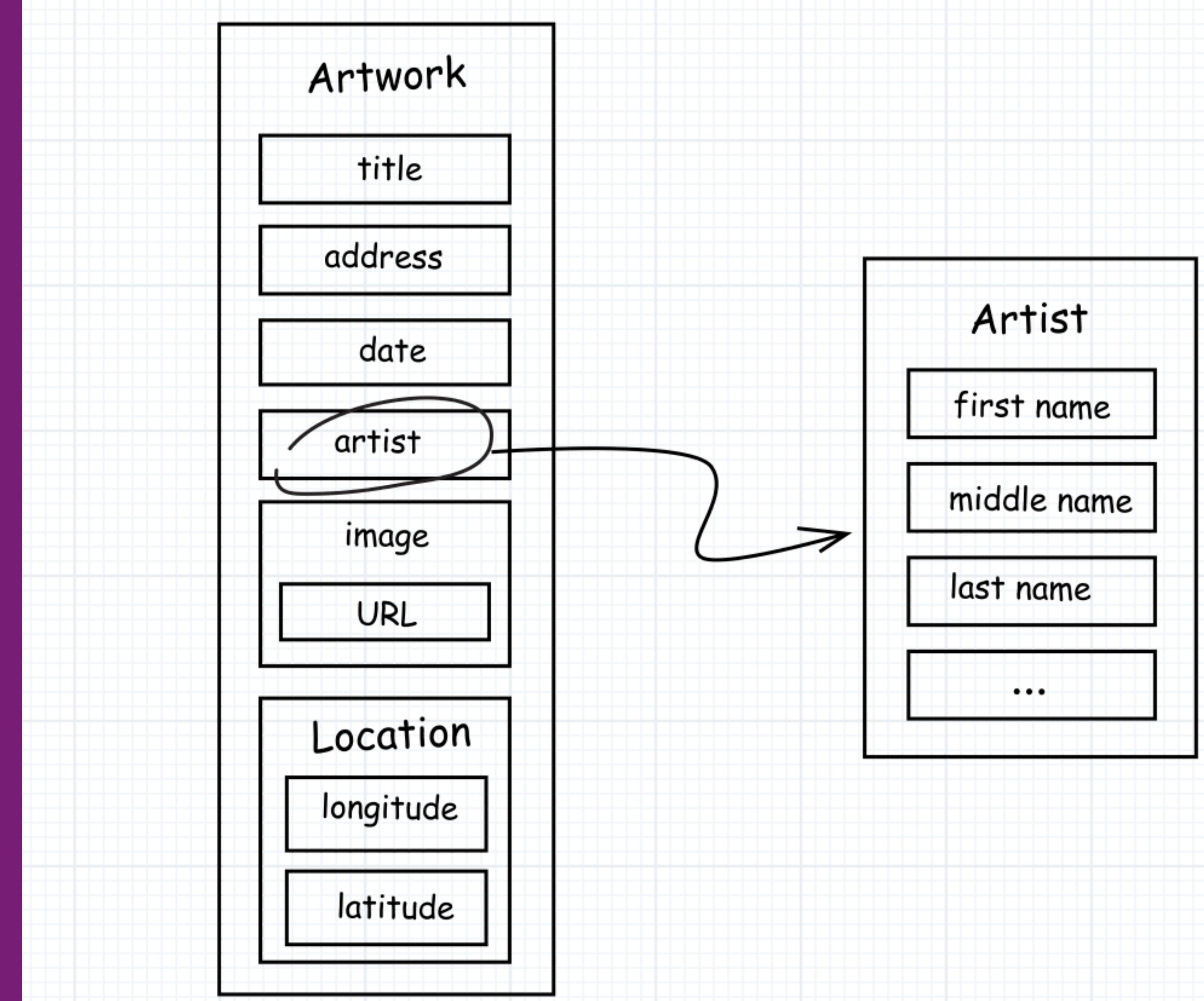
- Schema is built on-demand
- You can edit it in the console



CLOUDKIT OBJECTS

DATABASE

- Changing it may be painful
 - Requires coordination between device and console

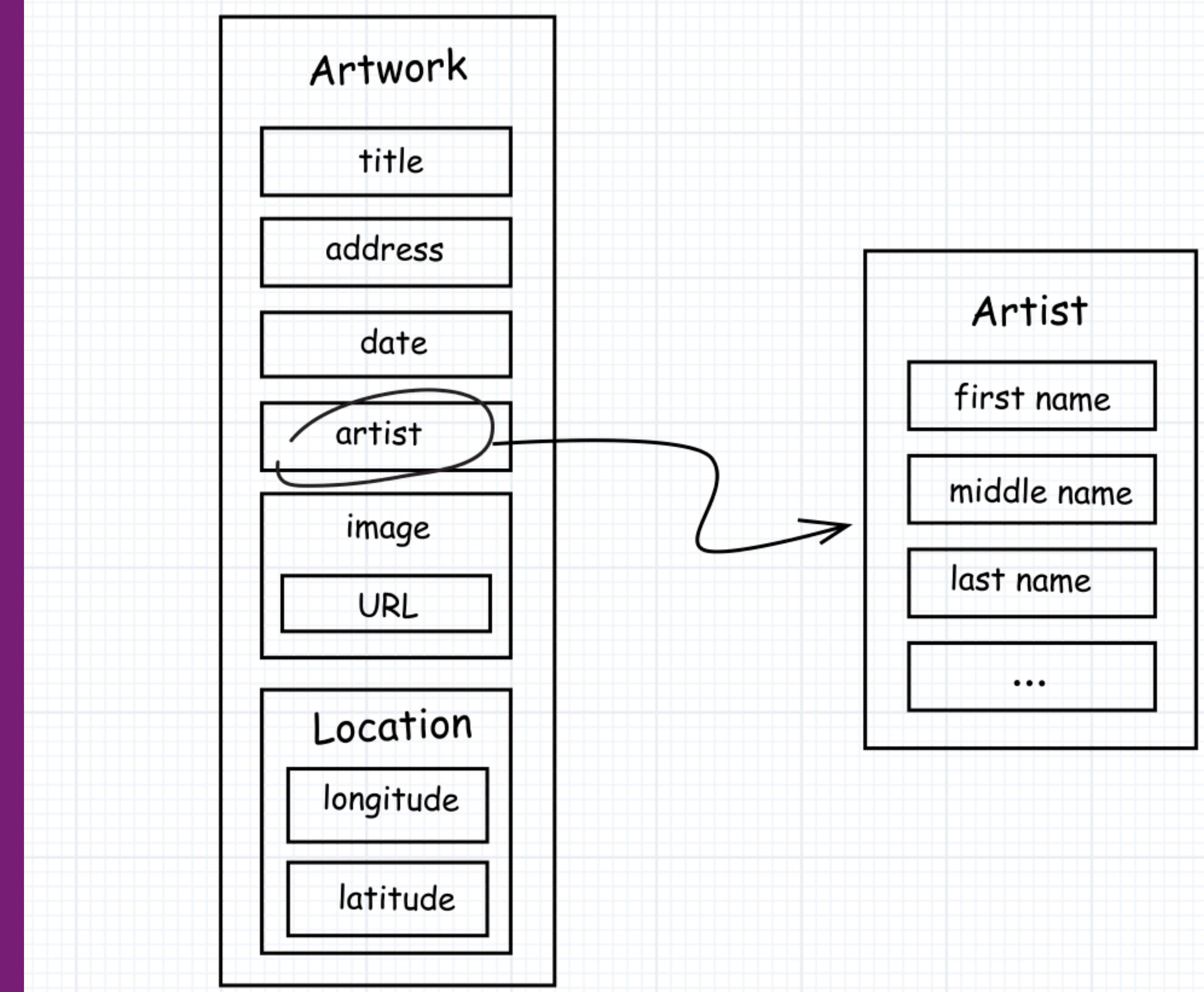


RECORDS

CLOUDKIT OBJECTS

RECORDS

- Record Values
 - String
 - Number
 - Data
 - Date
 - CLLocation
 - CKReference
 - CKAsset
 - Arrays of the above

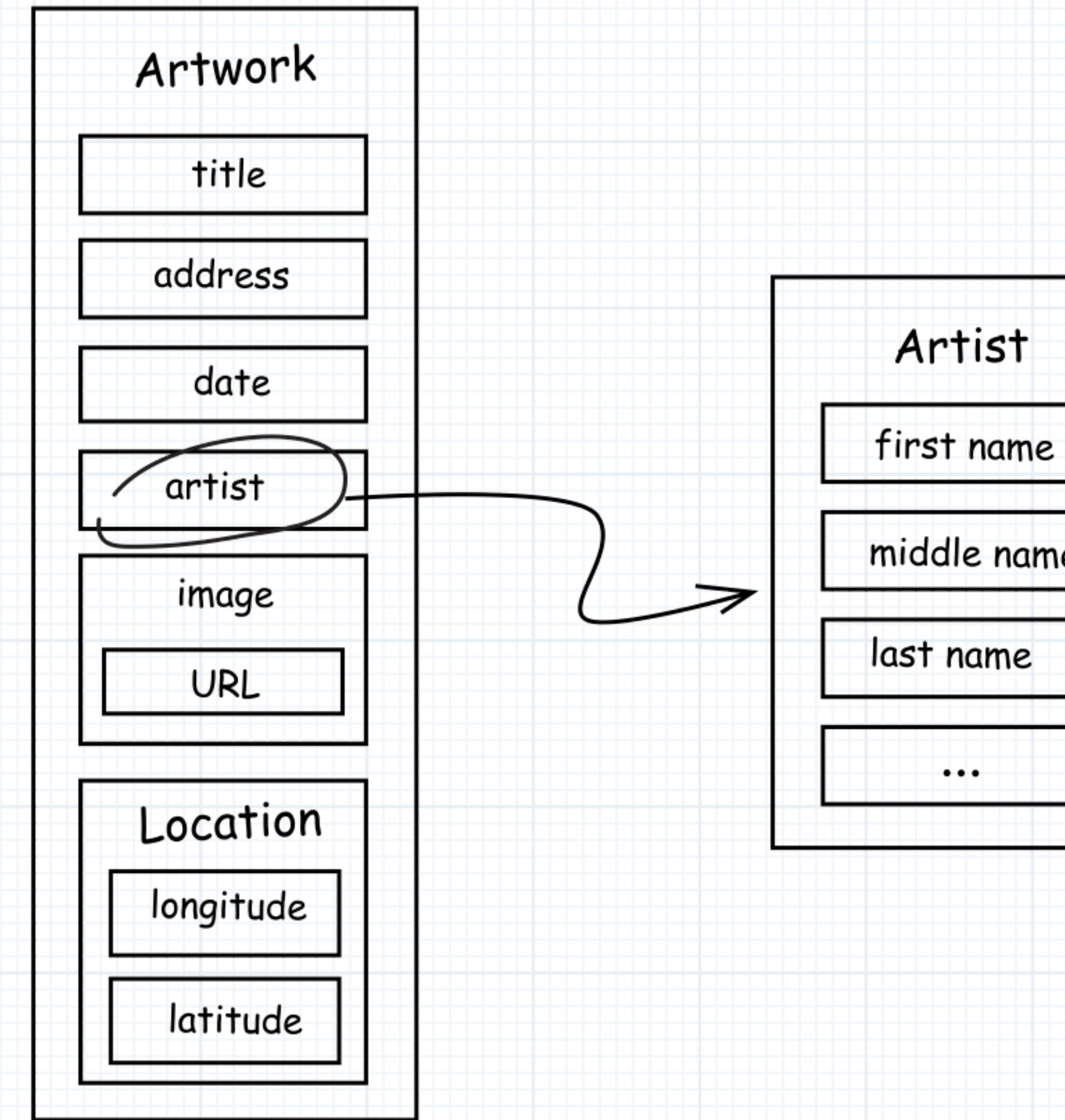


CLOUDKIT OBJECTS

RECORDS

- Records are uniquely stored by a record identifier made up of a combination of the "Zone+Name"

SIMILAR TO APP ENGINE PARENT KEY



CLOUDKIT OBJECTS

RECORDS

- Meta data is automatically assigned to all records
- Used for many tasks

```
/* These create the record in the default zone. */
public init(recordType: String)

public init(recordType: String, recordID: CKRecordID)

public init(recordType: String, zoneID: CKRecordZoneID)

open var recordType: String { get }

@NSCopying open var recordID: CKRecordID { get }

/* Change tags are updated by the server to a unique value every time a record is modified.
   A different change tag necessarily means that the contents of the record are different.*/
open var recordChangeTag: String? { get }

/* This is a User Record recordID, identifying the user that created this record.*/
@NSCopying open var creatorUserRecordID: CKRecordID? { get }

open var creationDate: Date? { get }

/* This is a User Record recordID, identifying the user that last modified this record.*/
@NSCopying open var lastModifiedUserRecordID: CKRecordID? { get }

open var modificationDate: Date? { get }

/*
   In addition to objectForKey: and setObject:forKey:, dictionary-style subscripting (record[key] and record[key] = value)
   used to get and set values.
   Acceptable value object classes are:
   CKReference
   CKAsset
   CLLocation
   NSData
   NSDate
   NSNumber
   NSString
   NSArray containing objects of any of the types above

   Any other classes will result in an exception with name NSInvalidArgumentException.

   Derived field keys are prefixed with '_'. Attempting to set a key prefixed with a '_' will result in an error.

   Key names roughly match C variable name restrictions. They must begin with an ASCII letter and can contain ASCII
   letters and numbers and the underscore character.
   The maximum key length is 255 characters.
*/
open func object(forKey key: String) -> CKRecordValue?
```

CLOUDKIT OBJECTS

RECORDS

The screenshot shows the CloudKit Records interface. On the left, there's a sidebar with filters for 'LOAD RECORDS FROM:', 'USING:', and a 'QUERY FOR RECORDS OF TYPE:' dropdown set to 'joke'. Below these are buttons for 'Query', 'Fetch' (which is selected), and 'Changes'. A large blue button at the bottom says 'Query Records'. At the bottom of the sidebar, a note says 'Query for records of the type "joke" that are in the "_defaultZone" zone of the "public" database.' The main area is a table with columns: Record Name, Record Type, Fields, Ch. Tag, Created, and Modified. The table lists several 'joke' records. A yellow callout box with a large arrow points from the text 'METADATA ON EVERY RECORD' to the 'Fields' column of the table. To the right of the table is a detailed view of a single record. The record has the following metadata:

- Name: 07F1DD4B-4B71-42C1-BA23-F7C85B5B07E2
- Type: joke
- Database: public
- Zone: _defaultZone
- Created: May 10 2017 2:15 AM by Andrew Binkowski (7b892f2a3eeadd6afc77ff7158f69d9)
- Modified: May 10 2017 2:15 AM by Andrew Binkowski (7b892f2a3eeadd6afc77ff7158f69d9)
- Change Tag: j2inhqxw

The detailed view also shows a 'Fields' section with an 'audioFile' asset.

ZONES	RECORDS	RECORD TYPES	INDEXES	SUBSCRIPTIONS	SUBSCRIPTION TYPES	SECURITY ROLES
LOAD RECORDS FROM:	Public Database	Record Name	Record Type	Fields	Ch. Tag	Created
	_defaultZone	▶ 07F1DD4B-4B71...	joke	question, response, ra...	j2inh...	Wed May 10...
		▶ 297120a2-ee03...	joke	question	j34nb...	Thu May 25...
		▶ 2AB4B648-27C...	joke	question, response, ra...	j2q5f...	Mon May 15...
		▶ 2D62FE46-3816...	joke	question, response, ra...	j2inh...	Wed May 10...
		▶ 4523F31B-4521...	joke	question, response, ra...	j2q5f...	Mon May 15...
		▶ 552A9D5F-654...	joke	question, response	j2jl5lgr	Wed May 10 20...
		▶ 5F398818-EF05...	joke	question, response, ra...	j2jm8...	Wed May 10 20...
		▶ 60d82b7d-32ef...	joke	question, response	j2gst...	Mon May 08 20...
		▶ 72FD600E-CCF0...	joke	question, response, ra...	j2q5i...	Mon May 15 20...
		▶ 80645aee-b1d4...	joke	question	j34nc...	Thu May 25 20...
		▶ 8699E4E0-D1A7...	joke	question, response, ra...	j2q5e...	Mon May 15 20...
		▶ 8F1F43F2-1F61...	joke	question, response, ra...	j2q5h...	Mon May 15 20...

METADATA ON EVERY RECORD

Editing Record

CLOUDKIT OBJECTS

RECORDS

A screenshot of the CloudKit Development Data console in a web browser. The title bar shows the URL `iCloud.cloud.uchicago.CloudyWithAChanceOfErros > Development Data` and the user `ANDREW BINKOWSKI`. The top navigation bar has tabs for `ZONES`, `RECORDS` (which is selected), `RECORD TYPES`, `INDEXES`, `SUBSCRIPTIONS`, `SUBSCRIPTION TYPES`, and `SECURITY ROLES`. A yellow callout box points from the `RECORDS` tab to the text **RECORDS ARE DEFINED PROGRAMMATICALLY OR IN THE CONSOLE**. On the left, there are sections for `LOAD RECORDS FROM:` (set to `Private Database abinkowski@uchicago.edu`), `_defaultZone`, `USING:` (with tabs for `Query` (selected), `Fetch`, and `Changes`), `QUERY FOR RECORDS OF TYPE:` (set to `Alert`), `Filter by:` (with a button `Add filters...`), and `Sort by:` (with a button `Add sorts...`). At the bottom right, a message says **No "Alert" records were found in this zone matching this query.**

**RECORDS ARE DEFINED PROGRAMMATICALLY OR
IN THE CONSOLE**

No "Alert" records were found in this zone matching this query.

CLOUDKIT OBJECTS

RECORDS

```
/// Create a joke record and save to iCloud using CloudKit
/// - parameter joke: Funny string
/// - remark: Error handling leaves something to be desired
func saveJoke(_ joke: String) {

    let record = CKRecord(recordType: "joke")
    record.setValue(joke, forKey: "question")
    record["response"] = "To get to the other side." as CKRecordValue
    record["rating_positive"] = 0 as CKRecordValue
    record["rating_negative"] = 0 as CKRecordValue
}
```

CREATE A RECORD

```
publicDB.save(record) { (record, error) in
    if let error = error {
        print("Error: \(error.localizedDescription)")
        return
    }
    print("Saved record: \(record.debugDescription)")
}
```

CLOUDKIT OBJECTS

RECORDS

```
/// Create a joke record and save to iCloud using CloudKit
/// - parameter joke: Funny string
/// - remark: Error handling leaves something to be desired
func saveJoke(_ joke: String) {

    let record = CKRecord(recordType: "joke")
    record.setValue(joke, forKey: "question")
    record["response"] = "To get to the other side." as CKRecordValue
    record["rating_positive"] = 0 as CKRecordValue
    record["rating_negative"] = 0 as CKRecordValue
```

```
publicDB.save(record) { (record, error) in
    if let error = error {
        print("Error: \(error.localizedDescription)")
        return
    }
    print("Saved record: \(record.debugDescription)")
}
```

SAVE THE
RECORD

CLOUDKIT OBJECTS

RECORDS

- Create a Joke record in our view controller
- Data is key-value pairs

```
// MARK: - CloudKit
let container: CKContainer = CKContainer.default()
let publicDB: CKDatabase = CKContainer.default().publicCloudDatabase
let privateDB: CKDatabase = CKContainer.default().privateCloudDatabase

/// Create a joke record and save to iCloud using CloudKit
/// - parameter joke: Funny string
/// - remark: Error handling leaves something to be desired
func saveJoke(_ joke: String) {
    let record = CKRecord(recordType: "joke")
    record.setValue(joke, forKey: "question")
    record["response"] = "To get to the other side." as CKRecordValue

    publicDB.save(record) { (record, error) in
        if let error = error {
            print("🐞: \(error.localizedDescription)")
            return
        }
        print("Saved record: \(record.debugDescription)")
    }
}
```

CLOUDKIT OBJECTS

RECORDS

```
let container: CKContainer = CKContainer.default()
let publicDB: CKDatabase = CKContainer.default().publicCloudDatabase
let privateDB: CKDatabase = CKContainer.default().privateCloudDatabase

override func viewDidAppear(_ animated: Bool) {
    saveJoke("Why did the chicken cross the road?")
}

/// Create a joke record and save to iCloud using CloudKit
/// - parameter joke: Funny string
/// - remark: Error handling leaves something to be desired
func saveJoke(_ joke: String) {
    let record = CKRecord(recordType: "joke")
    record.setValue(joke, forKey: "text")
    publicDB.save(record) { (record, error) in
        if let error = error {
            print("🐞: \(error.localizedDescription)")
            return
        }
        print("Saved record: \(record.debugDescription)")
    }
}
```

