THE UNIVERSITY OF CHICAGO

ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 3

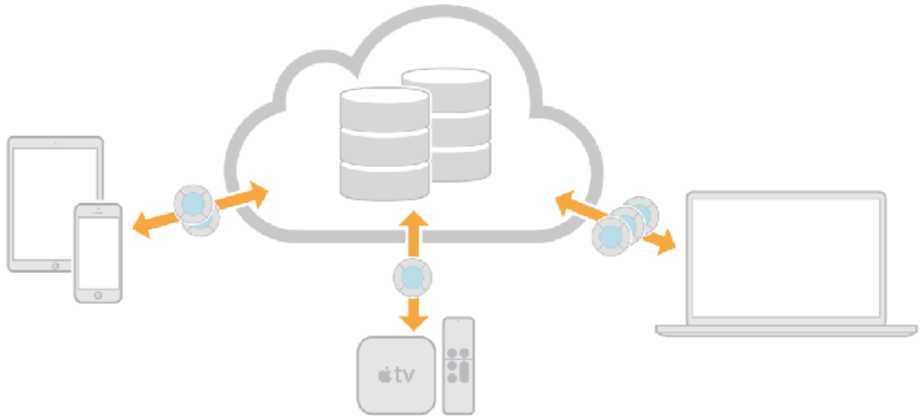
CLOUDKIT 

CLOUDKIT

- CloudKit QuickStart

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

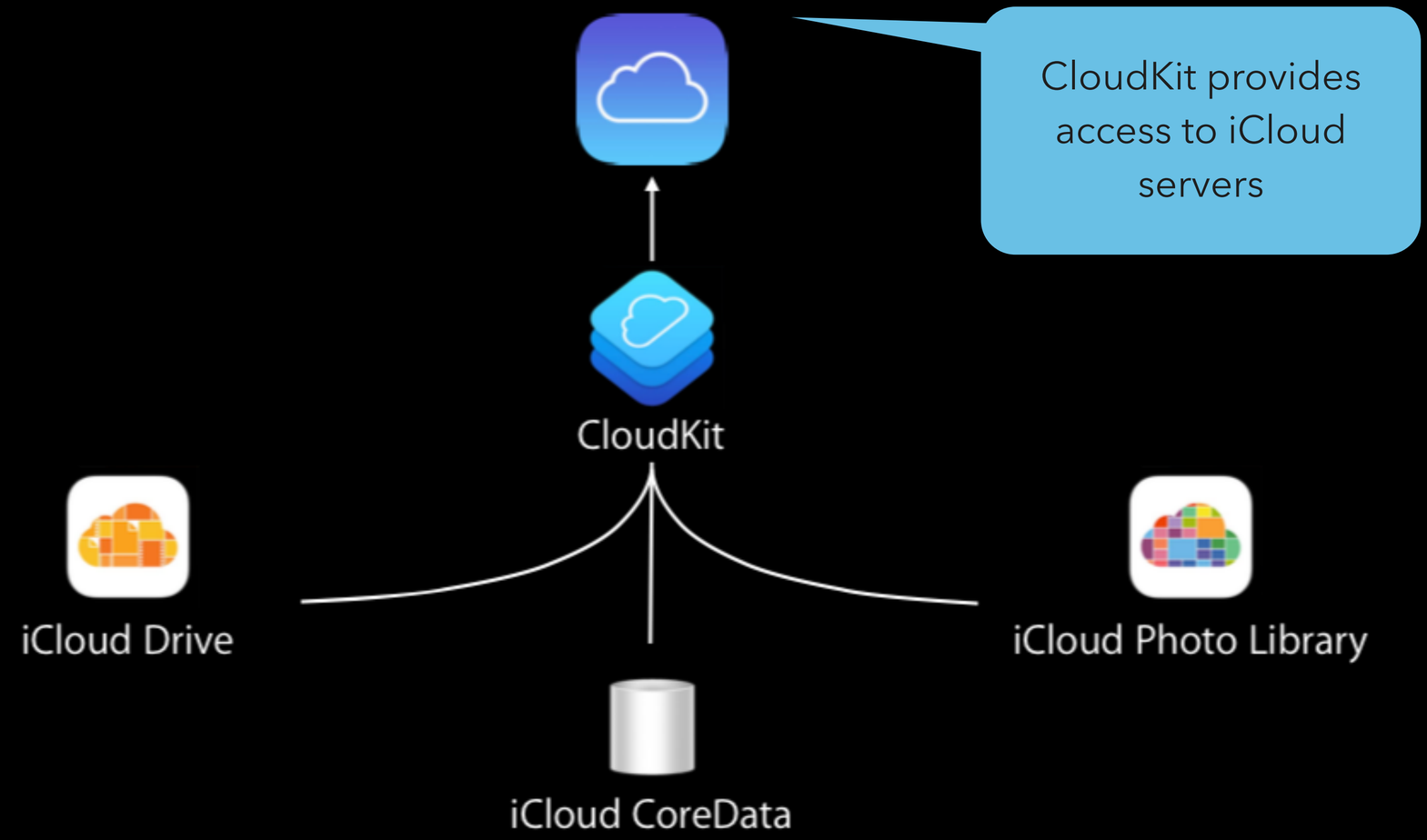
See Glossary for the definition of database terms used in this book.

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.

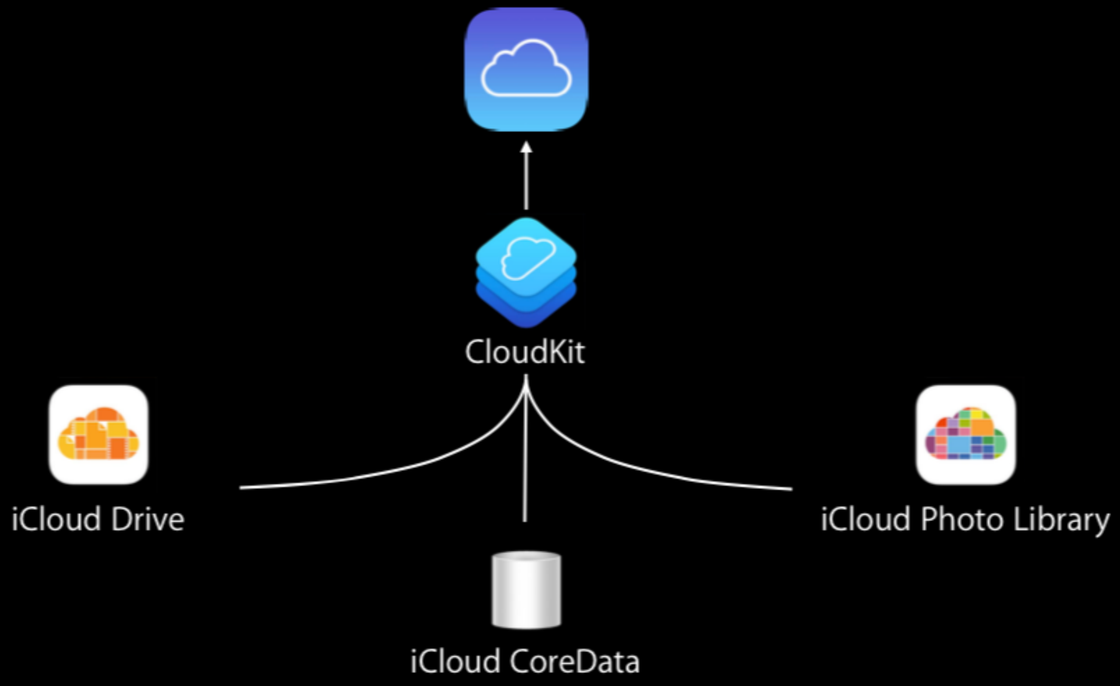
CLOUDKIT


CloudKit provides access to iCloud servers
CloudKit
iCloud Drive
iCloud Photo Library
iCloud CoreData