YOUR FIRST CLOUDKIT
ERROR

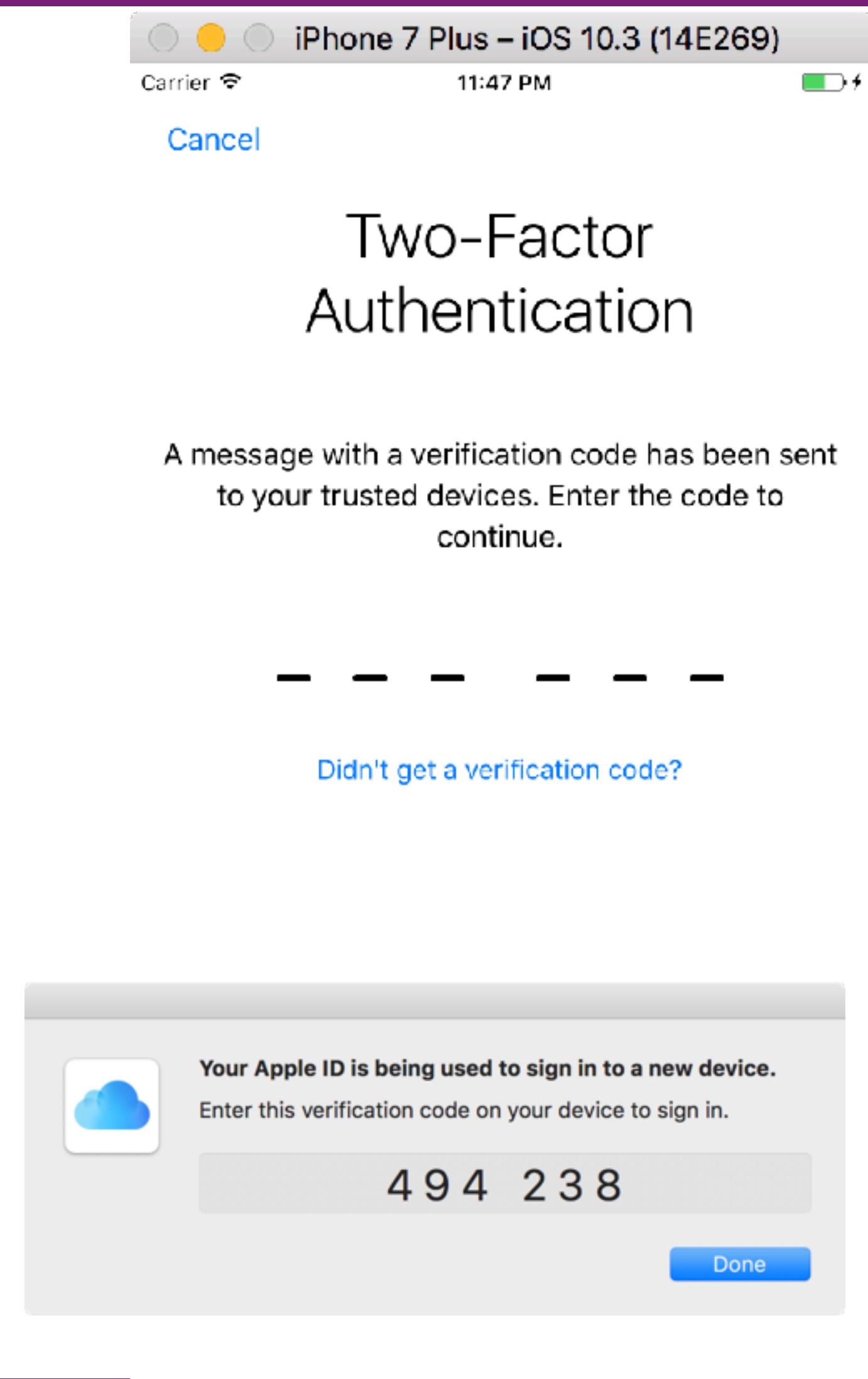
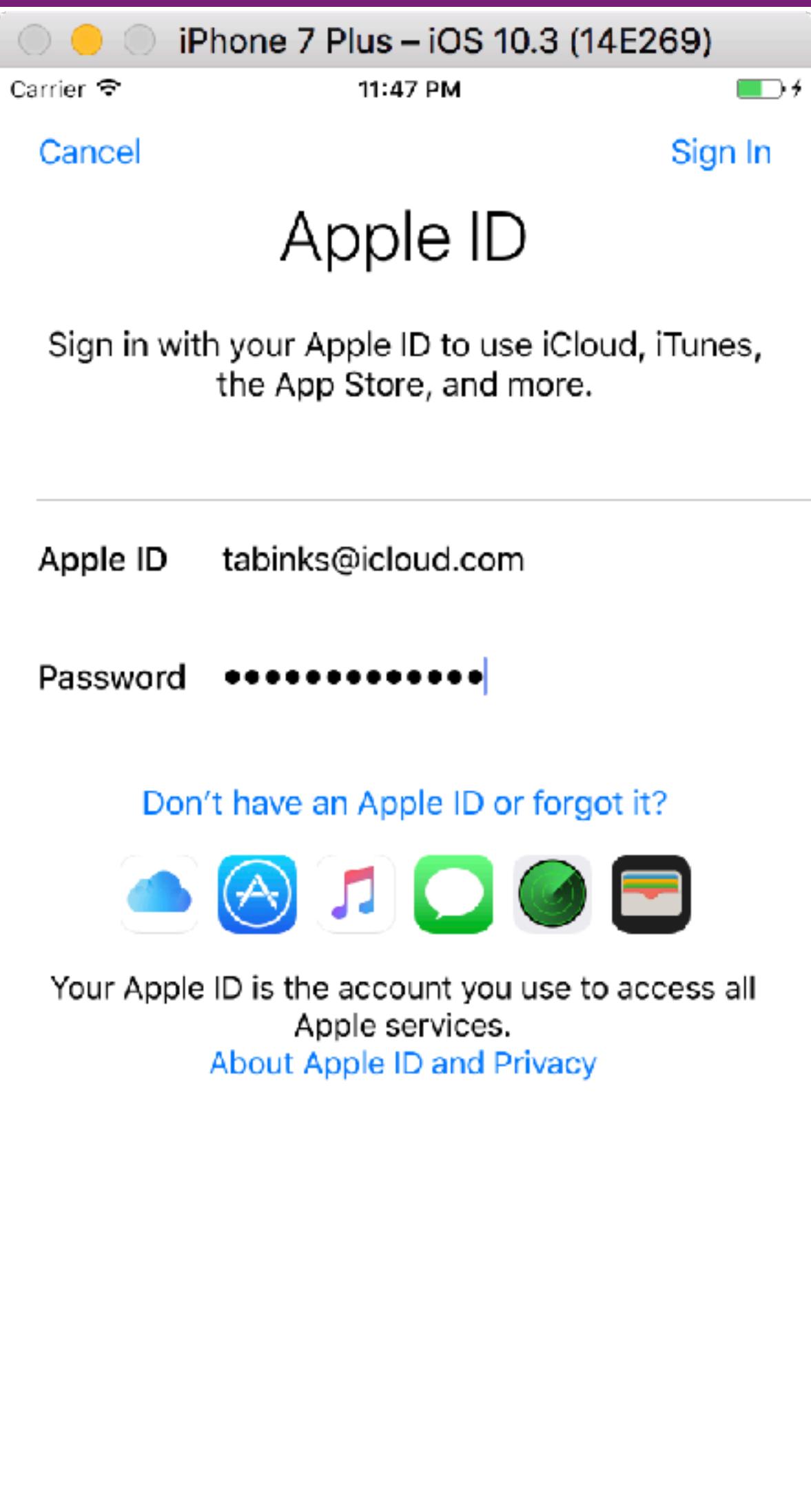
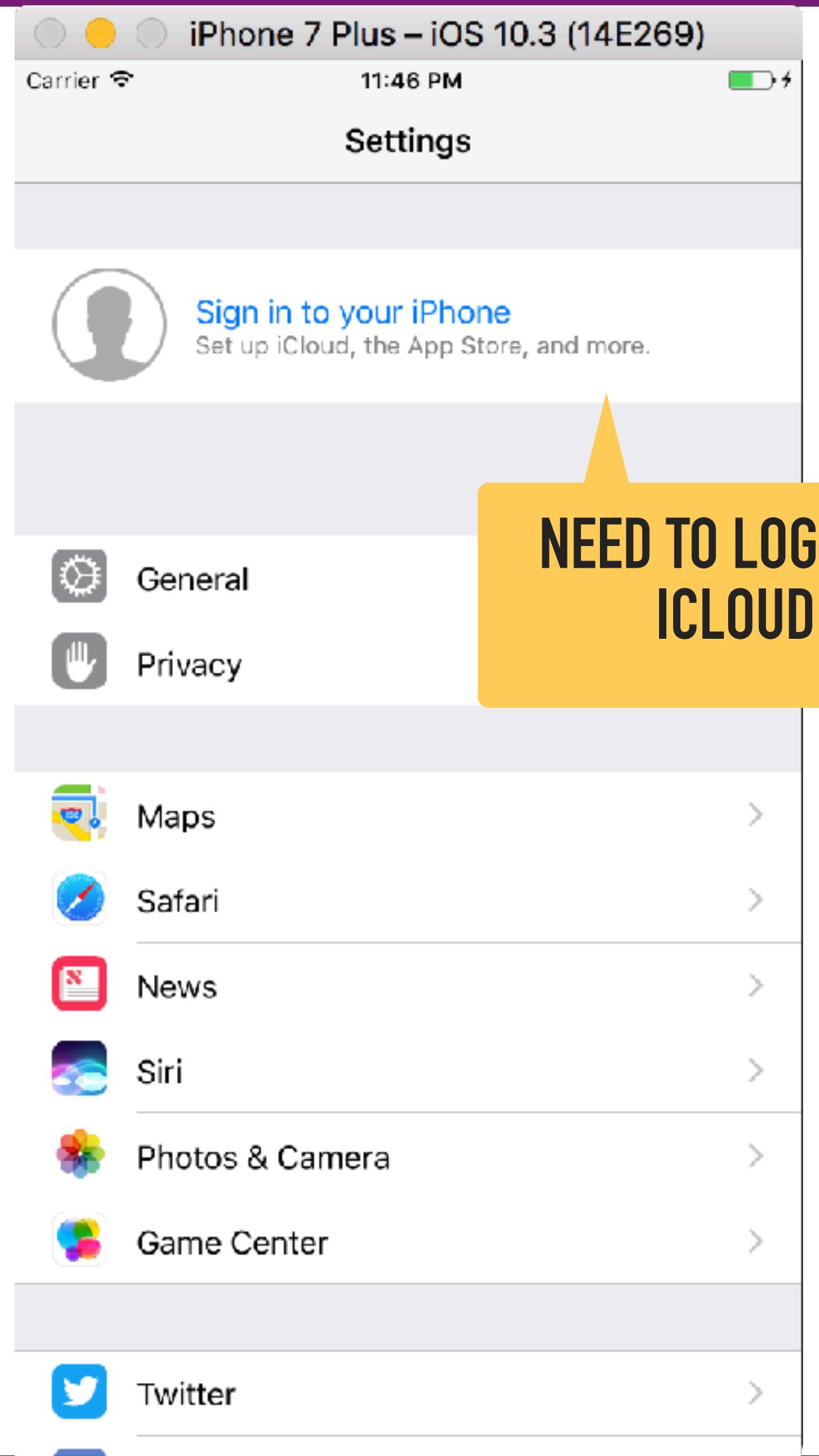


🐞: This request requires an authenticated account

CLOUDKIT OBJECTS

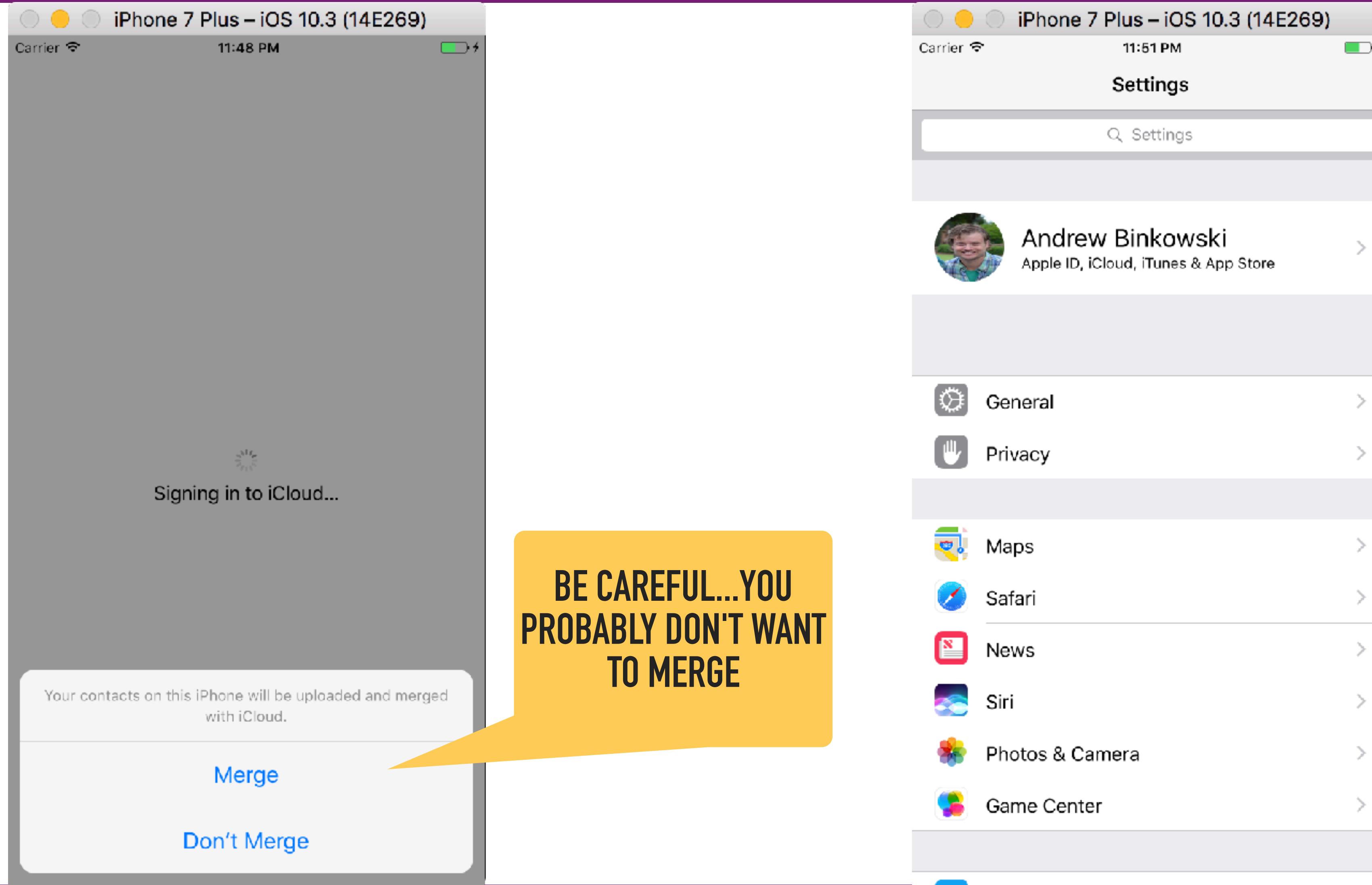
RECORDS

FULLY SUPPORTED BY
THE SIMULATOR



CLOUDKIT OBJECTS

RECORDS



CLOUDKIT OBJECTS

RECORDS

```
let record = CKRecord(recordType: "joke")
record.setValue(joke, forKey: "question")
record["response"] = "To get to the other side." as CKRecordValue

publicDB.save(record) { (record, error) in
    if let error = error {
        print("Error: \(error.localizedDescription)")
        return
    }
    print("Saved record: \(record.debugDescription)")
}
}
```



The screenshot shows a Xcode interface with a code editor and a terminal window.

Code Editor:

```
let record = CKRecord(recordType: "joke")
record.setValue(joke, forKey: "question")
record["response"] = "To get to the other side." as CKRecordValue

publicDB.save(record) { (record, error) in
    if let error = error {
        print("Error: \(error.localizedDescription)")
        return
    }
    print("Saved record: \(record.debugDescription)")
}
}
```

Terminal Output:

```
CloudyWithAChanceOfErrors
Saved record: Optional(<CKRecord: 0x7fad84001550; recordID=025F8A03-2047-4B26-98A0-A97CCA823DDB: (_defaultZone:_defaultOwner__), recordChangeTag=j28i9sj2, values={ text = "Why did the chicken cross the road?"; }, recordType=joke>
{
    creatorUserRecordID -> <CKRecordID: 0x610000028d40; recordName=_defaultOwner_, zoneID=_defaultZone:_defaultOwner_>
    lastModifiedUserRecordID -> <CKRecordID: 0x6100000290c0; recordName=_defaultOwner_, zoneID=_defaultZone:_defaultOwner_>
    creationDate -> 2017-05-03 04:51:41 +0000
    modificationDate -> 2017-05-03 04:51:41 +0000
    modifiedByDevice -> 409DEB7E403CA0DC6733851717BE926CA25C7D3B815F98929B7769596EA32D29
    text -> "Why did the chicken cross the road?"
```

CLOUDKIT OBJECTS

RECORDS

CloudKit > iCloud.cloud.uchicago.CloudyWithAChanceOfErrors > Development Data ANDREW BINKOWSKI

ZONES	RECORDS	RECORD TYPES	INDEXES	SUBSCRIPTIONS	SUBSCRIPTION	EDITING RECORD	X	
LOAD RECORDS FROM:	Public Database	Record N...	Reco...	Fields	C...	Created	Modified	audioFile ASSET
_defaultZone		▶ 07F1DD4...	joke	question, resp	j2i...	Wed Ma...	Wed Ma...	Drag file here to upload. Choose File
USING:		▶ 297120a...	joke	question	j3...	Thu May...	Thu May...	comments STRING (LIST)
		▶ 2AB4B64...	joke	question, resp	j2...	Mon Ma...	Mon Ma...	+ There are no items in this list.
		▶ 2D62FE4...	joke	question, resp	j2i...	Wed Ma...	Wed Ma...	question STRING
		▶ 4523F31...	joke	question, resp	j2...	Mon Ma...	Mon Ma...	Why did the chicken cross the road?
		▶ 552A9D5...	joke	question, resp	j2j...	Wed Ma...	Wed Ma...	
		▶ 5F39881...	joke	question, resp	j2j...	Wed Ma...	Wed Ma...	
		▶ 60d82b7...	joke	question, resp	j2...	Mon Ma...	Tue May...	
		▶ 72FD600...	joke	question, resp	j2...	Mon Ma...	Mon Ma...	
		▶ 80645ae...	joke	question	j3...	Thu May...	Thu May...	

Query Records

Query for records of type "joke" that are in

CLOUDKIT OBJECTS

RECORDS

LOAD RECORDS FROM:

Public Database

_defaultZone

USING:

Query Fetch Changes

QUERY FOR RECORDS OF TYPE:

joke

Filter by: Add filters...

Sort by: Add sorts...

Query Records

Query for records of the type "joke" that are in the "_defaultZone" zone of the "public" database.

Record N...	Reco...	Fields	C...	Created	Modified	
► 07F1DD4...	joke	question, resp	j2i...	Wed Ma...	Wed Ma...	
► 297120a...	joke	question	j3...	Thu May...	Thu May...	
► 2AB4B64...	joke	question, resp	j2...	Mon Ma...	Mon Ma...	
► 2D62FE4...	joke	question, resp	j2i...	Wed Ma...	Wed Ma...	
► 4523F31...	joke	question, resp	j2...	Mon Ma...	Mon Ma...	
► 552A9D5...	joke	question, resp	j2j...	Wed Ma...	Wed Ma...	
► 5F39881...	joke	question, resp	j2j...	Wed Ma...	Wed Ma...	
► 60d82b7...	joke	question, re				
► 72FD600...	joke	question, re				
► 80645ae...	joke	question				
► 8699E4E...	joke	question, resp	j2...	Mon Ma...	Mon Ma...	
► 8F1F43F...	joke	question, resp	j2...	Mon Ma...	Mon Ma...	

audioFile ASSET

Drag file here to upload. Choose File

comments STRING (LIST)

There are no items in this list. +

question STRING

Why did the chicken cross the road?

rating_negative INT(64)

10

rating_positive INT(64)

10

response STRING

To get to the other side

CHANGE SCHEMA FROM CONSOLE OR IN CODE

CLOUDKIT OBJECTS

RECORDS



> iCloud.cloud.uchicago.CloudyWithAChanceOfErrors > Development Data

ANDREW BINKOWSKI ▾

ZONES	RECORDS	RECORD TYPES	INDEXES	SUBSCRIPTIONS	SUBSCRIPTION TYPES	SECURITY ROLES	
DEFAULT TYPES		Field Name		Field Type	Indexes		
Users		audioFile		Asset			X
CUSTOM TYPES		comments		String (List)	Queryable, Searchable		X
Alert		question		String	Queryable, Searchable, Sorta...		X
Daily							
joke		rating_negative		Int(64)	Queryable, Sortable		X
Records the de the na		rating_positive		Int(64)	Queryable, Sortable		X
		response		String	Queryable, Searchable, Sorta...		X
		recordName	SYSTEM FIELD	Reference	Queryable		
		createdBy	SYSTEM FIELD	Reference	Queryable		

CLOUDKIT OBJECTS

RECORDS

CloudKit > iCloud.cloud.uchicago.CloudyWithAChanceOfErrors > Development Data ANDREW BINKOWSKI

ZONES	RECORDS	RECORD TYPES	INDEXES	SUBSCRIPTIONS	SUBSCRIPTION	EDITING RECORD	X
LOAD RECORDS FROM: Public Database	Record N... ▶ 07F1DD4... joke ▶ 297120a... joke ▶ 2AB4B64... joke question, resp j2... Mon Ma... Mon Ma... ▶ 2D62FE4... joke question, resp j2i... Wed Ma... Wed Ma... ▶ 4523F31... joke question, resp j2... Mon Ma... Mon Ma... ▶ 552A9D5... joke question, resp j2j... Wed Ma... Wed Ma... ▶ 5F39881... joke question, resp j2j... Wed Ma... Wed Ma... ▶ 60d82b7... joke question, resp j2... Mon Ma... Tue May... ▶ 72FD600... joke question, resp j2... Mon Ma... Mon Ma... ▶ 80645ae... joke question j3... Thu May... Thu May...	▶ 07F1DD4... joke ▶ 297120a... joke ▶ 2AB4B64... joke question, resp j2... Mon Ma... Mon Ma... ▶ 2D62FE4... joke question, resp j2i... Wed Ma... Wed Ma... ▶ 4523F31... joke question, resp j2... Mon Ma... Mon Ma... ▶ 552A9D5... joke question, resp j2j... Wed Ma... Wed Ma... ▶ 5F39881... joke question, resp j2j... Wed Ma... Wed Ma... ▶ 60d82b7... joke question, resp j2... Mon Ma... Tue May... ▶ 72FD600... joke question, resp j2... Mon Ma... Mon Ma... ▶ 80645ae... joke question j3... Thu May... Thu May...	audioFile ASSET Drag file here to upload. Choose File	comments STRING (LIST) There are no items in this list. +	question STRING Why did the chicken cross the road?	rating_negative INT(64) 10	rating_positive INT(64) 10

USING:
Query Fetch Changes

QUERY FOR RECORDS OF TYPE:
joke
Filter by: Add filters...
Sort by: Add sorts...