CLOUDKIT

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

#1 CONSIDERATION ON CHOOSING OVER OTHER SERVICES

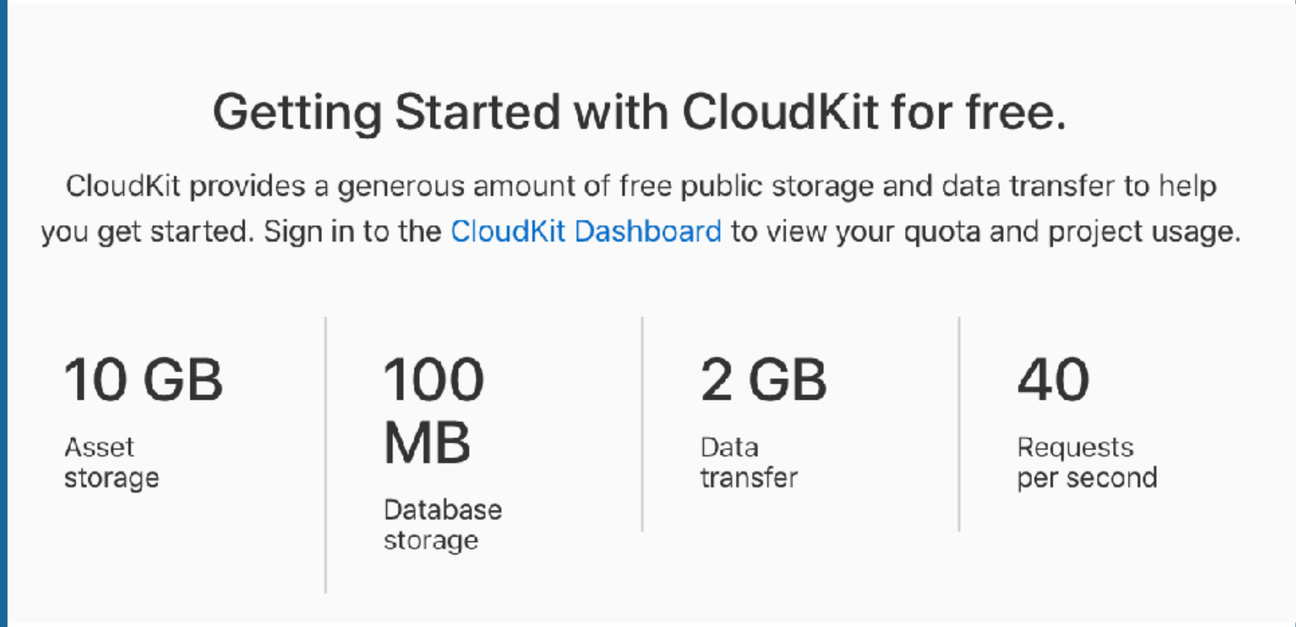


CLOUDKIT

- Free...with limits

CLOUDKIT

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



CLOUDKIT

- Nice problem to have

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

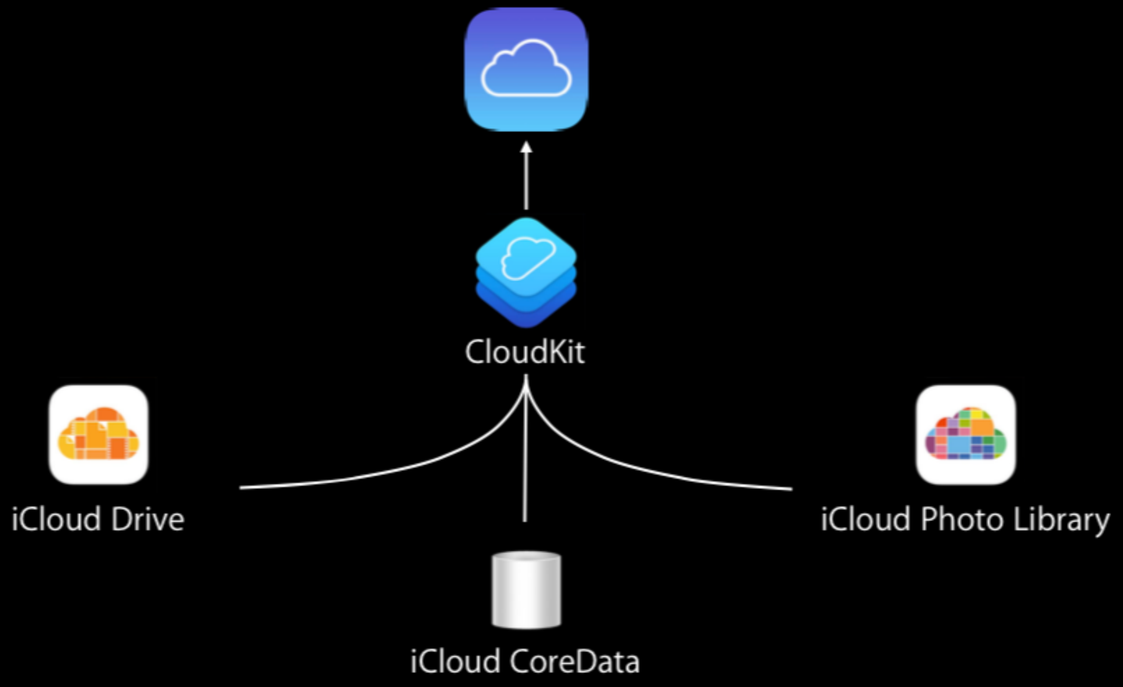
1 PB	10 TB	200 TB	400
Asset storage	Database storage	Data transfer	Requests per sec.
Based on: 100 MB per user	Based on: 1 MB per user	Based on: 20 MB per user	Based on: 4 per 100,000 users