Query Records

ADDED NEW FIELD

RECORD ZONES

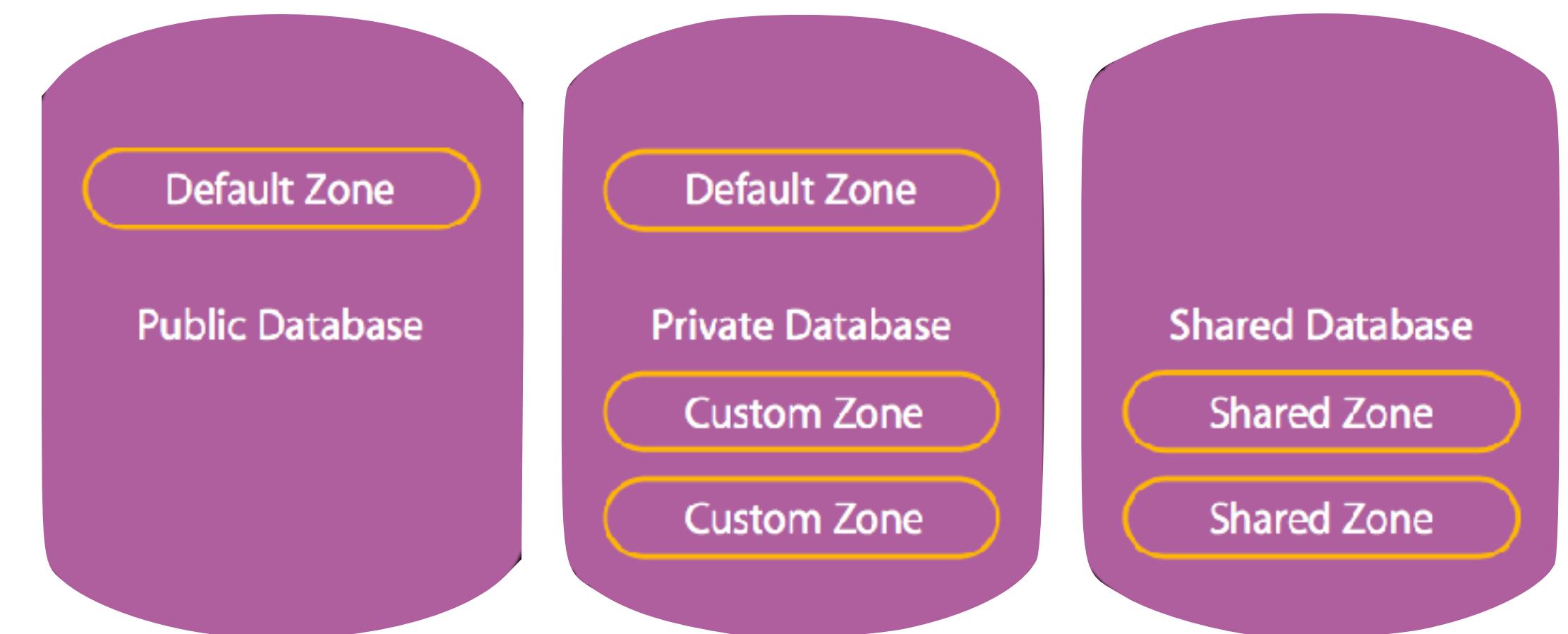
CLOUDKIT OBJECTS

RECORD ZONES

- CKRecordZones

- Zones are a useful way to arrange a discrete group of records
- Supported only in Private and Shared database
- Atomic group writes

PUBLIC HAS A DEFAULT ZONE

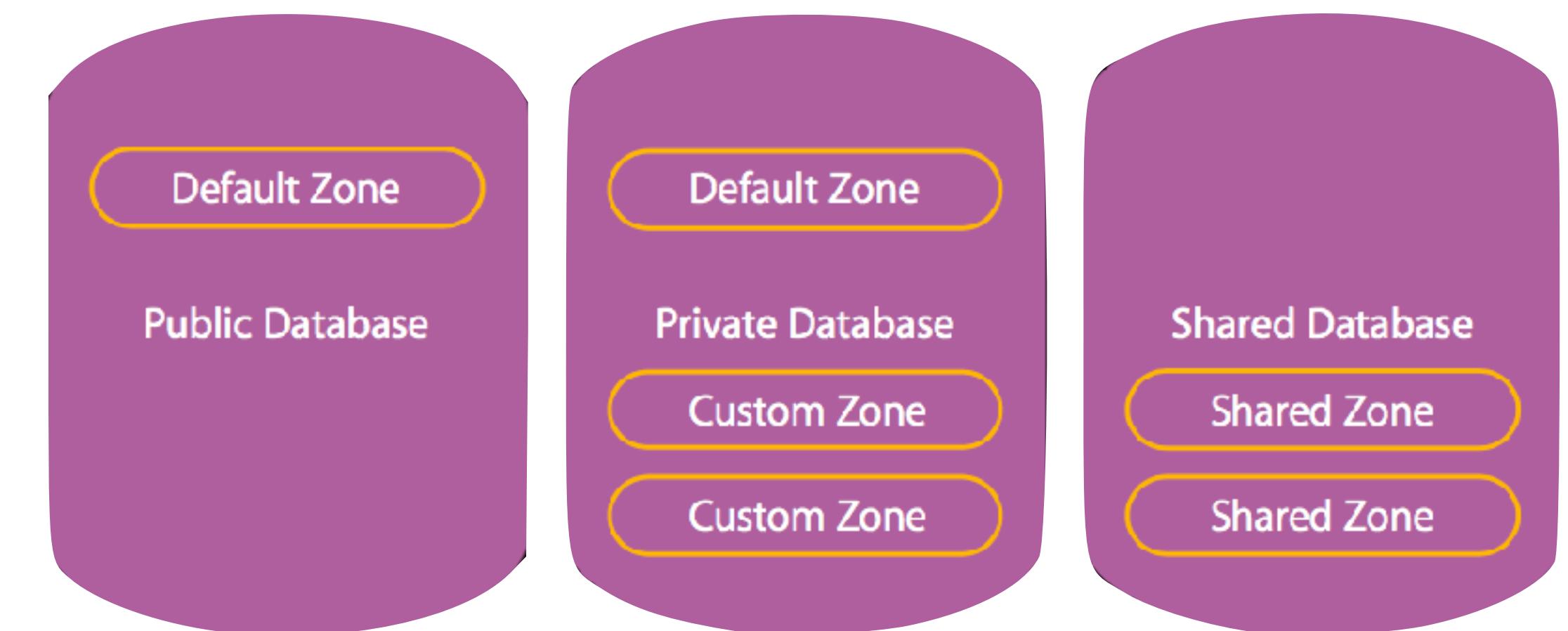


CLOUDKIT OBJECTS

RECORD ZONES

- CKRecordZones

- Zones are a useful way to arrange a discrete group of records
- Supported only in Private and Shared database
- Atomic group writes



This seems familiar

CLOUDKIT OBJECTS

RECORDS

- Created by the client
- They represent the location of the record
- External data set foreign key

```
//  
// CKRecordID.h  
// CloudKit  
//  
// Copyright (c) 2014 Apple  
//  
#import <Foundation/Foundation.h>  
  
@class CKRecordZoneID;  
  
NS_CLASS_AVAILABLE(10_10, 8_0)  
@interface CKRecordID : NSObject <NSSecureCoding>  
- (instancetype)init NS_UNAVAILABLE;  
/* Record names must be 255 characters or less */  
/* This creates a record ID in the default zone */  
- (instancetype)initWithRecordName:(NSString *)  
- (instancetype)initWithRecordName:(NSString *)  
    name  
    zone  
    identifier  
@property (nonatomic, readonly, strong) NSString *name;  
@property (nonatomic, readonly, strong) CKRecordZoneID *zone;  
@end
```



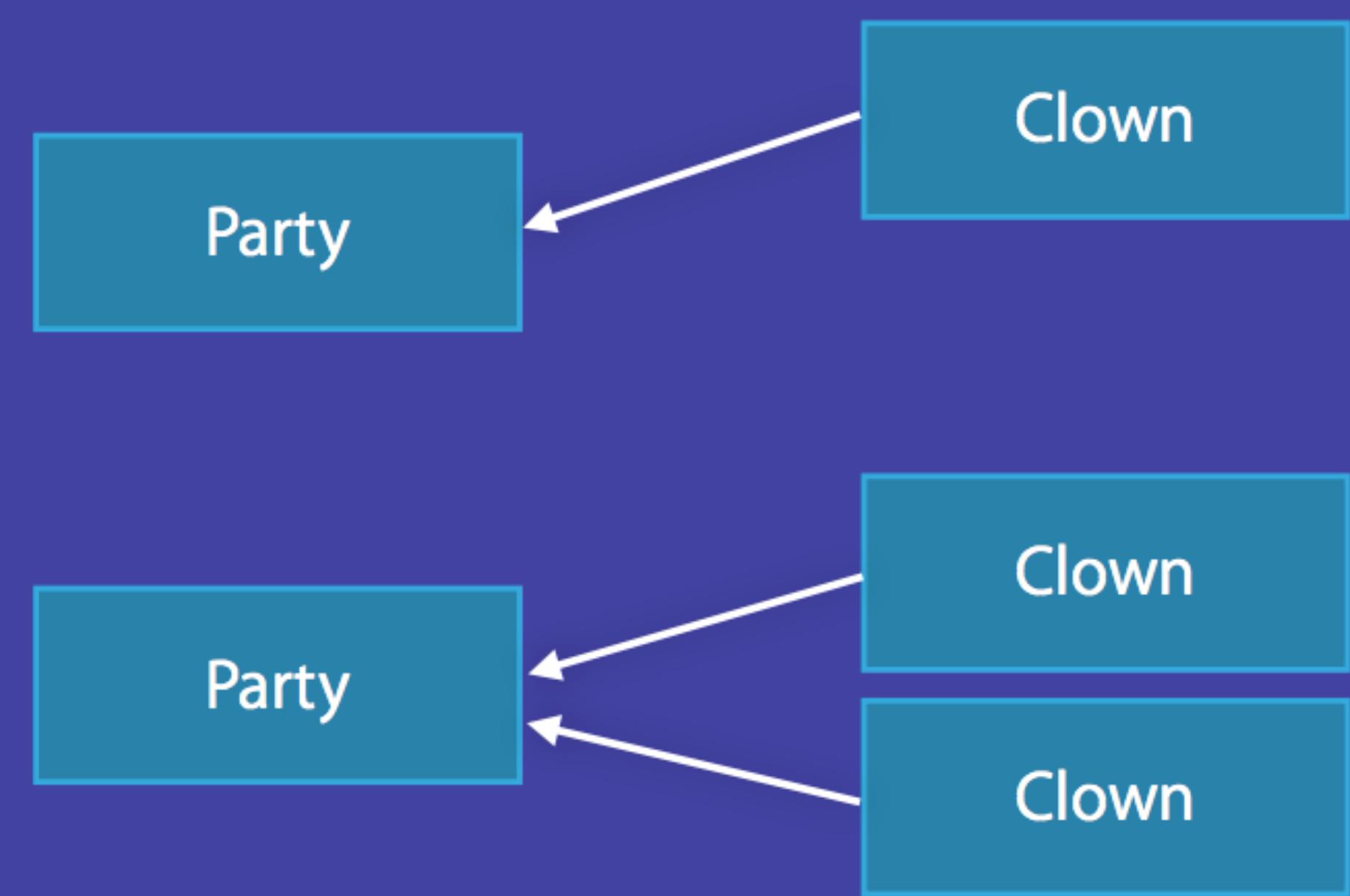
REFERENCES

CLOUDKIT OBJECTS

RECORDS

- CKReference
 - Points to another record
 - Server understands relationship (takes some responsibility for managing; if you want)

Public Database

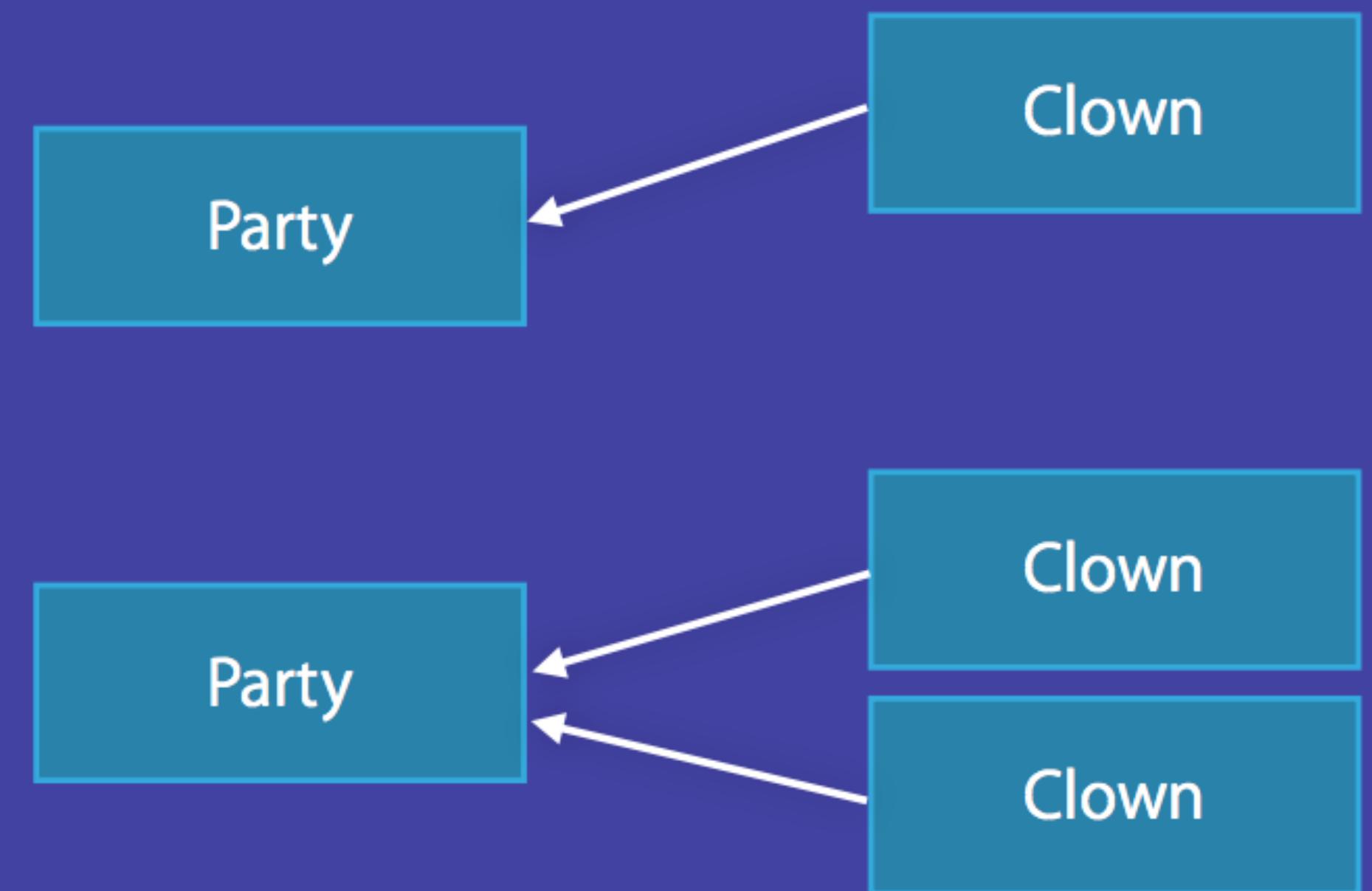


CLOUDKIT OBJECTS

RECORDS

- Cascade Deletes
 - What happens when you delete something with a reference?
 - Dangling Pointers
- Use "Back References" strategy

Public Database



CLOUDKIT API

```
// Create a reference with a record  
// Set the delete 'action' preference  
let reference = CKReference(recordID: recordID,  
                           action: .none)
```

WHAT HAPPENS ON DELETE

CLOUDKIT API

Screenshot of the CloudKit Development Data interface in a web browser.

The URL in the address bar is [iCloud.cloud.uchicago.CloudyWithAChanceOfErros](#) > Development Data.

The sidebar on the left lists record types: ZONES, RECORDS, RECORD TYPES, INDEXES, SUBSCRIPTIONS, SUBSCRIPTION TYPES, SECURITY ROLES, and a dropdown menu.

The RECORD TYPES section is active, showing the following table:

	Field Name	Field Type	Indexes	
DEFAULT TYPES				
Users	audioFile	Asset		X
CUSTOM TYPES	comments	String (List)	Queryable, Searchable	X
Alert	question	String	Queryable, Searchable, Sorta...	X
Daily	rating_negative	Int(64)		X
joke	rating_positive	Int(64)		X
	response	String	Queryable, Searchable, Sorta...	X
	recordName	SYSTEM FIELD	Reference	Queryable

A yellow callout bubble with the word "REFERENCE" points to the "recordName" row.

A note in the sidebar states: "Record types will be automatically created in the development environment when you use the native API to create records of that type."

A blue button at the bottom left says "Create New Type".

CLOUDKIT API

CloudKit API interface showing the Records section.

LOAD RECORDS FROM:

- Public Database
- _defaultZone

USING:

Development Data

RECORD TYPES

Record Name	Record T...	Fields	Ch. T...	Created	Modified
977D8136-0A78-430E-93C2-513B7B0674ED	joke	<p>question Why did the student throw a clock out the window?</p> <p>response She wanted to see if it would fit.</p>	j9pri30j	Tue Nov 07 2017 09:20:06 GM	Tue Nov 07 2017 09:20:06 GM
				T-0600 (CST)	T-0600 (CST)

INDEXES

SUBSCRIPTIONS

SUBSCRIPTION TYPES

SECURITY R

EDITING RECORD

Metadata

- Name: 977D8136-0A78-430E-93C2-513B7B0674ED
- Type: joke
- Database: public
- Zone: _defaultZone
- Created: Nov 7 2017 9:20 AM by _23d865322080d4185ad95db121c80663
- Modified: Nov 7 2017 9:20 AM by _23d865322080d4185ad95db121c80663
- Change Tag: j9pri30j

References

There are no records referenced by this record.

Fields

- audioFile ASSET
- Drag file here to upload. Choose File

comments STRING (LIST)

USER REFERENCE TO RECORD IS BUILT IN

USER REFERENCE TO RECORD IS BUILT IN

USER RECORDS

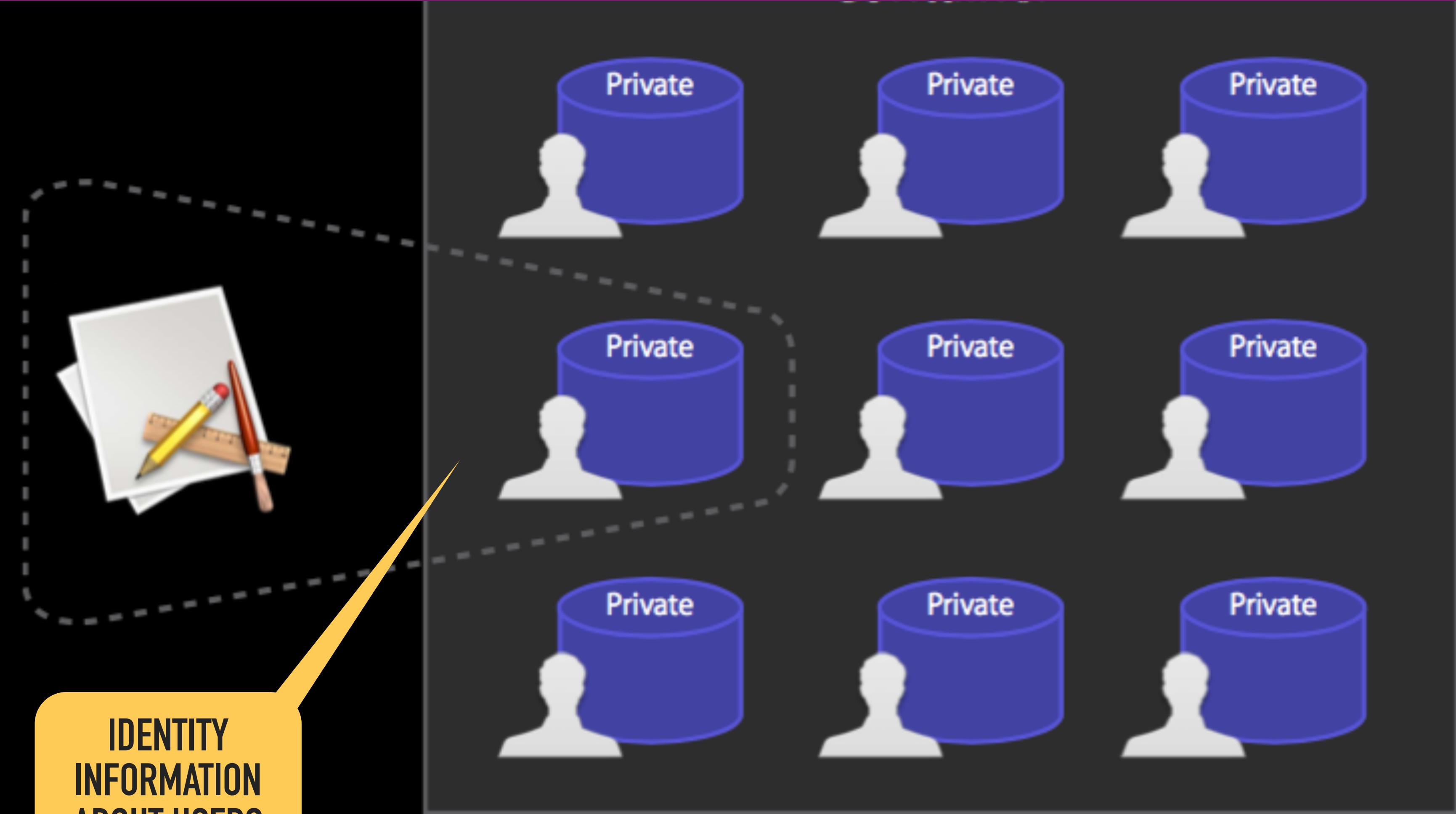
USER RECORDS

- Identity
- Metadata
- Privacy
- Discovery
 - Users can discover friends who use the application

USER RECORDS



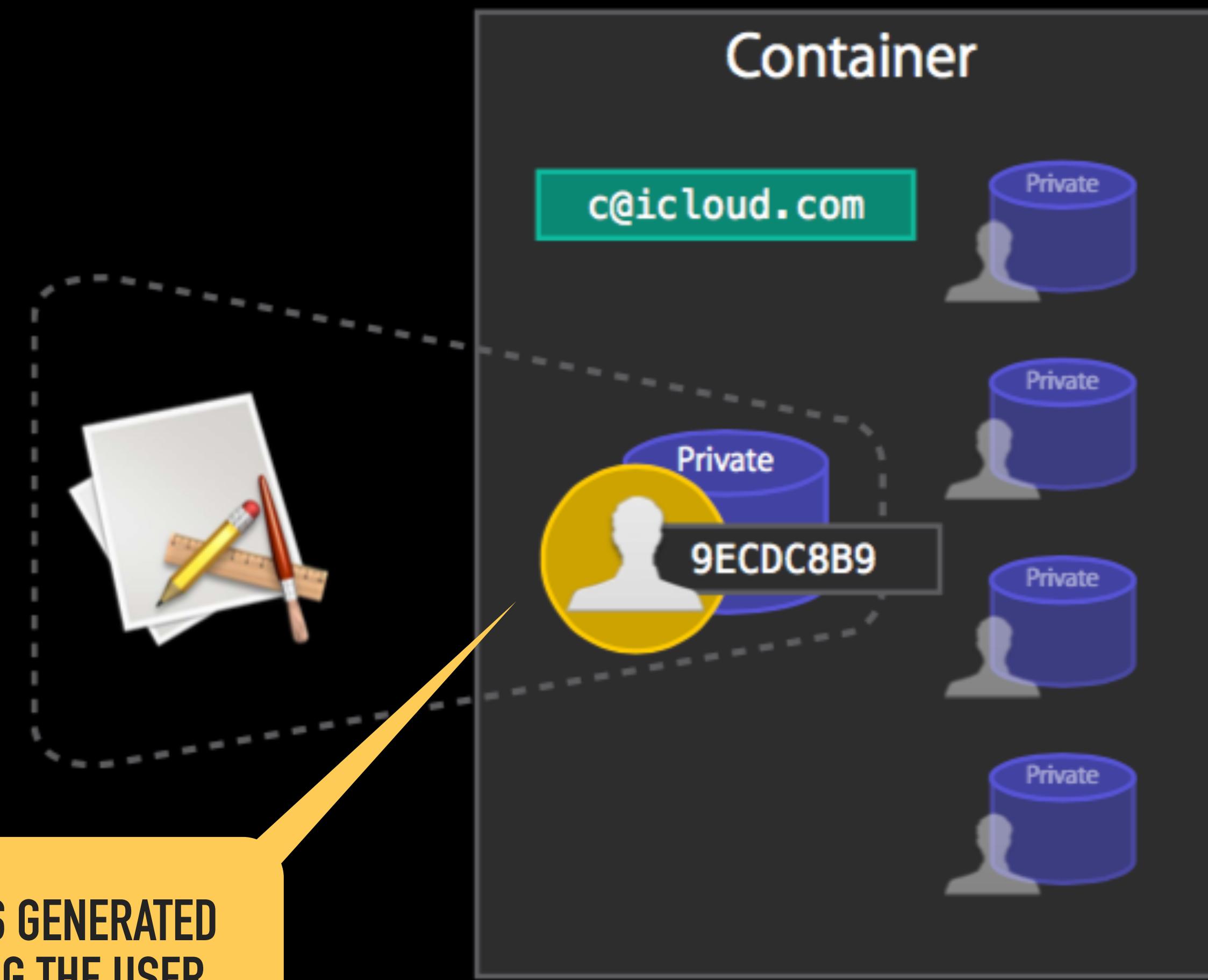
IDENTITY
INFORMATION
ABOUT USERS



USER RECORDS



A UNIQUE ID IS GENERATED
REPRESENTING THE USER



USER RECORDS



iCloud.cloud.uchicago.CloudyWithAChanceOfErros > Development Data

ANDREW BINKOWSKI ▾

ZONES	RECORDS	RECORD TYPES	INDEXES	SUBSCRIPTIONS	SUBSCRIPTION TYPES	SECURITY ROLES	
DEFAULT TYPES		Field Name		Field Type		Indexes	
Users		recordName		SYSTEM FIELD	Reference	Queryable	
CUSTOM TYPES		createdBy		SYSTEM FIELD	Reference		
Alert		createdAt		SYSTEM FIELD	Date/Time		
Daily		modifiedBy		SYSTEM FIELD	Reference		
joke		modifiedAt		SYSTEM FIELD	Date/Time		
Records		roles		SYSTEM FIELD	Int(64) (List)		
the de		changeTag		SYSTEM FIELD	String		
the na							

USER RECORD IS ALREADY
CREATED

Add Field

ADD ADDITIONAL FIELDS TO
CUSTOMIZE YOUR DATA; NOT
RECOMMENDED BY APPLE

Created: May 2 2017 9:55 PM by _1

Modified: May 3 2017 12:27 AM by _7b892f2a3eeadd6afc77ff7158f69d9

USER RECORDS

- To associate a User with other data
 - Create a `User` record type on other record that has a reference to the `UserId`
 - Store any other data user

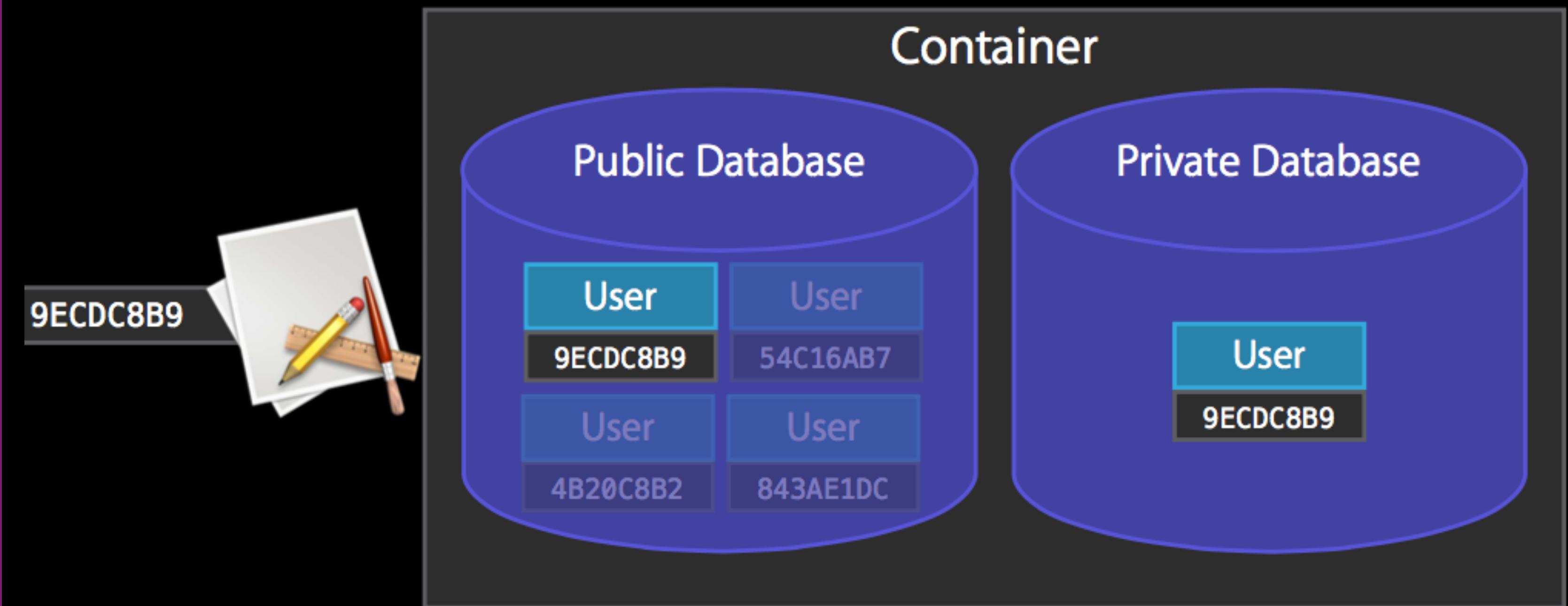


Screenshot of the CloudKit Record Types interface. The top navigation bar shows the iCloud domain: `iCloud.cloud.uchicago.CloudyWithAChanceOfE`. The main menu includes ZONES, RECORDS, RECORD TYPES (which is selected), and INDEXES. The RECORD TYPES section lists several record types: DEFAULT TYPES (Users), and CUSTOM TYPES (Alert, Daily, joke). A note states: "Record types will be automatically created in the development environment when you use the native API to create records of that type." At the bottom right are buttons for "Create New Type" and "Add Field".

Field Name
recordName
createdBy
createdAt
modifiedBy
modifiedAt
roles
changeTag

CloudKit User Accounts

- User metadata is a CKRecordTypeUser record
 - One for each database



UNIQUE TO EACH APPLICATION; WILL NOT BE THE SAME FOR SAME USER IN DIFFERENT APPLICATIONS FROM THE SAME DEVELOPER

USER RECORDS

GET A USER RECORD

```
let container = CKContainer.default()

container.fetchUserRecordID() {
    recordID, error in

        if error != nil {
            print(error!.localizedDescription)
            complete(nil, error as NSError?)
        } else {
            // We have access to the user's record
            print("fetched ID \(recordID?.recordName ?? "")")
            complete(recordID, nil)
        }
}
```

USER RECORDS

```
let container = CKContainer.default()

container.fetchUserRecord(recordID: "23D865322080D4185AD95DB121C80663") { [weak self] error in
    if let error = error {
        print("Error fetching record: \(error.localizedDescription)")
        complete(nil, error as NSError?)
    } else {
        // We have access to the user's record
        print("fetched ID \(recordID?.recordName ?? "")")
        complete(recordID, nil)
    }
}
```

USER RECORDS

"_23D865322080D4185AD95DB121C80663"

ZONES	RECORDS	RECORD TYPES	INDEXES	SUBSCRIPTIONS	SUBSCRIPTION TYPES	SECURITY R	EDITING RECORD	X
LOAD RECORDS FROM:		Record Name	Record T...	Fields	Ch. T...	Created	Modified	
Public Database		_f84642634df...	Users		j29c...	Wed May 03 ...	Wed May 03 ...	
_defaultZone		_23d865322080	Users	d4185ad95db12	j28i9s	Tue May 02 201	Tue May 02 201	X
USING:		gv	7 23:51:41 GMT	-0500 (CDT)	7 23:51:41 GMT	-0500 (CDT)		
QUERY FOR RECORDS OF TYPE:		_7b892f2a3ee...	Users		j28e...	Tue May 02 2...	Tue May 02 2...	
		_7f991400b15...	Users		j2q5...	Mon May 15 2...	Mon May 15 2...	
Filter by: Add filters...								
Sort by: Add sorts...								
Query Records								
Query for records of the type "Users" that are in the "_defaultZone" zone of the "public" database.								
Create New Record...								

USER RECORDS

▼ Metadata

Name: _23d865322080d4185ad95db121c80663

Type: Users

Database: public

Zone: _defaultZone

Created: May 2 2017 11:51 PM
by _23d865322080d4185ad95db121c80663

Modified: May 2 2017 11:51 PM
by _23d865322080d4185ad95db121c80663

Change Tag: j28i9sgv

▼ References 0

There are no records referenced by this record.

▼ Security Roles

This environment has no custom security roles.

▼ Fields 0 of 0

USER RECORDS

```
/// Get the users `RecordID`  
/// - parameters complete: A completion block passing two parameters  
func getUserRecordId(complete: @escaping (CKRecordID?, NSError?) -> ()) {  
  
    let container = CKContainer.default()  
    container.fetchUserRecordID() {  
        recordID, error in  
  
        if error != nil {  
            print(error!.localizedDescription)  
            complete(nil, error as NSError?)  
        } else {  
            // We have access to the user's record  
            print("fetched ID \(recordID?.recordName ?? "")")  
            complete(recordID, nil)  
        }  
    }  
}
```

```
getUserRecordId { (recordID, error) in  
    if let userID = recordID?.recordName {  
        print("iCloudID: \(userID)")  
        self.getJokesByCurrentUser(recordID!)  
    } else {  
        print("iCloudID: nil")  
    }  
}
```

USER RECORDS

- User privacy
 - No disclosure by default
 - Disclosure requested by application
- Opting in allows you to be discoverable by your friends

**Allow people using
"cloud.uchicago.CloudyWith
AChanceOfErros" to look you
up by email?**

People who know your email address
will be able to see that you use this
app.

Don't Allow

OK

USER RECORDS

- Input
 - User RecordID
 - Email address
 - Entire address book
- Output
 - User RecordID
 - First and last names
- Personally identifying information
 - Requires opt-in

**Allow people using
“cloud.uchicago.CloudyWith
AChanceOfErros” to look you
up by email?**

People who know your email address
will be able to see that you use this
app.

Don't Allow

OK

USER RECORDS

GET USER NAME ASSOCIATED
WITH ICLOUD ACCOUNT

```
/// Get the users `RecordId`  
/// - parameters complete: A completion block passing two parameters  
open func getUserIdentity(complete: @escaping (String?, NSError?) -> ()) {  
  
    container.requestApplicationPermission(.userDiscoverability) { (status, error) in  
        self.container.fetchUserRecordID { (record, error) in  
  
            if error != nil {  
                print(error!.localizedDescription)  
                complete(nil, error as NSError?)  
  
            } else {  
                //print("fetched ID \(record?.recordName ?? "")")  
                self.container.discoverUserIdentity(withUserRecordID: record!, completionHandler: { (userID, error) in  
                    let userName = (userID?.nameComponents?.givenName)! + " " + (userID?.nameComponents?.familyName)!  
                    print("CK User Name: " + userName)  
                    complete(userName, nil)  
                })  
            }  
        }  
    }  
}
```

USER RECORDS

```
        print("CK User Name: " + userName)
    }
}
}

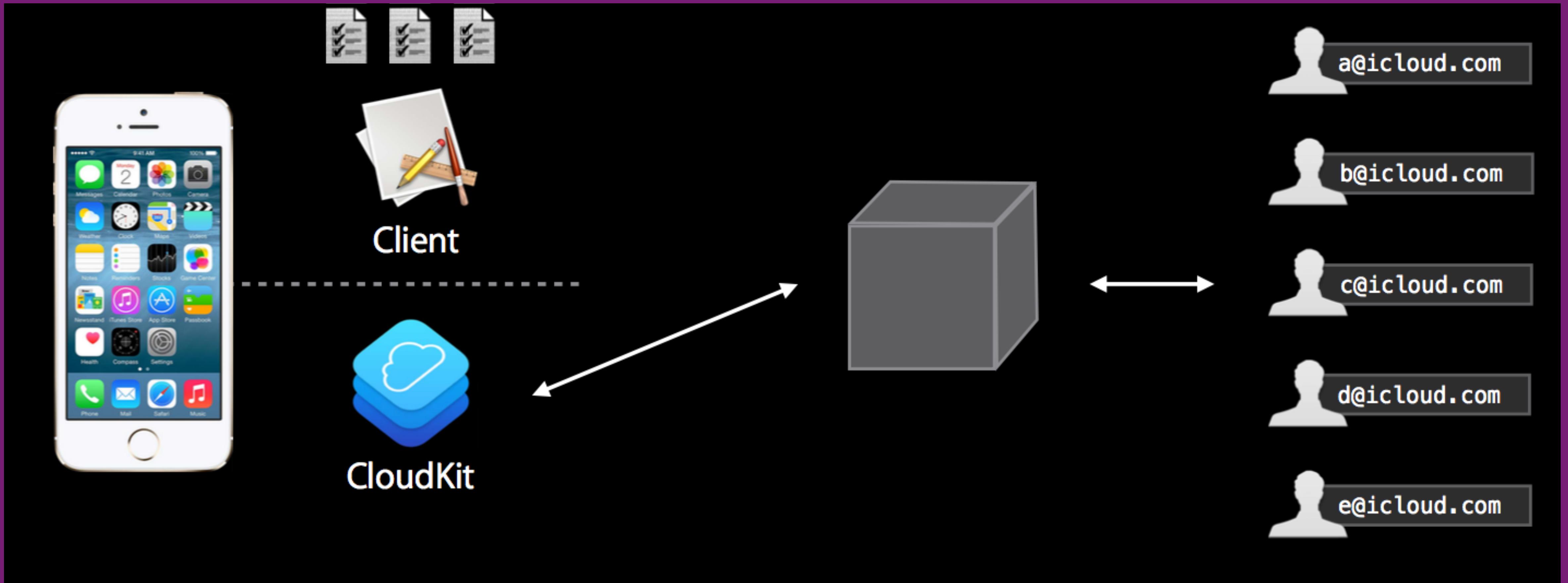
CKContainer.default().requestApplicationPermission(.userDiscoverability) { (status, error) in
    CKContainer.default().fetchUserRecordID { (record, error) in
        CKContainer.default().discoverUserIdentity(withUserRecordID: record!, completionHandler: { (userID, error) in
            userName = (userID?.nameComponents?.givenName)! + " " + (userID?.nameComponents?.familyName)!
            print("CK User Name: " + userName)
        })
    }
}

func getJokesByCurrentUser(_ recordID: CKRecordID) {
    // The user is a reference, so we need to query against a reference
    let reference = CKReference(recordID: recordID, action: .none)
    let predicate = NSPredicate(format: "creatorUserRecordID == %@", reference)
```

CloudyWithAChanceOfErrors

```
fetched ID _23d865322080d4185ad95db121c80663
iCloudID: _23d865322080d4185ad95db121c80663
CK User Name: Andrew Binkowski
```

USER RECORDS



- You can get all friends who use the same app (discover users api)

DATA MODELING

DATA MODELING

- How should 1 to many relationships be handled?

Album

PhotoArray

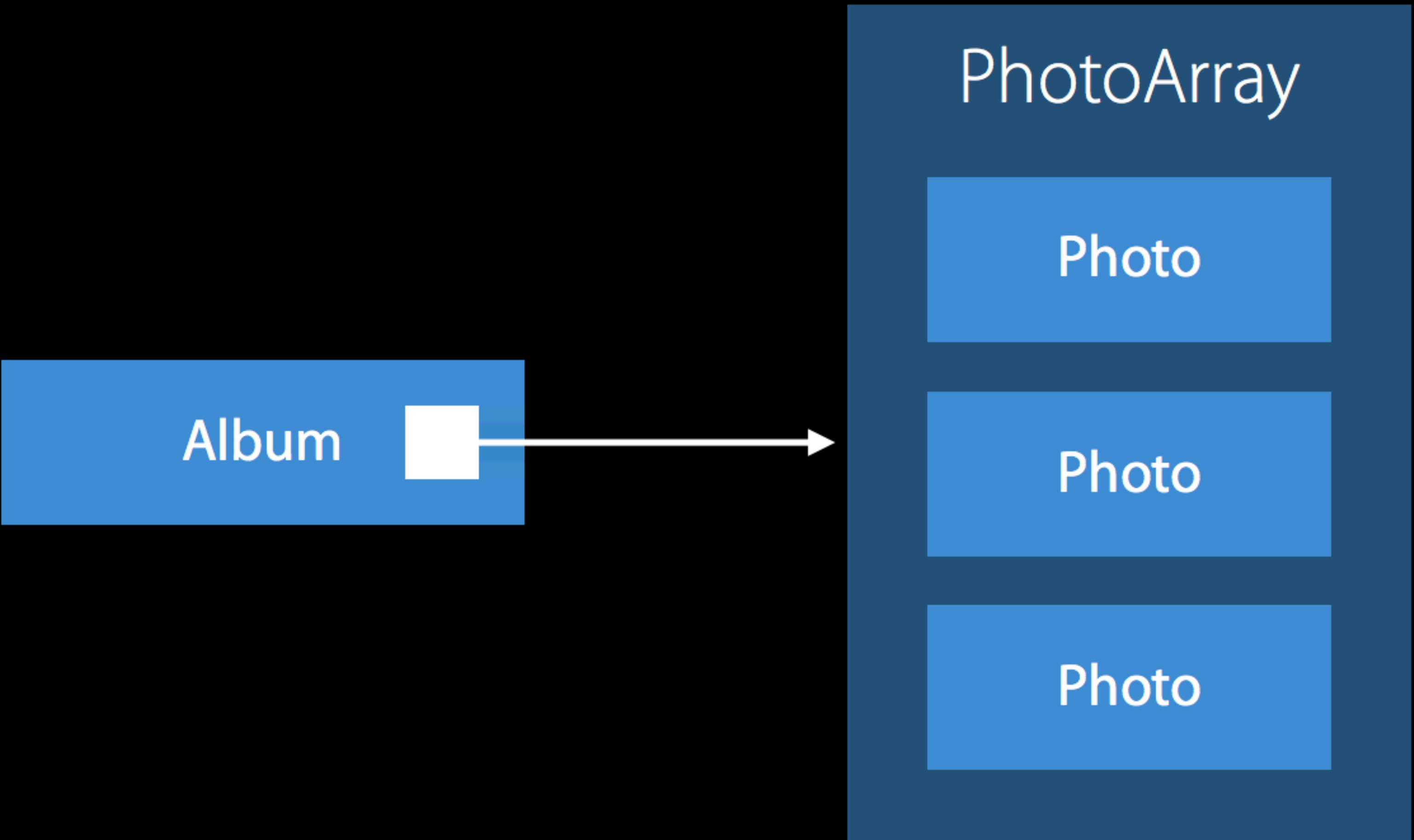
Photo

Photo

Photo

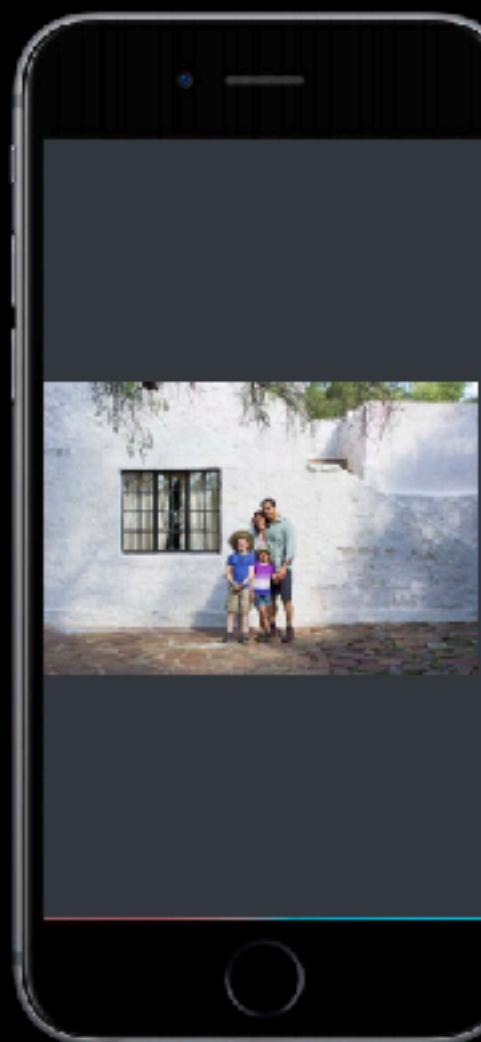
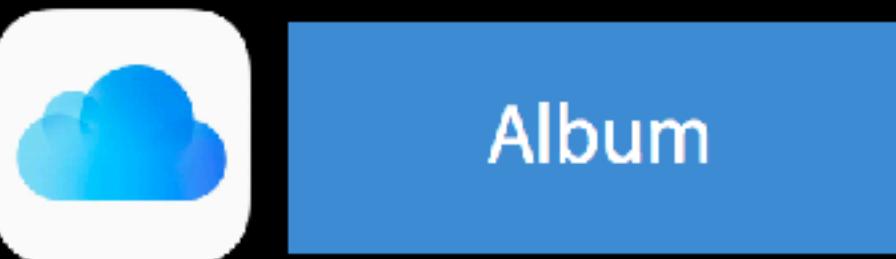
DATA MODELING

- Typical setup to model

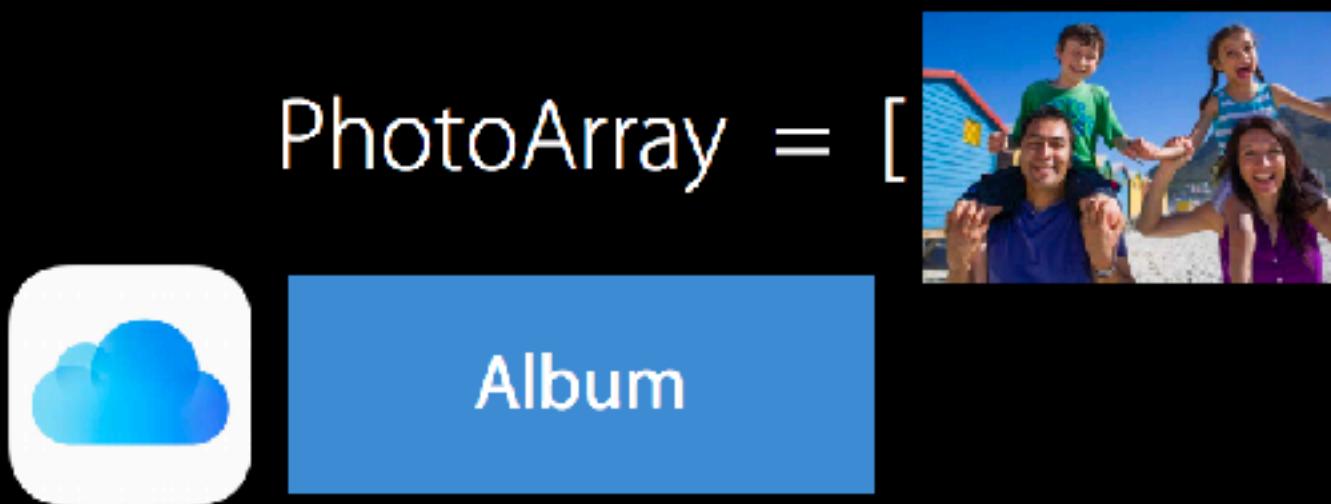


DATA MODELING

PhotoArray = []



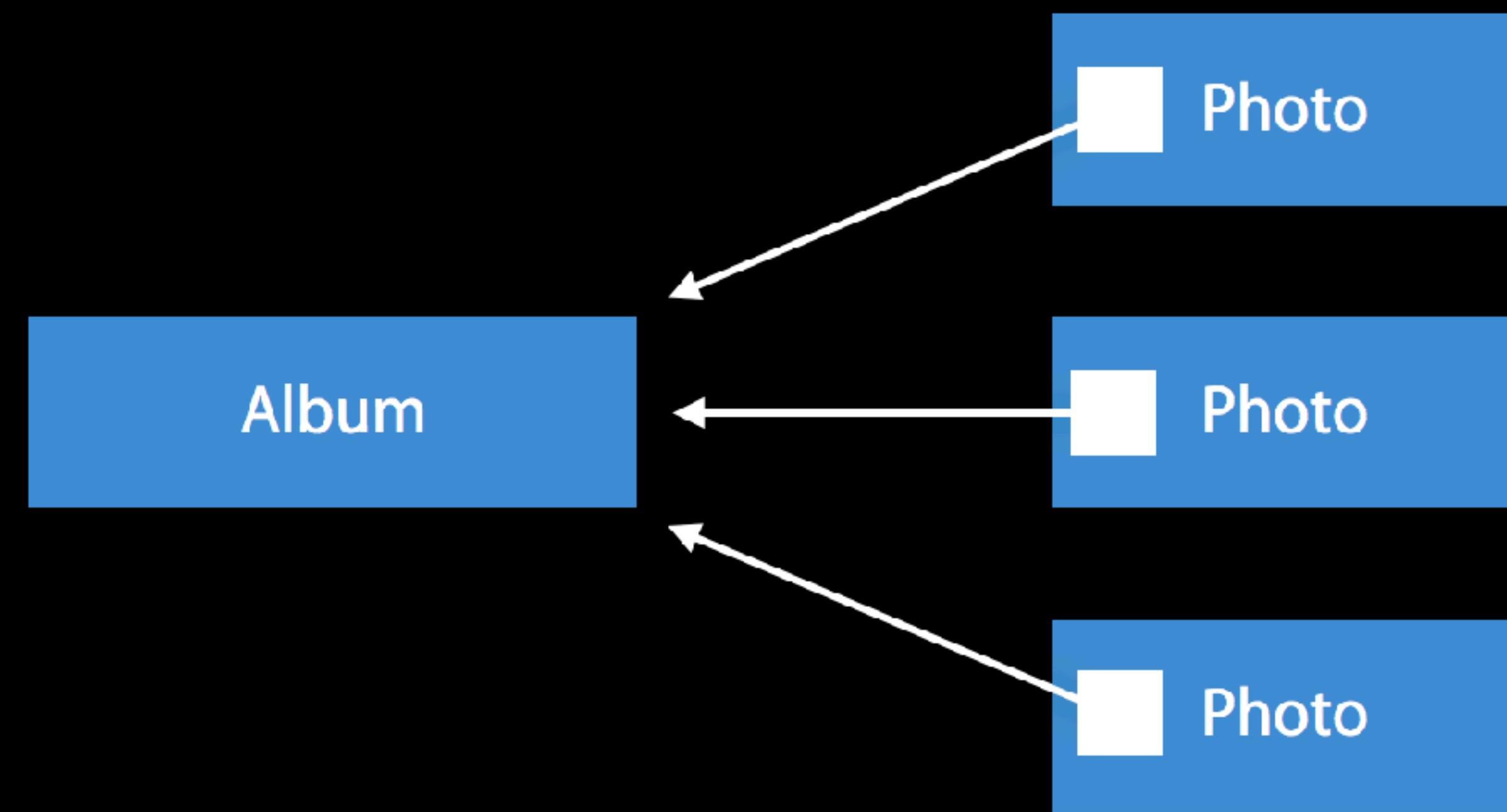
DATA MODELING



serverRecordChanged



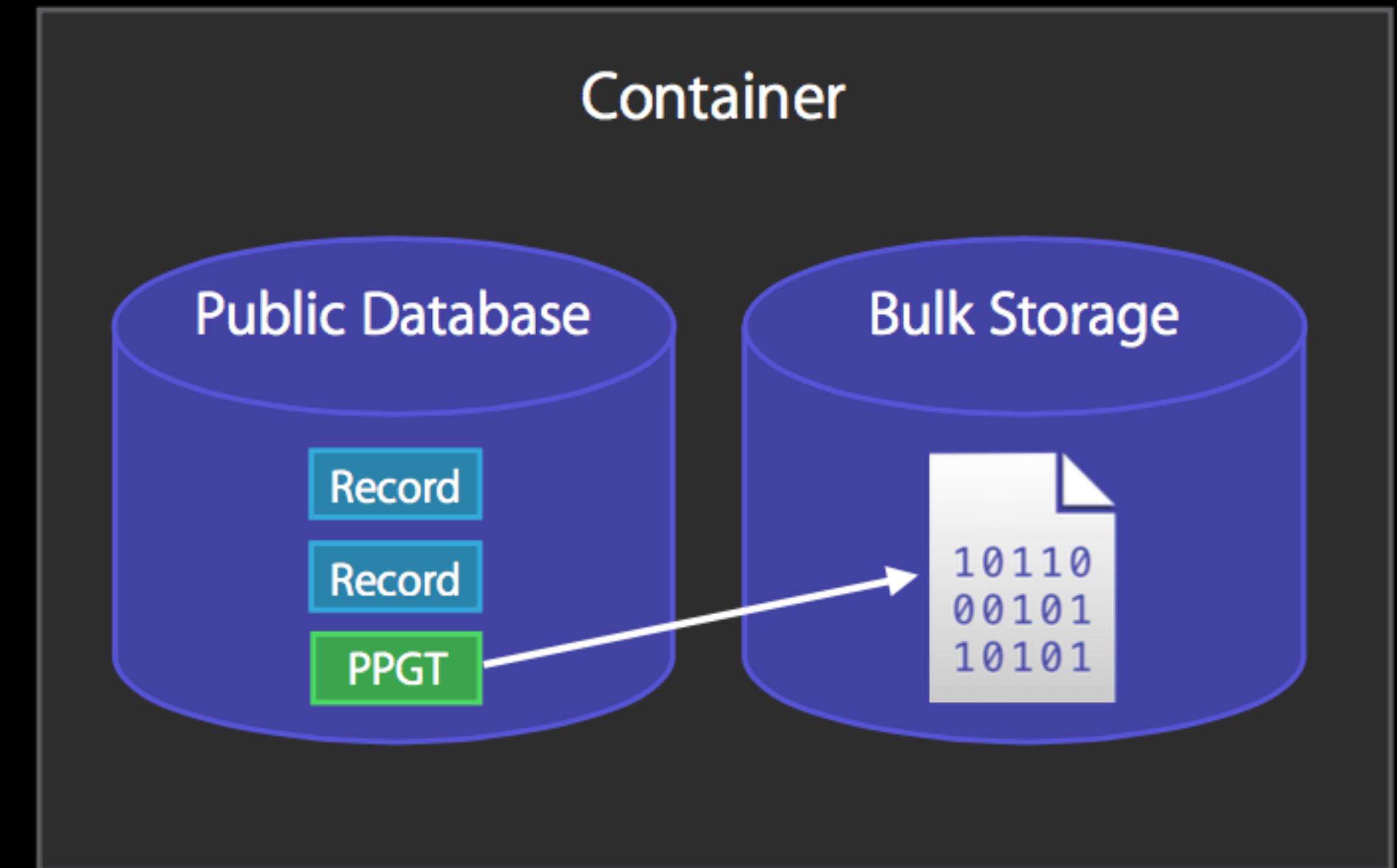
DATA MODELING



ASSETS

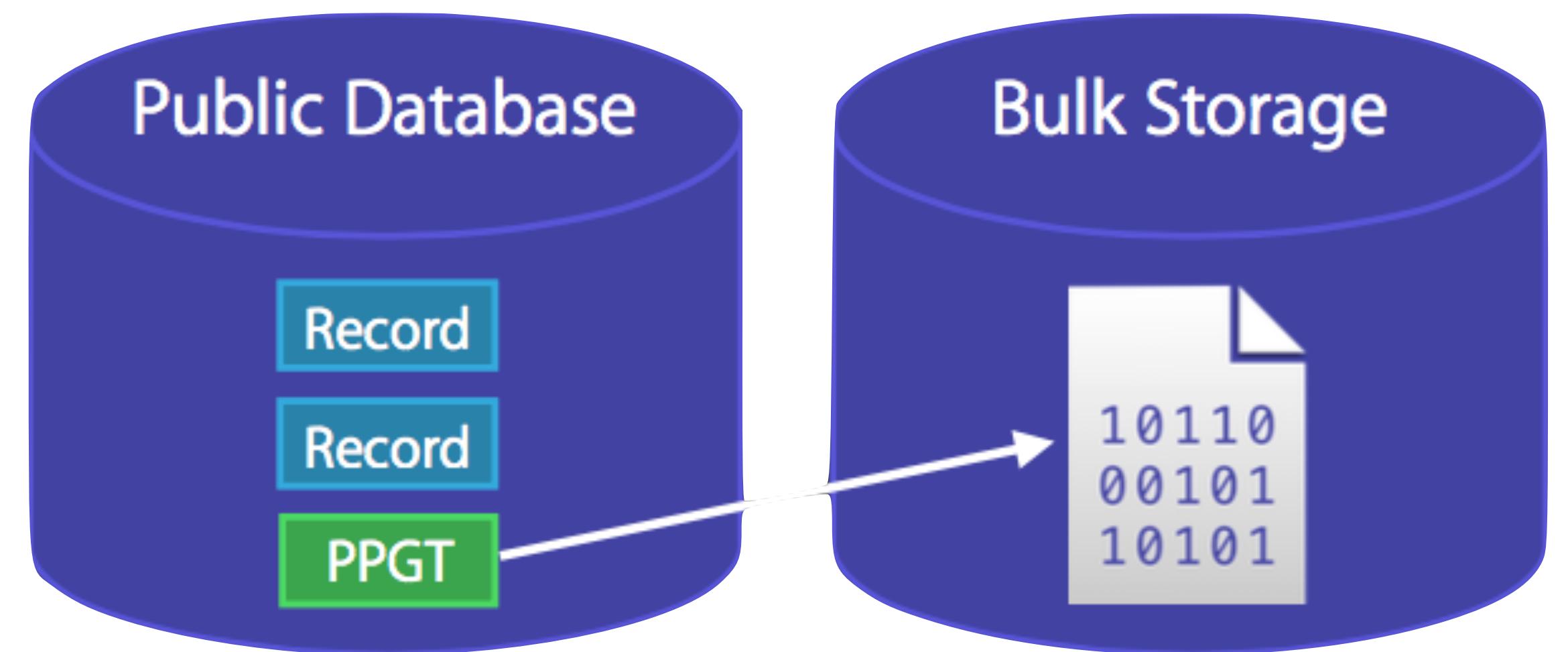
ASSETS

- CKAssets is bulk storage for CloudKit



ASSETS

- CKAsset
 - Large, unstructured data
 - Files on disk
 - Owned by a CKRecord
 - Garbage collected
 - Deletes with a record
 - Efficient uploads and downloads



ASSETS

Cloud > iCloud.cloud.uchicago.CloudyWithAChanceOfErros > Development Data ANDREW BINKOWSKI

ZONES	RECORDS	RECORD TYPES	INDEXES	SUBSCRIPTIONS	SUBSCRIPTION TYPES	SECURITY R	◀ ▶	EDITING RECORD	X
LOAD RECORDS FROM:	Public Database	Record Name	Record T...	Fields	Ch. T...	Created	Modified		
	_defaultZone	977D8136-0A78-430E-93C2-513B7B0674ED	joke	question Why did the student throw a clock out the window?	j9pri30j	Tue Nov 07 2017 09:20:06 GM	Tue Nov 07 2017 09:20:06 GM	X	
USING:				response She wanted to see time fly.		T-0600 (CST)	T-0600 (CST)		
				rating_positive 0					
				rating_negative 0					
QUERY FOR RECORDS OF TYPE:	joke	be9916ef-e31...	joke	question	j29q...	Wed May 03 ...	Wed May 03 ...		
Filter by:	Add filters...								
Sort by:	Add sorts...								
Query Records									
Query for records of the type "joke" that are in the "_defaultZone" zone of the "public" database.									
ASSETS									
There are no records referenced by this record.									
4 of 6									
audioFile ASSET									
Drag file here to upload.									
Choose File									

ASSETS

```
do {  
    let data = UIImagePNGRepresentation(myImage)!  
    try data.writeToURL(tempURL, options: NSDataWritingOptions.AtomicWrite)  
    let asset = CKAsset(fileURL: tempURL)  
    record["myImageKey"] = asset  
}  
catch {  
    print("Error writing data", error)  
}
```

NEED TO CONVERT FILE TO A
CKASSET AND THEN ASSOCIATE
WITH A RECORD

ASSETS

```
if let asset = record["myImageKey"] as? CKAsset,  
    data = NSData(contentsOfURL: asset.fileURL),  
    image = UIImage(data: data)  
{  
    // Do something with the image  
}
```

RETRIEVE AN IMAGE ASSOCIATED WITH A RECORD;
RETURNED A FILE THAT NEEDS TO BE ACTED ON IMMEDIATELY

BREAK TIME



QUERIES

QUERIES

- Big Data, Tiny Phone
 - Keep your large data in the cloud
 - Client views slice of that data
 - Client view can change
 - Clients use queries to focus their viewpoint



CLOUDKIT API

- Two ways to interact with CloudKit
 - Convenience API for single record interactions
- NSOperation based API
 - Customizable and fully featured

QUERIES

```
publicDB.save(record) { (record, error) in
    if let error = error {
        print("Error: \(error.localizedDescription)")
        return
    }
    print("Saved record: \(record.debugDescription)")
}
```

- Records are always live when they return
- Saving modified records directly

QUERIES

- CKQuery
 - RecordType
 - NSPredicate
 - CloudKit supports a subset of NSPredicate
 - NSSortDescriptors

Operation	Supported operators
Basic comparisons	=, ==
Boolean value predicates	TRUE PREDICATE
Basic compound predicates	AND, &&
String comparisons	BEGINSWITH
Aggregate Operations	IN
Functions	distanceToLocation:fromLocation:

QUERIES

```
[NSPredicate predicateWithFormat:@"name = %@", partyName];
```

```
[NSPredicate predicateWithFormat:@"%@%K = %@", dynamicKey, value];
```

```
[NSPredicate predicateWithFormat:@"start > %@", [NSDate date]];
```

```
CLLocation *location = [[CLLocation alloc] initWithLatitude:37.783 longitude:-122.404];
[NSPredicate predicateWithFormat:@"distanceToLocation:fromLocation:(Location, %@) < 100",
location];
```

```
[NSPredicate predicateWithFormat:@"ALL tokenize(@%, 'Cdl') IN allTokens",
@\"after session\"];
```

```
[NSPredicate predicateWithFormat:@"name = %@ AND startDate > %@", partyName, [NSDate date]];
```

QUERIES

CONNIVENCE API

```
/// Get all jokes in the public database
open func getJokes() {
    let predicate = NSPredicate(format: "TRUEPREDICATE")

    let query = CKQuery(recordType: "joke", predicate: predicate)
    publicDB.perform(query, inZoneWith: nil) { (records, error) -> Void in
        if let error = error {
            print("Error: \(String(describing: error.localizedDescription))")
            return
        }
        for record in records! {
            print("😂: \(record["question"] as! String)")
        }
    }
}
```

DON'T FORGOT TO HANDLE
ERRORS

QUERIES

```
/// Get all jokes by a user
/// - parameter recordID: The `CKRecordID` of the current user
open func getJokesByCurrentUser(_ recordID: CKRecordID) {

    // The user is a reference, so we need to query against a reference
    let reference = CKReference(recordID: recordID, action: .none)

    let predicate = NSPredicate(format: "creatorUserRecordID == %@", reference)

    let query = CKQuery(recordType: "joke", predicate: predicate)
    publicDB.perform(query, inZoneWith: nil) { (records, error) -> Void in
        if let error = error {
            print("Error: \(String(describing: error.localizedDescription))")
            return
        }
        for record in records! {
            print(record["question"] as! String)
        }
    }
}
```

QUERIES

```
/// Get all jokes by a user
/// - parameter recordID: The `CKRecordID` of the current user
open func getJokesByCurrentUser(_ recordID: CKRecordID) {

    // The user is a reference, so we need to query against a reference
    let reference = CKReference(recordID: recordID, action: .none)

    let predicate = NSPredicate(format: "creatorUserRecordID == %@", reference)

    let query = CKQuery(recordType: "joke", predicate: predicate)
    publicDB.perform(query, inZoneWith: nil) { (records, error) -> Void in
        if let error = error {
            print("Error: \(String(describing: error.localizedDescription))")
            return
        }
        for record in records! {
            print(record["question"] as! String)
        }
    }
}
```

NOTE WE HAVE
TO CREATE A
REFERENCE
FROM THE
RECORD

QUERIES

- Potentially have a problem with this query
- Async retrieval of user record required to make query
 - When will the value return?
 - Which order?

```
/// Get all jokes by a user
/// - parameter recordID: The `CKRecordID` of the user
open func getJokesByCurrentUser(_ recordID: CKRecordID) {
    // The user is a reference, so we need to fetch it
    let reference = CKReference(recordID: recordID,
                                 action: .read)
    let predicate = NSPredicate(format: "jokeAuthor == %@", reference)
    let query = CKQuery(recordType: "joke",
                        predicate: predicate)
    publicDB.perform(query, inZoneWith: nil) { [weak self] (records, error) in
        if let error = error {
            print("Error: \(String(describing: error))")
            return
        }
        for record in records! {
            print(record["question"] as! String)
        }
    }
}
```

QUERIES

- NSOperation based query
 - More control of responses
 - Results cursor
 - Guarantee the order of operations

Class

CKQueryOperation

A CKQueryOperation object is a concrete operation that you can use to execute queries against a database. A query operation takes the query parameters you provide and applies those parameters to the specified database and zone, delivering any matching records asynchronously to the blocks that you provide.

Overview

To perform a new search:

1. Initialize a CKQueryOperation object with a [CKQuery](#) object containing the search criteria and sorting information for the records you want.
2. Assign a block to the [queryCompletionBlock](#) property so that you can process the results and execute the operation.

If the search yields many records, the operation object may deliver a portion of the total results to your blocks.

QUERIES

- Operation query with cursor

```
/// Use the operation API to make a query and use cursors
/// to control the flow of data
/// - parameter query: A `CKQuery?`, most likely represents the initial query
open func getJokesWithOperation(query: CKQuery?, cursor: CKQueryCursor?) {
    var queryOperation: CKQueryOperation!

    if query != nil {
        let predicate = NSPredicate(value: true)
        let query = CKQuery(recordType: "joke", predicate: predicate)
        queryOperation = CKQueryOperation(query: query)
    } else if let cursor = cursor {
        print("== Cursor =====")
        queryOperation = CKQueryOperation(cursor: cursor)
    }

    // Query parameters
//queryOperation.desiredKeys = ["", "", ""]
queryOperation.queuePriority = .veryHigh
queryOperation.resultsLimit = 2
queryOperation.qualityOfService = .userInteractive

    // This gets called each time per record
queryOperation.recordFetchedBlock = {
    (record: CKRecord!) -> Void in
    if record != nil {
        print("😂 operation: \(record["question"] as! String)")
    }
}

    // This is called after all records are retrieved and iterated
// on
queryOperation.queryCompletionBlock = { cursor, error in
    if (error != nil) {
        print("Error:\(String(describing: error))")
        return
    }

    if let cursor = cursor {
        print("There is more data to fetch")
        self.getJokesWithOperation(query: nil, cursor: cursor)

        print("Done with operation...")
        //OperationQueue.main.addOperation() {
        // Do anything else with the record after downloaded that
        // needs to be on the main thread
        //}
    }
}
self.publicDB.add(queryOperation)
}
```

QUERIES

```
/// Use the operation API to make a query and use cursors
/// to control the follow of data
/// - parameter query: A `CKQuery?`, most likely represents the initial query
open func getJokesWithOperation(query: CKQuery?, cursor: CKQueryCursor?) {
    var queryOperation: CKQueryOperation!

    if query != nil {
        let predicate = NSPredicate(value: true)
        let query = CKQuery(recordType: "joke", predicate: predicate)
        queryOperation = CKQueryOperation(query: query)
    } else if let cursor = cursor {
        print("== Cursor =====")
        queryOperation = CKQueryOperation(cursor: cursor)
    }

    // Query parameters
    //queryOperation.desiredKeys = ["", "", ""]
    queryOperation.queuePriority = .veryHigh
    queryOperation.resultsLimit = ?
```

QUERIES

```
// Query parameters
//queryOperation.desiredKeys = ["", "", ""]
queryOperation.queuePriority = .veryHigh
queryOperation.resultsLimit = 2
queryOperation.qualityOfService = .userInteractive

// This gets called each time per record
queryOperation.recordFetchedBlock = {
    (record: CKRecord!) -> Void in
    if record != nil {
        print("😂 operation: \(record["question"] as! String)")
    }
}

// This is called after all records are retrieved and iterated
// on
queryOperation.queryCompletionBlock = { cursor, error in
    if (error != nil) {
        print("Error: \(String(describing: error))")
    }
}
```

QUERIES

```
// Query parameters
//queryOperation.desiredKeys = ["", "", ""]
queryOperation.queuePriority = .veryHigh
queryOperation.resultsLimit = 2
queryOperation.qualityOfService = .userInteractive

// This gets called each time per record
queryOperation.recordFetchedBlock = {
    (record: CKRecord!) -> Void in
    if record != nil {
        print("😂 operation: \(record["question"] as! String)")
    }
}

// This is called after all records are retrieved and iterated
// on
queryOperation.queryCompletionBlock = { cursor, error in
    if (error != nil) {
        print("Error: \(String(describing: error))")
    }
}
```

QUERIES

```
// This is called after all records are retrieved and iterated
// on
queryOperation.queryCompletionBlock = { cursor, error in
    if (error != nil) {
        print("Error:\(String(describing: error))")
        return
    }

    if let cursor = cursor {
        print("There is more data to fetch")
        self.getJokesWithOperation(query: nil, cursor: cursor)

        print("Done with opeartion...")
        //OperationQueue.main.addOperation() {
        // Do anything else with the record after downloaded that
        // needs to be on the main thread
        //}
    }
}
```

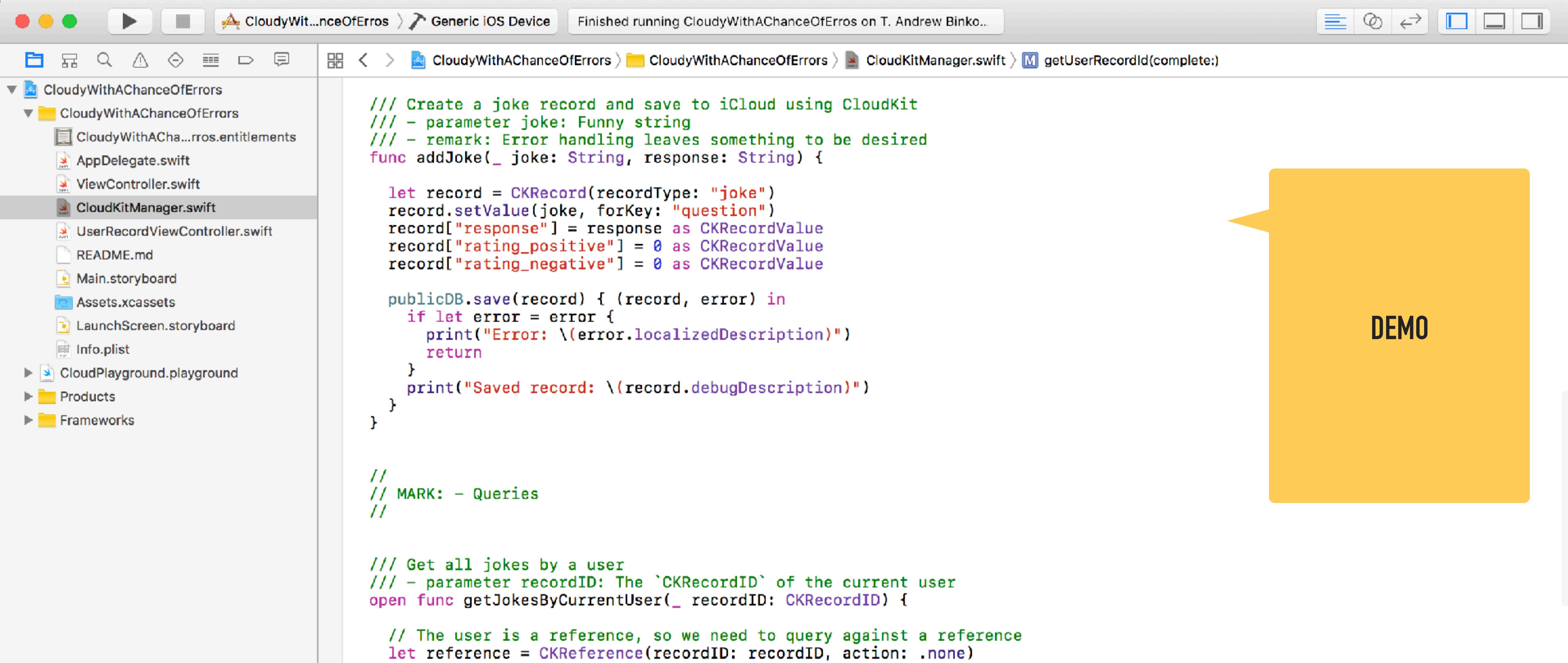
QUERIES

```
if let cursor = cursor {
    print("There is more data to fetch")
    self.getJokesWithOperation(query: nil, cursor: cursor)

    print("Done with opeartion...")
    //OperationQueue.main.addOperation() {
    // Do anything else with the record after downloaded that
    // needs to be on the main thread
    //}
}
}

self.publicDB.add(queryOperation)
}
```

QUERIES



CloudyWithAChanceOfErrors > Generic iOS Device Finished running CloudyWithAChanceOfErrors on T. Andrew Binko...

CloudyWithAChanceOfErrors > CloudyWithAChanceOfErrors > CloudKitManager.swift > M getUserRecordId(complete:)

```
/// Create a joke record and save to iCloud using CloudKit
/// - parameter joke: Funny string
/// - remark: Error handling leaves something to be desired
func addJoke(_ joke: String, response: String) {

    let record = CKRecord(recordType: "joke")
    record.setValue(joke, forKey: "question")
    record["response"] = response as CKRecordValue
    record["rating_positive"] = 0 as CKRecordValue
    record["rating_negative"] = 0 as CKRecordValue

    publicDB.save(record) { (record, error) in
        if let error = error {
            print("Error: \(error.localizedDescription)")
            return
        }
        print("Saved record: \(record.debugDescription)")
    }
}

// MARK: - Queries

/// Get all jokes by a user
/// - parameter recordID: The `CKRecordID` of the current user
open func getJokesByCurrentUser(_ recordID: CKRecordID) {

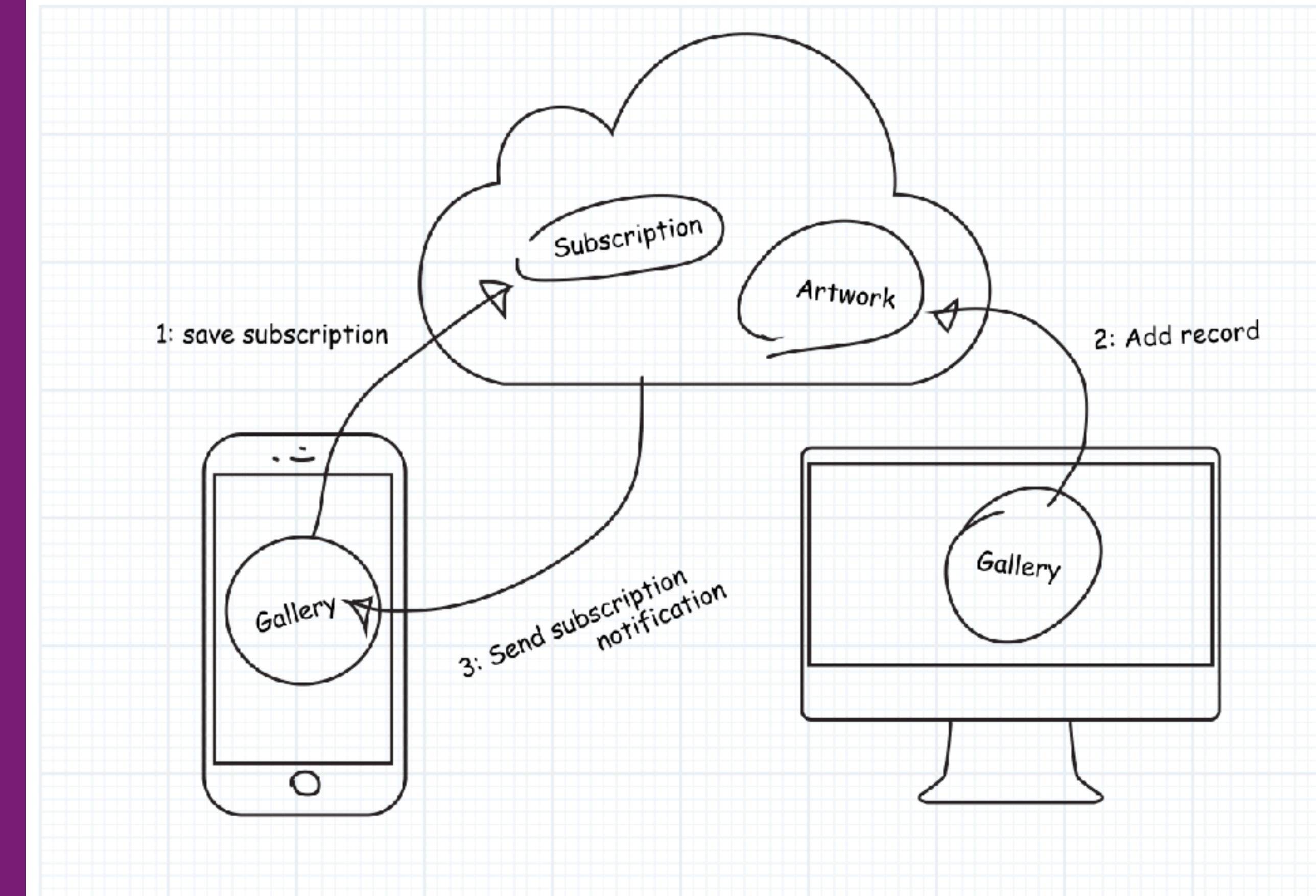
    // The user is a reference, so we need to query against a reference
    let reference = CKReference(recordID: recordID, action: .none)
```

DEMO

SUBSCRIPTIONS

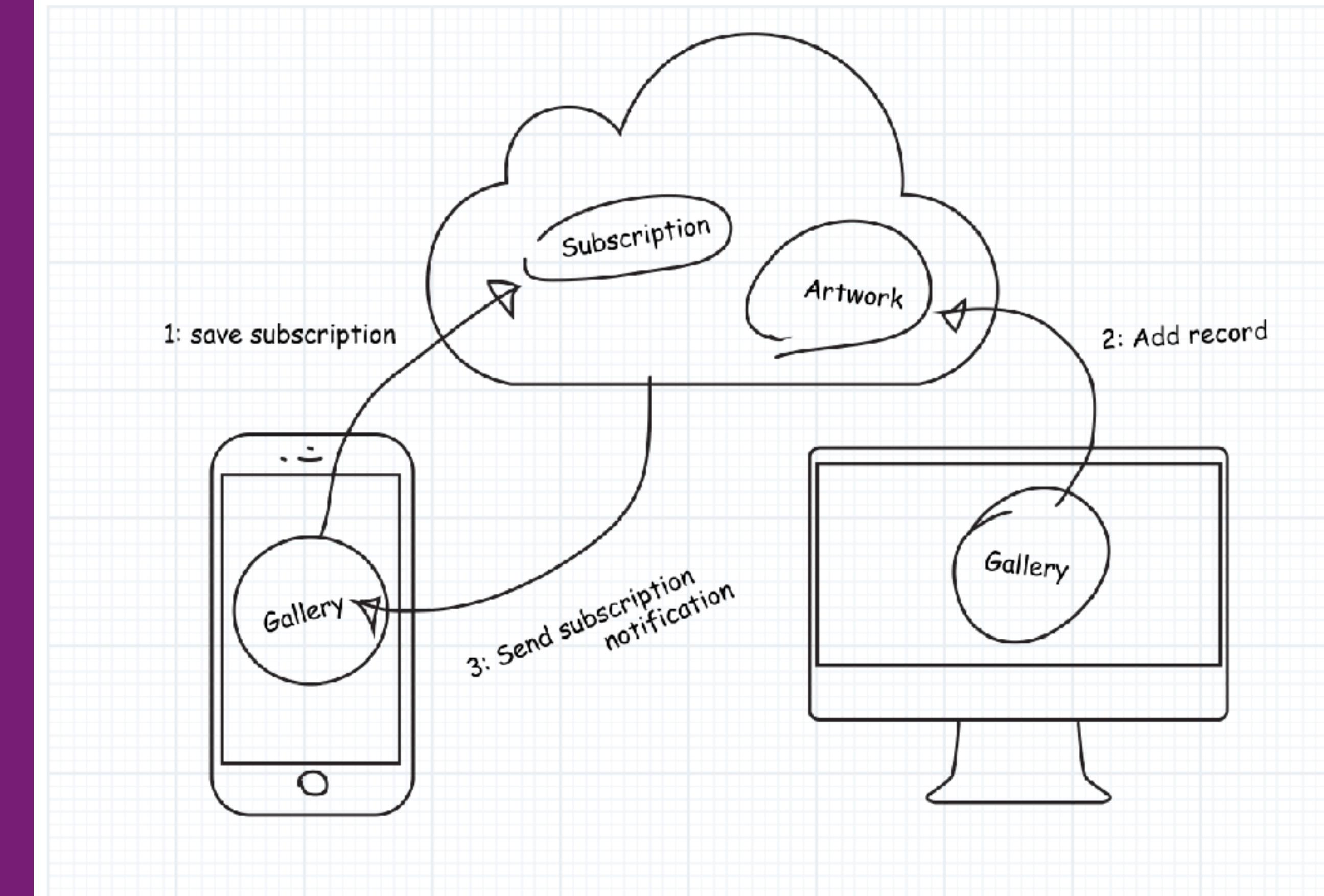
SUBSCRIPTIONS

- Queries are polls
- Great for slicing through large server data
- Bad for large, mostly static data set
 - Battery life
 - Networking traffic
 - User experience



SUBSCRIPTIONS

- The server runs your query
 - In the background
 - After every record save
 - Push results to other other devices



SUBSCRIPTIONS

Apple Inc.

iCloud.cloud.uchicago.CloudyWithAChanceOfErros > Development Data ANDREW BINKOWSKI

ZONES	RECORDS	RECORD TYPES	INDEXES	SUBSCRIPTIONS	SUBSCRIPTION TYPES	SECURITY ROLES	
CHOOSE A SUBSCRIPTION TYPE:							
Alert f96474eb92a95a9802434f289f03d839		<p>Signature f96474eb92a95a9802434f289f03d839</p> <p>Record Type Alert</p> <p>Triggers INSERT</p> <p>Filters None</p>					
Daily b995870e8254b265cce02bfd2c090ec4							
joke 686bf79f17b47aef6f7ee1a7fb7bc82							
joke 7c401b3dddc25d4979ef61e55040e5da							
Subscription types are automatically created when your app creates a Query Subscription.							

SUBSCRIPTIONS

- CKQuerySubscription
 - RecordType
 - NSPredicate
 - Push
- Push delivered via Apple Push Notification Service (APNS) augmented payload
 - JSON

CloudKit Dashboard showing a subscription configuration:

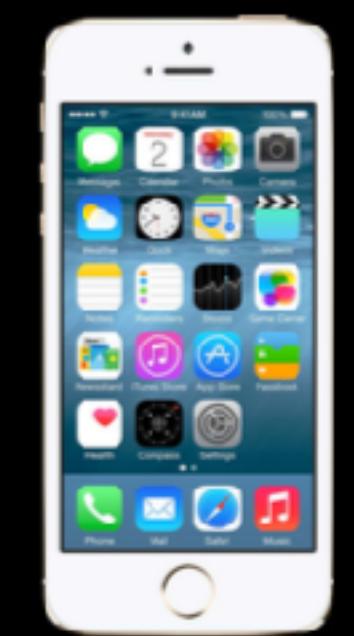
ZONES	RECORDS	RECORD TYPES	INDEXES	SUBSCRIPTIONS	SUBS
CHOOSE A SUBSCRIPTION TYPE:					
Alert f96474eb92a95a9802434f289f03d839					
Daily b995870e8254b265cce02bfd2c090ec4					
joke 686bfb79f17b47aef6f7ee1a7fb7bc82					
joke 7c401b3dddc25d4979ef61e55040e5da					
Subscription types are automatically created when your app creates a Query Subscription.					
				Signature	f96474eb92a95a9802434f289f03d839
				Record Type	Alert
				Triggers	INSERT
				Filters	None

A yellow callout bubble points from the bottom right towards the "joke" subscription type, containing the text:

CHEAPEST, EASIEST WAY TO ADD PUSH NOTIFICATIONS TO AN APP

SUBSCRIPTIONS

- Subscription in action

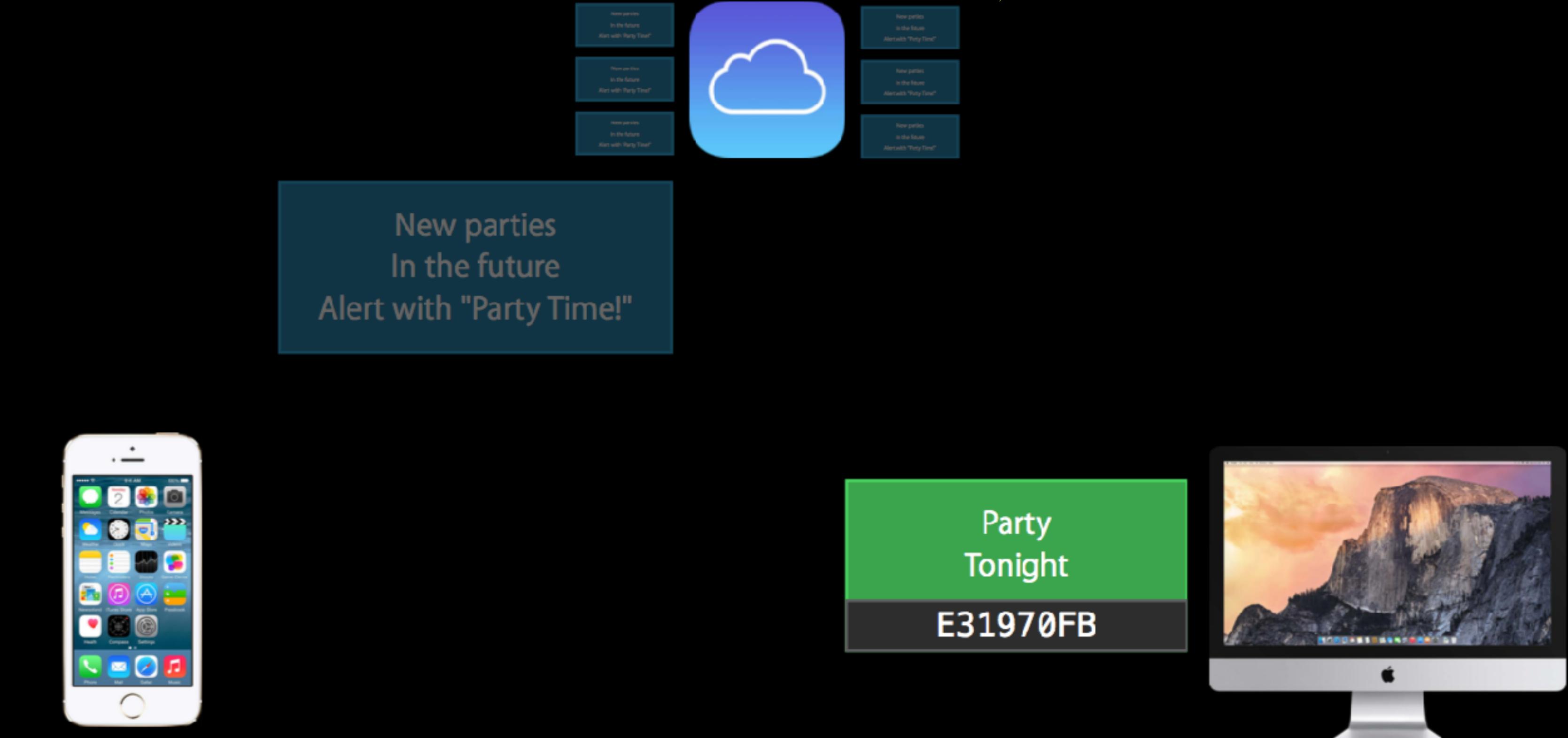


APPLE PARTY NOTIFICATION EXAMPLE

APPLE PARTY NOTIFICATION EXAMPLE

SUBSCRIPTIONS

- Subscription in action



APPLE PARTY NOTIFICATION EXAMPLE

SUBSCRIPTIONS

- Subscription in action



New parties
In the future
Alert with "Party Time!"



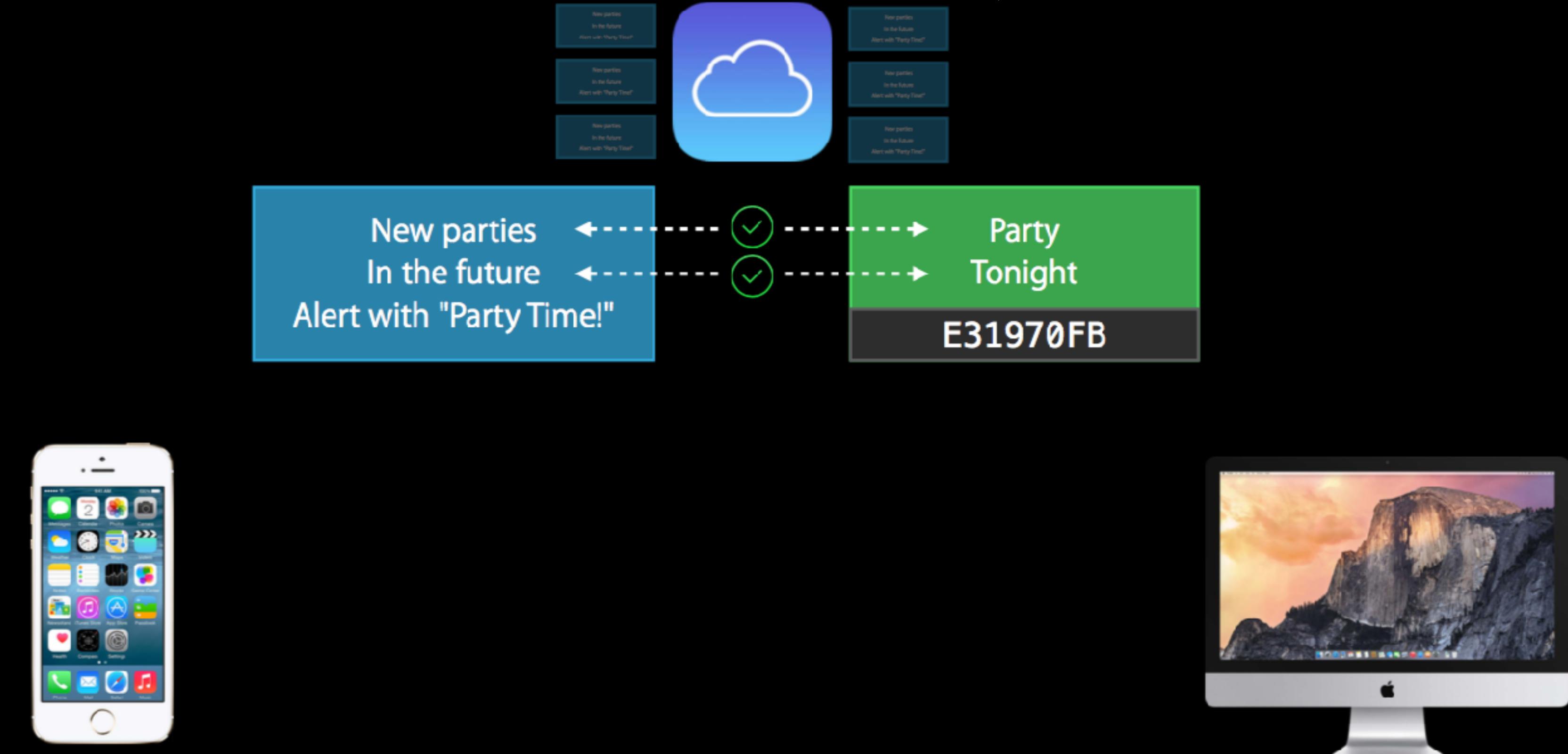
Party
Tonight
E31970FB



APPLE PARTY NOTIFICATION EXAMPLE

SUBSCRIPTIONS

- Subscription in action



APPLE PARTY NOTIFICATION EXAMPLE

SUBSCRIPTIONS

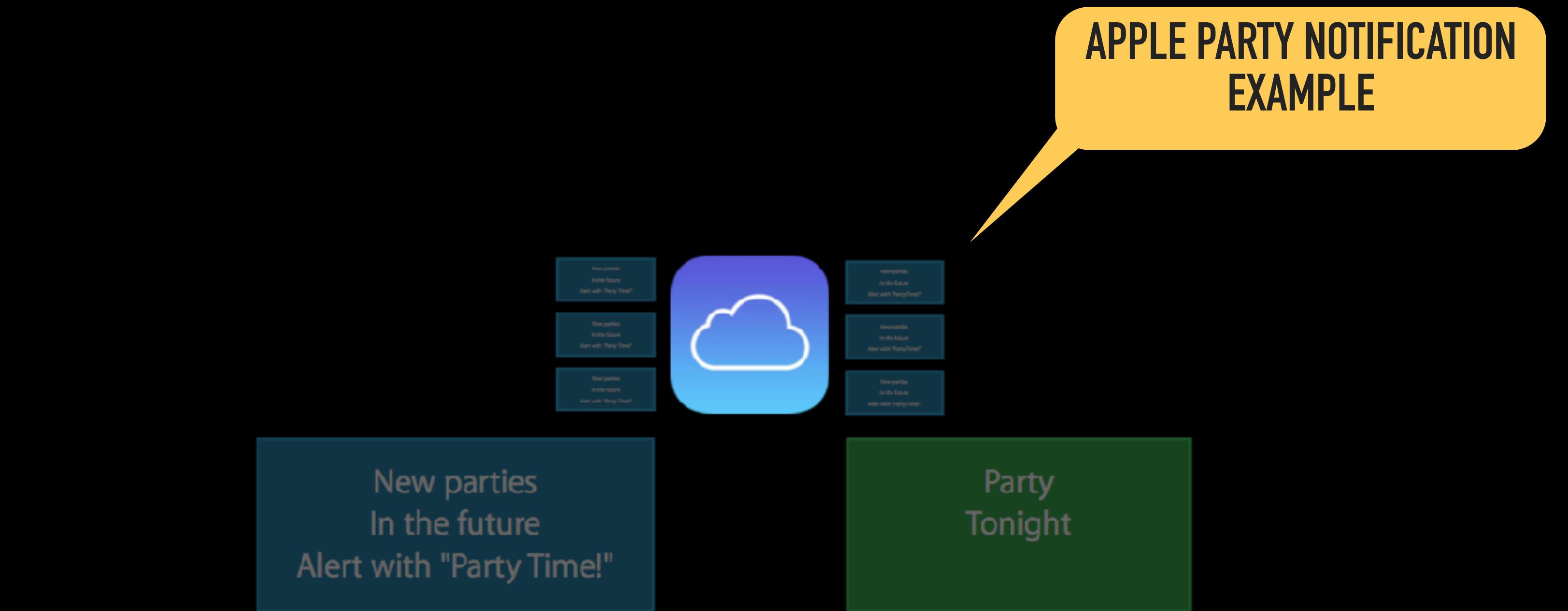
- Subscription in action



APPLE PARTY NOTIFICATION EXAMPLE

SUBSCRIPTIONS

- Subscription in action



SUBSCRIPTIONS

- Subscriptions are user based
- Require a unique id

```
func registerSubscriptionsWithIdentifier(_ identifier: String) {  
  
    let uuid: UUID = UIDevice().identifierForVendor!  
    let identifier = "\u{uuid}-creation"  
  
    // Create the notification that will be delivered  
    let notificationInfo = CKNotificationInfo()  
    notificationInfo.alertBody = "A new joke was added."  
    notificationInfo.shouldBadge = true  
  
    let subscription = CKQuerySubscription(recordType: "joke",  
                                            predicate: NSPredicate(value: true),  
                                            subscriptionID: identifier,  
                                            options: [.firesOnRecordCreation])  
    subscription.notificationInfo = notificationInfo  
    publicDB.save(subscription, completionHandler: {returnRecord, error in  
        if let err = error {  
            print("subscription failed \(err.localizedDescription)")  
        } else {  
            print("subscription set up")  
        }  
    })  
}
```

SUBSCRIPTIONS

- Create a generic notification
- Some tricks to make your notifications seem more dynamic
 - Alert view when received

```
func registerSubscriptionsWithIdentifier(_ identifier: String) {  
  
    let uuid: UUID = UIDevice().identifierForVendor!  
    let identifier = "\u{uuid}-creation"  
  
    // Create the notification that will be delivered  
    let notificationInfo = CKNotificationInfo()  
    notificationInfo.alertBody = "A new joke was added."  
    notificationInfo.shouldBadge = true  
  
    let subscription = CKQuerySubscription(recordType: "joke",  
                                             predicate: NSPredicate(value: true),  
                                             subscriptionID: identifier,  
                                             options: [.firesOnRecordCreation])  
    subscription.notificationInfo = notificationInfo  
    publicDB.save(subscription, completionHandler: {returnRecord, error in  
        if let err = error {  
            print("subscription failed \(err.localizedDescription)")  
        } else {  
            print("subscription set up")  
        }  
    })  
}
```

SUBSCRIPTIONS

- Create a subscription

```
func registerSubscriptionsWithIdentifier(_ identifier: String) {  
  
    let uuid: UUID = UIDevice().identifierForVendor!  
    let identifier = "\u{uuid}-creation"  
  
    // Create the notification that will be delivered  
    let notificationInfo = CKNotificationInfo()  
    notificationInfo.alertBody = "A new joke was added."  
    notificationInfo.shouldBadge = true  
  
    let subscription = CKQuerySubscription(recordType: "joke",  
                                             predicate: NSPredicate(value: true),  
                                             subscriptionID: identifier,  
                                             options: [.firesOnRecordCreation])  
    subscription.notificationInfo = notificationInfo  
    publicDB.save(subscription, completionHandler: {returnRecord, error in  
        if let err = error {  
            print("subscription failed \(err.localizedDescription)")  
        } else {  
            print("subscription set up")  
        }  
    })  
}
```

SUBSCRIPTIONS

- Save a subscription
 - Need to do some local work to remember state of subscription preferences

```
func registerSubscriptionsWithIdentifier(_ identifier: String) {  
  
    let uuid: UUID = UIDevice().identifierForVendor!  
    let identifier = "\u{uuid}-creation"  
  
    // Create the notification that will be delivered  
    let notificationInfo = CKNotificationInfo()  
    notificationInfo.alertBody = "A new joke was added."  
    notificationInfo.shouldBadge = true  
  
    let subscription = CKQuerySubscription(recordType: "joke",  
                                            predicate: NSPredicate(value: true),  
                                            subscriptionID: identifier,  
                                            options: [.firesOnRecordCreation])  
    subscription.notificationInfo = notificationInfo  
    publicDB.save(subscription, completionHandler: {returnRecord, error in  
        if let err = error {  
            print("subscription failed \(err.localizedDescription)")  
        } else {  
            print("subscription set up")  
        }  
    })  
}
```

SUBSCRIPTIONS

SHOW ALL SUBSCRIPTIONS

ANDREW BINKOWSKI ▾

ZONES	RECORDS	RECORD TYPES	INDEXES	SUBSCRIPTIONS	SUBSCRIPTION TYPES	SECURITY ROLES	
CHOOSE A SUBSCRIPTION TYPE:							
Alert f96474eb92a95a9802434f289f03d839							
Daily b995870e8254b265cce02bfd2c090ec4							
joke 686bfb79f17b47aef6f7ee1a7fb7bc82							
joke 7c401b3dddc25d4979ef61e55040e5da							
Subscription types are automatically created when your app creates a Query Subscription.							
Signature f96474eb92a95a9802434f289f03d839 Record Type Alert Triggers INSERT Filters None							
SEE DETAILS							
Delete							

SETUP YOUR APP TO
HANDLE NOTIFICATIONS

SUBSCRIPTIONS

- Required capabilities
 - Push notifications

CloudyWithAChanceOfErrors > Generic iOS Device Finished running CloudyWithAChanceOfErrors on T. Andrew Binko..

CloudyWithAChanceOfErrors

General Capabilities Resource Tags Info Build Settings Build Phases Build Rules

PROJECT iCloud ON

TARGETS CloudyWithAChanceOf... Push Notifications ON

Steps: ✓ Add the Push Notifications feature to your App ID.
✓ Add the Push Notifications entitlement to your entitlements file

Game Center OFF

Wallet OFF

Siri OFF

Apple Pay OFF

In-App Purchase OFF

Maps OFF

Personal VPN OFF

Network Extensions OFF

Background Modes ON

Modes:

- Audio, AirPlay, and Picture in Picture
- Location updates
- Voice over IP
- Newsstand downloads
- External accessory communication
- Uses Bluetooth LE accessories
- Acts as a Bluetooth LE accessory
- Background fetch
- Remote notifications

Steps: ✓ Add the Required Background Modes key to your info.plist file

SUBSCRIPTIONS

- Required capabilities
 - Remote notification
 - Background fetch

▼ Background Modes

- Modes:
- Audio, AirPlay, and Picture in Picture
 - Location updates
 - Voice over IP
 - Newsstand downloads
 - External accessory communication
 - Uses Bluetooth LE accessories
 - Acts as a Bluetooth LE accessory
 - Background fetch
 - Remote notifications

Steps:  Add the Required Background Modes key to your info.plist file

► Inter-App Audio

► Keychain Sharing

► Associated Domains

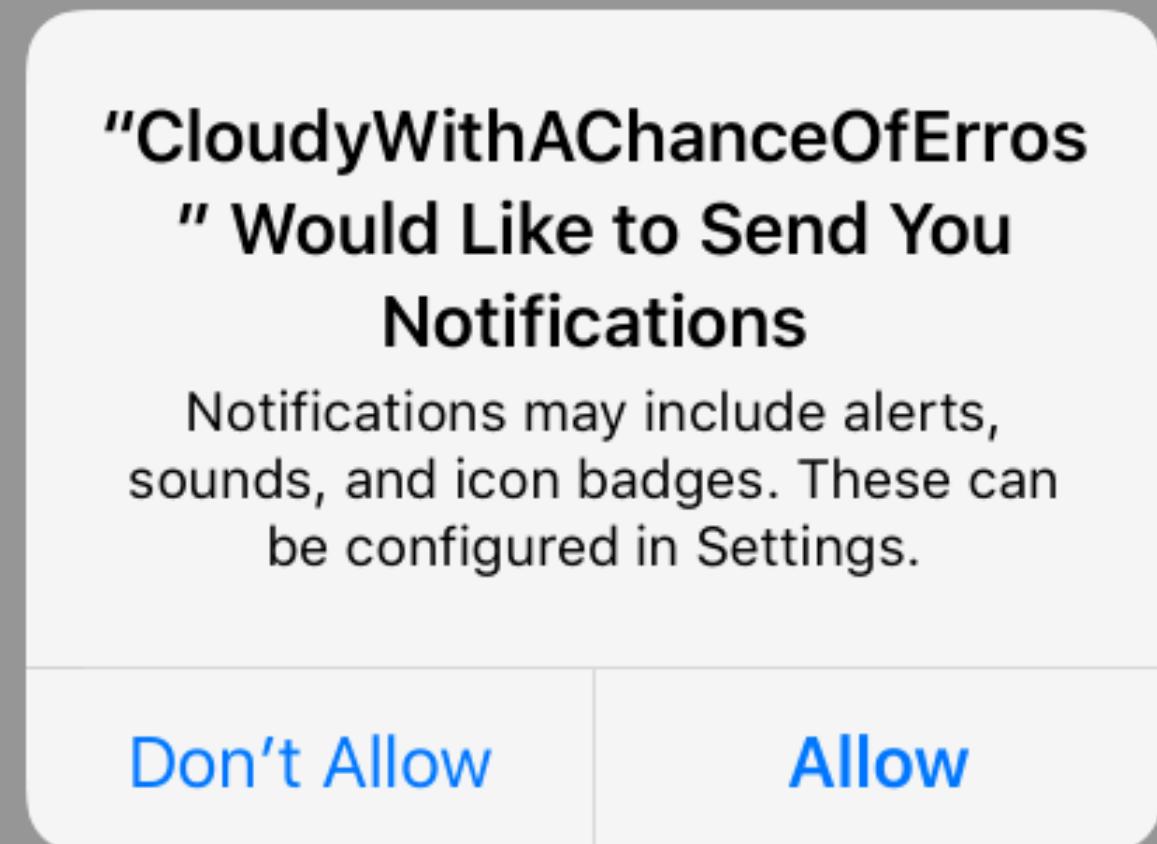
SUBSCRIPTIONS

REQUEST PERMISSIONS TO RECEIVE
NOTIFICATIONS

```
func application(_ application: UIApplication,  
                 didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {  
  
    // Set the notification delegate  
    UNUserNotificationCenter.current().requestAuthorization(options: [.alert, .sound, .badge]) { granted, error in  
        if let error = error {  
            print("Error: \(error.localizedDescription)")  
        } else {  
            application.registerForRemoteNotifications()  
        }  
    }  
    UNUserNotificationCenter.current().delegate = self  
  
    // Register subscriptions  
    CloudKitManager.sharedInstance.registerSubscriptionsWithIdentifier("id2")  
    CloudKitManager.sharedInstance.registerSilentSubscriptionsWithIdentifier("id3")  
    //configureUserNotificationsCenter()  
  
    if let notification = launchOptions?[UIApplicationLaunchOptionsKey.remoteNotification] as? [String: AnyObject] {  
        // 2  
        let aps = notification["aps"] as! [String: AnyObject]  
        print("APS: \(aps)")  
        // 3  
        //((window?.rootViewController as? UITabBarController)?.selectedIndex = 1  
    }  
    return true  
}
```

SUBSCRIPTIONS

- Users will have to grant permissions for notifications that you see
- You can send silent notifications without permission



SUBSCRIPTIONS

```
func application(_ application: UIApplication,  
                 didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {  
  
    // Set the notification delegate  
    UNUserNotificationCenter.current().requestAuthorization(options: [.ale  
        if let error = error {  
            print("Error: \(error.localizedDescription)")  
        } else {  
            application.registerForRemoteNotifications()  
        }  
    }  
    UNUserNotificationCenter.current().delegate = self  
  
    // Register subscriptions  
    CloudKitManager.sharedInstance.registerSubscriptionsWithIdentifier("id2")  
    CloudKitManager.sharedInstance.registerSilentSubscriptionsWithIdentifier("id3")  
    //configureUserNotificationsCenter()  
  
    if let notification = launchOptions?[UIApplicationLaunchOptionsKey.remoteNotification] as? [String: AnyObject] {  
        // 2  
        let aps = notification["aps"] as! [String: AnyObject]  
        print("APS: \(aps)")  
        // 3  
        //((window?.rootViewController as? UITabBarController)?.selectedIndex = 1  
    }  
    return true  
}
```

REGISTER SUBSCRIPTIONS

error in

SUBSCRIPTIONS

- Register a subscription

```
func registerSubscriptionsWithIdentifier(_ identifier: String) {  
  
    let uuid: UUID = UIDevice().identifierForVendor!  
    let identifier = "\(uuid)-creation"  
  
    // Create the notification that will be delivered  
    let notificationInfo = CKNotificationInfo()  
    notificationInfo.alertBody = "A new joke was added."  
    notificationInfo.shouldBadge = true  
    notificationInfo.shouldSendContentAvailable = true  
  
  
    let subscription = CKQuerySubscription(recordType: "joke",  
                                            predicate: NSPredicate(value: true),  
                                            subscriptionID: identifier,  
                                            options: [.firesOnRecordCreation])  
    subscription.notificationInfo = notificationInfo  
    CKContainer.default().publicCloudDatabase.save(subscription, completionHandler: {returnRecord, error in  
        if let err = error {  
            print("subscription failed \(err.localizedDescription)")  
        } else {  
            print("subscription set up")  
        }  
    })  
}
```

SUBSCRIPTIONS

```
UNUserNotificationCenter.current().delegate = self

// Register subscriptions
CloudKitManager.sharedInstance.registerSubscriptionsWithIdentifier("id2")
CloudKitManager.sharedInstance.registerSilentSubscriptionsWithIdentifier("id3")
//configureUserNotificationsCenter()

if let notification = launchOptions?[UIApplicationLaunchOptionsKey.remoteNotification] as? [String: AnyObject] {
    let aps = notification["aps"] as! [String: AnyObject]
    print("APS: \(aps)")
}
return true
}
```

- Handle CloudKit notifications in two places (depending on the state of the app)
 - applicationWillFinishLaunching
 - didReceiveRemoteNotification

SUBSCRIPTIONS

```
func application(_ application: UIApplication, didReceiveRemoteNotification userInfo: [AnyHashable : Any],  
    fetchCompletionHandler completionHandler: @escaping (UIBackgroundFetchResult) -> Void) {  
    let aps = userInfo["aps"] as! [String: AnyObject]  
  
    if (aps["content-available"] as? NSString)?.integerValue == 1 {  
        // Pull data  
        completionHandler(.newData)  
    } else {  
        completionHandler(.newData)  
    }  
}
```

Description

Tells the app that a remote notification arrived that indicates there is data to be fetched.

Use this method to process incoming remote notifications for your app. Unlike the `application(_:didReceiveRemoteNotification:)` method, which is called only when your app is running in the foreground, the system calls this method when your app is running in the foreground or background. In addition, if you enabled the remote notifications background mode, the system launches your app (or wakes it from the suspended state) and puts it in the background state when a remote notification arrives. However, the system does not automatically launch your app if the user has force-quit it. In that situation, the user must relaunch your app or restart the device before the system attempts to launch your app automatically again.

SUBSCRIPTIONS

- Notification can be silent
 - Push only data
- Pay attention to state of application similar to other notifications

```
// AppDelegate.swift
// CloudyWithAChanceOfErrors
//
// Created by T. Andrew Binkowski on 5/2/17.
// Copyright © 2017 T. Andrew Binkowski. All rights reserved.
//

import UIKit
import UserNotifications
import CloudKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate, UNUserNotificationCenterDelegate {

    var window: UIWindow?

    func application(_ application: UIApplication,
                     didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey: Any]?) {
        // Set the notification delegate
        UNUserNotificationCenter.current().requestAuthorization(options: [.alert, .sound, .badge]) { granted, error in
            if let error = error {
                print("Error: \(error.localizedDescription)")
            } else {
                application.registerForRemoteNotifications()
            }
        }
        UNUserNotificationCenter.current().delegate = self

        // Register subscriptions
        CloudKitManager.sharedInstance.registerSubscriptionsWithIdentifier("id2")
        CloudKitManager.sharedInstance.registerSilentSubscriptionsWithIdentifier("id3")
        //configureUserNotificationsCenter()

        if let notification = launchOptions?[UIApplicationLaunchOptionsKey.remoteNotification] as? [String: Any] {
            // 2
            let aps = notification["aps"] as! [String: Any]
            print("APS: \(aps)")
            // 3
            //((window?.rootViewController as? UITabBarController)?.selectedIndex = 1
        }
        return true
    }
}
```

SUBSCRIPTIONS

- Subscriptions have advanced behavior over notifications
- Delivery behavior
- Can contain record references

```
//  
// MARK: - Notifications Support  
  
/// Create an alert to show if the application is active and receives a local  
/// notification  
/// - parameter notification: The `UILocalNotification` received  
func showAlertForNotification(userInfo: [NSObject : AnyObject]) {  
  
    // Do not show unless the application is active  
    guard UIApplication.sharedApplication().applicationState == .Active else { return }  
  
    // Create the alert  
    let alertController = UIAlertController(title: "Received Notification",  
                                           message: userInfo.description,  
                                           preferredStyle: .Alert)  
  
    // Create cancel action that does nothing  
    let cancelAction = UIAlertAction(title: "Cancel", style: .Cancel, handler: nil)  
    alertController.addAction(cancelAction)  
  
    // Show the alert and exit early  
    self.window?.rootViewController?.presentViewController(alertController,  
                                                          animated: true,  
                                                          completion: nil)  
  
    //let vc = window?.rootViewController as? ViewController  
    //vc?.refreshTable()  
}  
  
/// Alternative (more flexible way)  
func showNotificationNowFromCloudKit(notification: UILocalNotification) {  
    // Create a notification  
    let notification = UILocalNotification()  
    notification.alertBody = "This was from a Now notification."  
    notification.alertAction = "Ok!"  
  
    // Schedule the notification  
    UIApplication.sharedApplication().presentLocalNotificationNow(notification)  
}
```

SUBSCRIPTIONS

```
didReceiveResponseNotification: <UNNotificationResponse: 0x1c422e600; actionIdentifier: com.apple.UNNotificationDefaultActionIdentifier, notification: <UNNotification: 0x1c4422280; date: 2017-11-07 16:27:44 +0000, request: <UNNotificationRequest: 0x1c42285e0; identifier: 58374967-54A3-443F-A511-8C5B1A5F0252, content: <UNNotificationContent: 0x1c411f650; title: (null), subtitle: (null), body: A new joke was added., categoryIdentifier: , launchImageName: , peopleIdentifiers: ( ), threadIdentifier: , attachments: ( ), badge: 125, sound: (null), hasDefaultAction: YES, defaultActionTitle: (null), shouldAddToNotificationsList: YES, shouldAlwaysAlertWhileAppIsForeground: NO, shouldLockDevice: NO, shouldPauseMedia: NO, isSnoozeable: NO, fromSnooze: NO, darwinNotificationName: (null), darwinSnoozedNotificationName: (null), trigger: <UNPushNotificationTrigger: 0x1c400f560; contentAvailable: YES, mutableContent: NO>>>
Notification: <UNNotificationContent: 0x1c411f650; title: (null), subtitle: (null), body: A new joke was added., categoryIdentifier: , launchImageName: , peopleIdentifiers: ( ), threadIdentifier: , attachments: ( ), badge: 125, sound: (null), hasDefaultAction: YES, defaultActionTitle: (null), shouldAddToNotificationsList: YES, shouldAlwaysAlertWhileAppIsForeground: NO, shouldLockDevice: NO, shouldPauseMedia: NO, isSnoozeable: NO, fromSnooze: NO, darwinNotificationName: (null), darwinSnoozedNotificationName: (null)
```

**DEBUGGING
NOTIFICATION
HANDLING**

SUBSCRIPTIONS

Guides and Sample Code

Apple Developer



Debugging issues with CloudKit subscriptions

Technical Q&A QA1917

Debugging issues with CloudKit subscriptions

Q: My CloudKit subscriptions don't trigger notifications after relevant changes are made. How to debug that?

A: Most CloudKit subscription issues are either due to incorrect assumptions about when and where notifications should fire or improperly configured CloudKit containers or subscriptions. This document will introduce you some cases that aren't expected to trigger CloudKit notifications, following with how to verify the state of your iCloud container and how to avoid some common issues related to CloudKit subscriptions.

Cases that aren't expected to trigger CloudKit notifications

When working with CloudKit subscriptions, be aware that:

- Notifications won't be delivered to your device if the notification settings for your app are off. CloudKit relies on the Apple Push Notification service (APNs) to deliver notifications. If your app is not allowed for push notifications on the device, you won't see them. To turn the settings on, go to `Settings > Notifications`, then navigate to your app's setting screen.
- CloudKit notifications won't be delivered to the device on which the relevant changes are made.
- The Simulators don't support push notifications. To test push notifications you must be running directly on the target platform.
- CloudKit notifications won't be delivered to your app if the notifications' `shouldSendContentAvailable` property is `true` and meanwhile your app is force quit. On iOS, users can force quit an app by double-tapping the home button and swiping it away from the multitasking UI.
- CloudKit generates a notification for every relevant change. However, notifications can be co-delivered. Thus can retrieve the unhandled ones with `CKFetchNotificationChangesOperation` and process them from there.

[HTTPS://DEVELOPER.APPLE.COM/LIBRARY/CONTENT/QA/QA1917/_INDEX.HTML](https://developer.apple.com/library/content/qa/QA1917/_index.html)

SUBSCRIPTIONS

```
// Alert that will show over the entire application to debug the notifications
func alert(title: String, message: String) {
    let alert = UIAlertController(title: title, message: message, preferredStyle: .alert)
    alert.addAction(UIAlertAction(title: "OK",
                                 style: .`default`,
                                 handler: { _ in
                                    NSLog("The \"OK\" alert occurred.")
        }))
    let rvc = (window?.rootViewController as? UITabBarController)
    rvc?.selectedViewController?.present(alert, animated: true, completion: nil)
}
```

SUBSCRIPTIONS

```
// Throw a local notification now to test the launch cycle
func now(message: String, time: Double = 2) {
    let notificationContent = UNMutableNotificationContent()
    notificationContent.title = "👋"
    notificationContent.body = message

    // Add Trigger
    let notificationTrigger = UNTimeIntervalNotificationTrigger(timeInterval: TimeInterval(time),
                                                                repeats: false)

    // Create Notification Request
    let notificationRequest = UNNotificationRequest(identifier: UUID.init().debugDescription,
                                                    content: notificationContent,
                                                    trigger: notificationTrigger)

    // Add Request to User Notification Center
    UNUserNotificationCenter.current().add(notificationRequest) { (error) in
        if let error = error {
            print("Unable to Add Notification Request (\(error), \(error.localizedDescription))")
        }
    }
}
```

SUBSCRIPTIONS



Cloudy...OfErrors T. Andrew Binkowski's iPhone 6 Finished running CloudyWithAChanceOfErrors on T. Andrew Binkowski's iPhone 6 6

Buildtime (6) Runtime

CloudyWithAChanceOfErrors (6)

- Auto Layout Localization
- Fixed width constraints may cause clipping. Main.storyboard width = 200
- Fixed width constraints may cause clipping. Main.storyboard width = 200
- Fixed width constraints may cause clipping. Main.storyboard width = 200
- Unsupported Configuration Prototype table cells must have reuse identifiers Main.storyboard
- Dependency Analysis Warning The use of Swift 3 @objc inference in Swift 4 mode is deprecated. Please ...
- Unsupported Configuration Prototype table cells must have reuse identifiers Main.storyboard

```
style: .`default`,
handler: { _ in
    NSLog("The \"OK\" alert occurred.")
})
let rvc = (window?.rootViewController as? UITabBarController)
rvc?.selectedViewController?.present(alert, animated: true, completion: nil)
}

/// Throw a local notification now to test the launch cycle
func now(message: String, time: Double = 2) {
    let notificationContent = UNMutableNotificationContent()
    notificationContent.title = "🕒"
    notificationContent.body = message

    // Add Trigger
    let notificationTrigger = UNTimeIntervalNotificationTrigger(timeInterval: TimeInterval(time), repeats: false)

    // Create Notification Request
    let notificationRequest = UNNotificationRequest(identifier: UUID().debugDescription,
                                                    content: notificationContent,
                                                    trigger: notificationTrigger)

    // Add Request to User Notification Center
    UNUserNotificationCenter.current().add(notificationRequest) { (error) in
        if let error = error {
            print("Unable to Add Notification Request (\(error), \(error.localizedDescription))")
        }
    }
}
```

Identity and Type

- Name AppDelegate.swift
- Type Default - Swift Source
- Location Relative to Group AppDelegate.swift
- Full Path /Users/tabinowski/Google Drive/g-Teaching/uchicago.cloud/mpcs51033-2017-autumn/mpcs51033-2017-autumn-code-samples/cloudkit/CloudyWithAChanceOfErrors/

View Controller - A controller that manages a view.

Storyboard Reference - Provides a placeholder for a view controller in an external storyboard.

Navigation Controller - A controller that manages navigation through a hierarchy of views.

Table View Controller - A controller that manages a table view.

Collection View Controller - A controller that manages a collection view.

WEB SERVICES

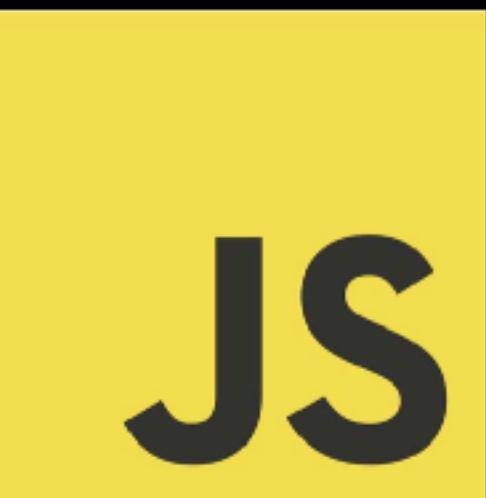


PREVIEW

WEB SERVICE API

- Javascript API
- Matches native CloudKit API
- No intermediate servers
- New notes web app built with CloudKit JS

```
<SCRIPT SRC="HTTPS://CDN.APPLE-CLOUDKIT.COM/CK/1/CLOUDKIT.JS" />
```



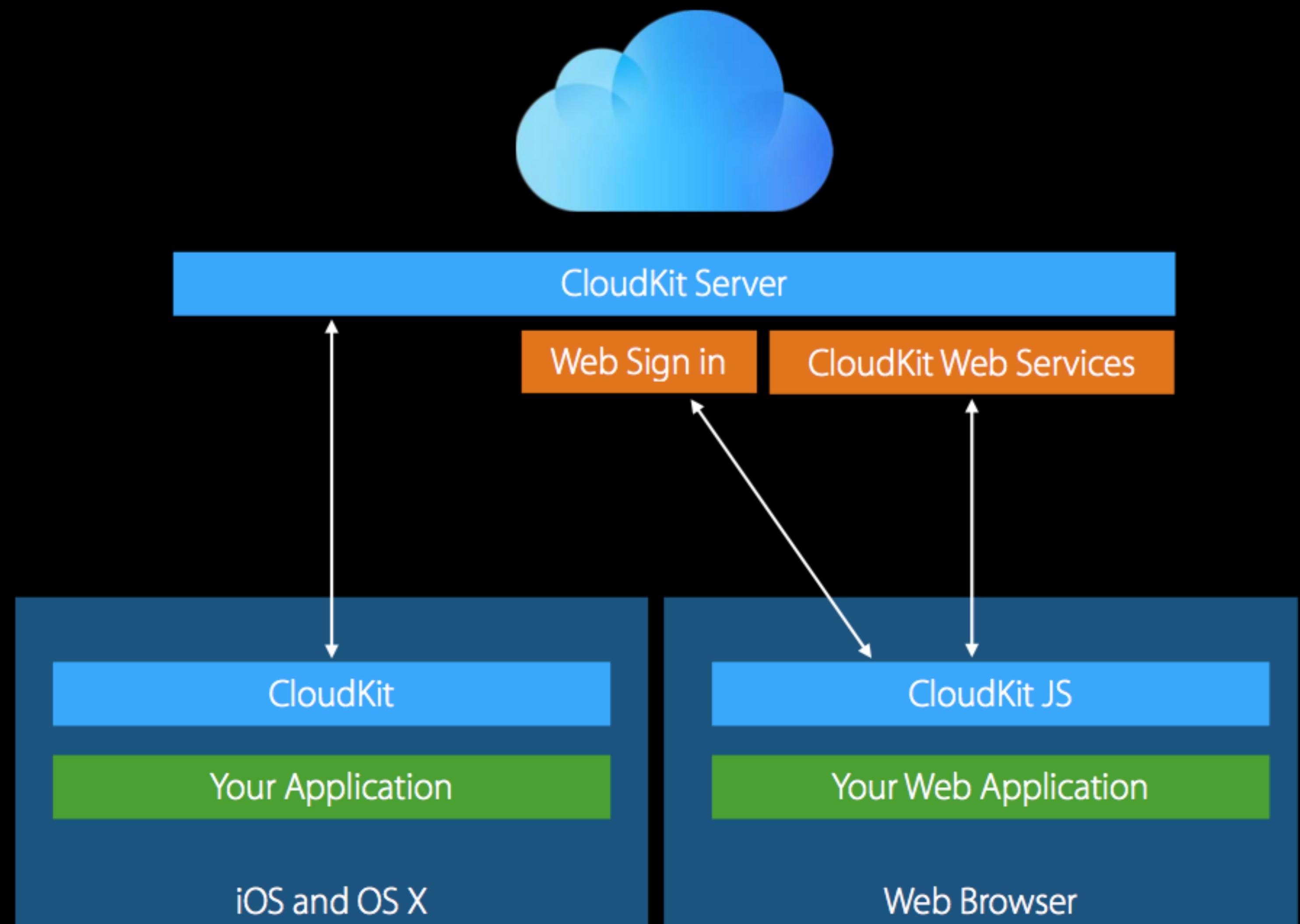
WEB SERVICE API

- Public and private database access
- Record operations
- Assets
- Query
- Subscriptions and notifications
- User discoverability
- Sync



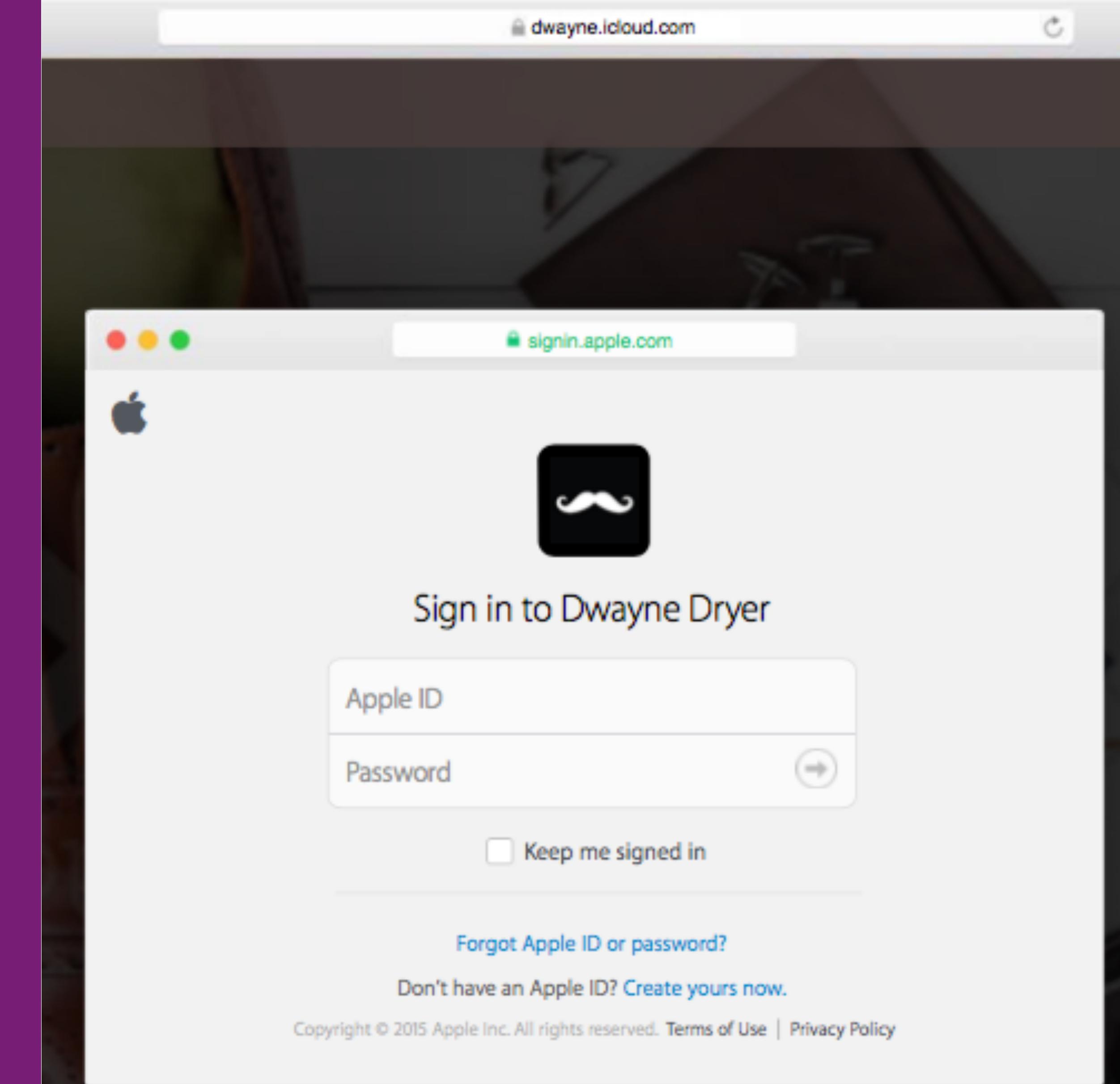
WEB SERVICES

- New server-to-server capabilities
- Add servers to cover the areas that cloud kit doesn't cover



WEB SERVICE API

- Web based authentication
- Need iCloud account
 - New users can sign-up for a free 1GB web only account



WEB SERVICE API

 CloudKit Catalog

 README

 Authentication

 Discoverability

 Query

 Zones

 Records

 Sync

 Sharing

 Subscriptions

 Notifications
Disconnected

 Run Code

Container: iCloud.com.example.CloudKitCatalog Environment: production

CloudKit on the web

This web application provides executable sample code for the core API methods provided by the CloudKit JS JavaScript library. While these methods cover many typical use cases, there are more flexible versions available if needed which allow for batch requests and more configuration. The user is advised to refer to the [CloudKit JS Reference](#) for more information.

All code examples can be run by clicking the play button at the top of the page. The results will be displayed below the sample code block.

Obtaining the CloudKit JS library

CloudKit JS is hosted at <https://cdn.apple-cloudkit.com/ck/2/cloudkit.js>. Include the library on your web page using either of the two methods below. You will automatically get updates and bug fixes as they are released.

Option #1 - Load CloudKit JS synchronously

```
<script src="https://cdn.apple-cloudkit.com/ck/2/cloudkit.js"></script>
```

Unauthenticated User

WHEN TO USE CLOUDKIT

WHEN TO USE CLOUDKIT

- You're developing exclusively for Apple platforms
- You use it as a partial solution

ASSIGNMENT 4

ASSIGNMENT 4

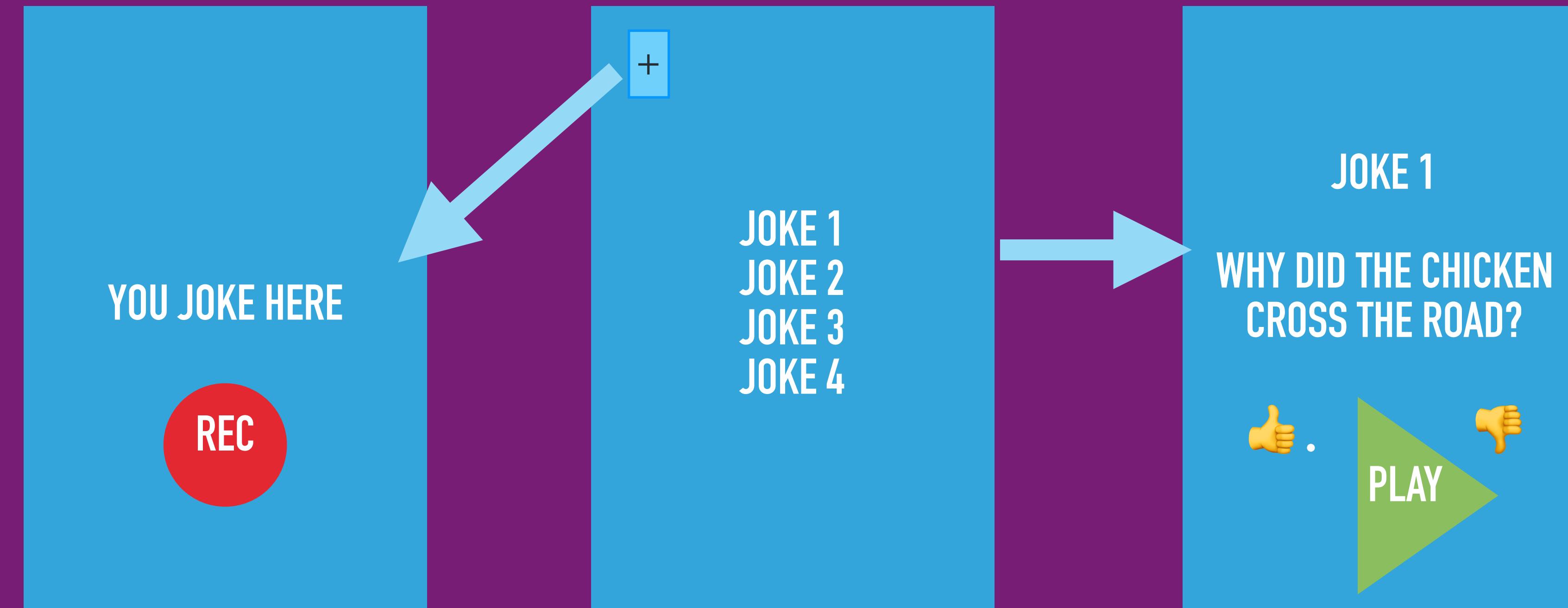
- Create a "Joke of the Day" application
- Users submit jokes (text and audio)
- Users can view/listen to all jokes and rate them
 - New jokes trigger a silent push notification to update local data
- Use the JS API to trigger a daily "Joke of the Day" notification sent to all users from a different service
- Create a simple interface to send push notifications to users
 - iOS, macOS or web app

ASSIGNMENT 3

GOOGLE CLOUD
PLATFORM



JOKE OF THE DAY





MPCS 51033 • AUTUMN 2017 • SESSION 7

BACKENDS FOR MOBILE APPLICATIONS