$0

Total Cost

CLOUDKIT

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



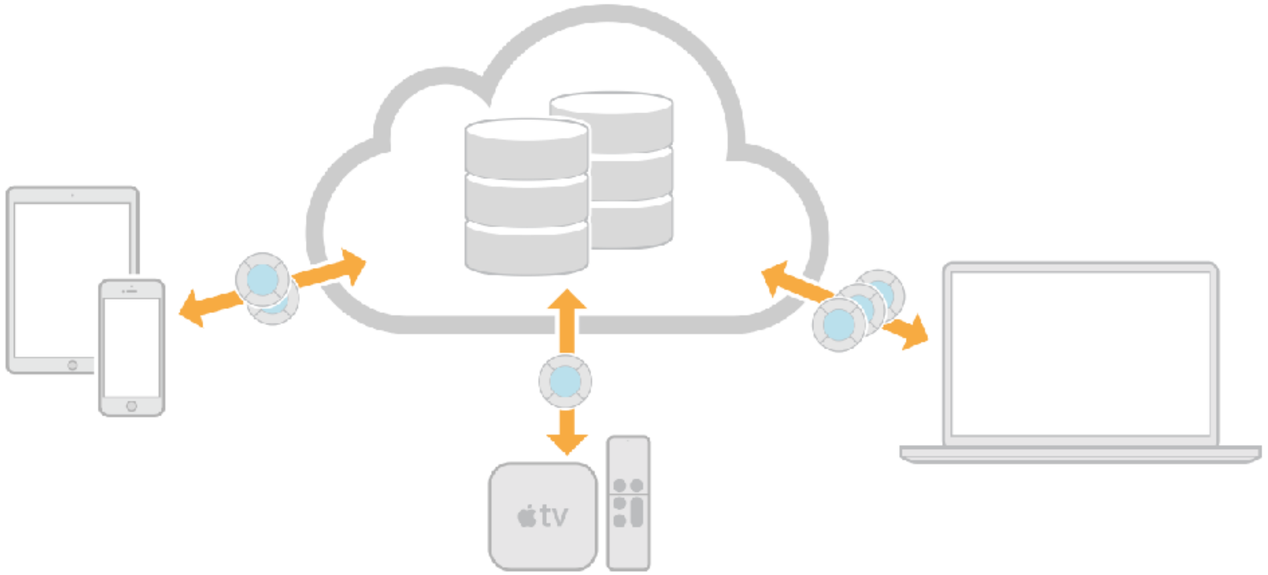
CLOUDKIT

- Databases
  - Public
  - Private
  - Shared

CLOUDKIT

SUBTITLE

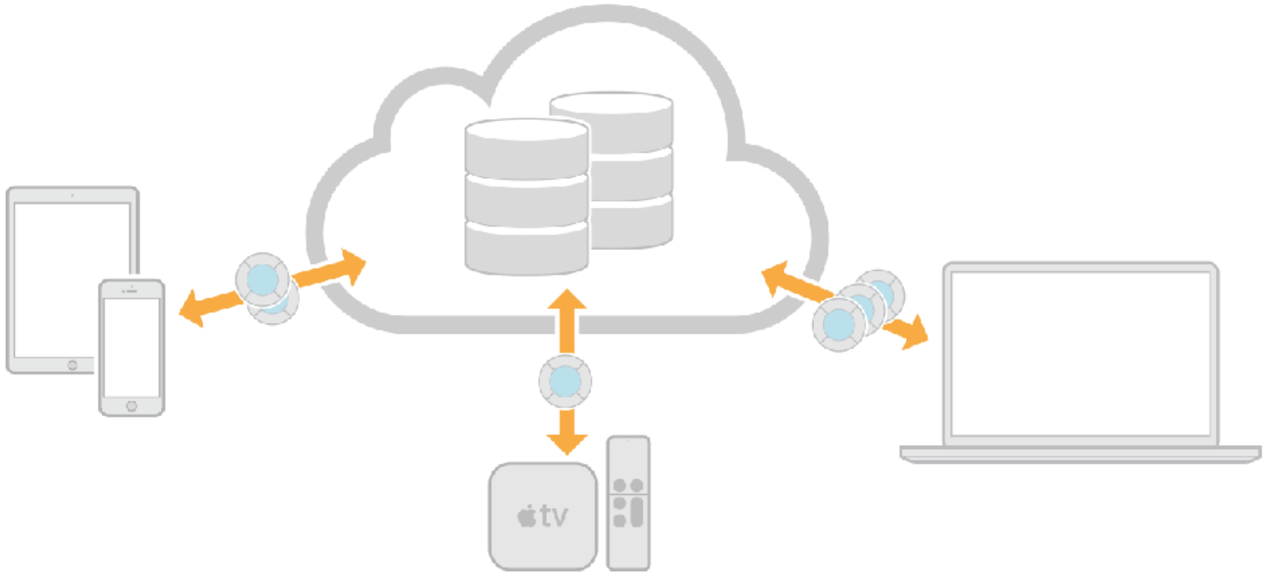
- Structured and bulk data
  - Data with types
  - Blobs



CLOUDKIT

CORE OBJECTS

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

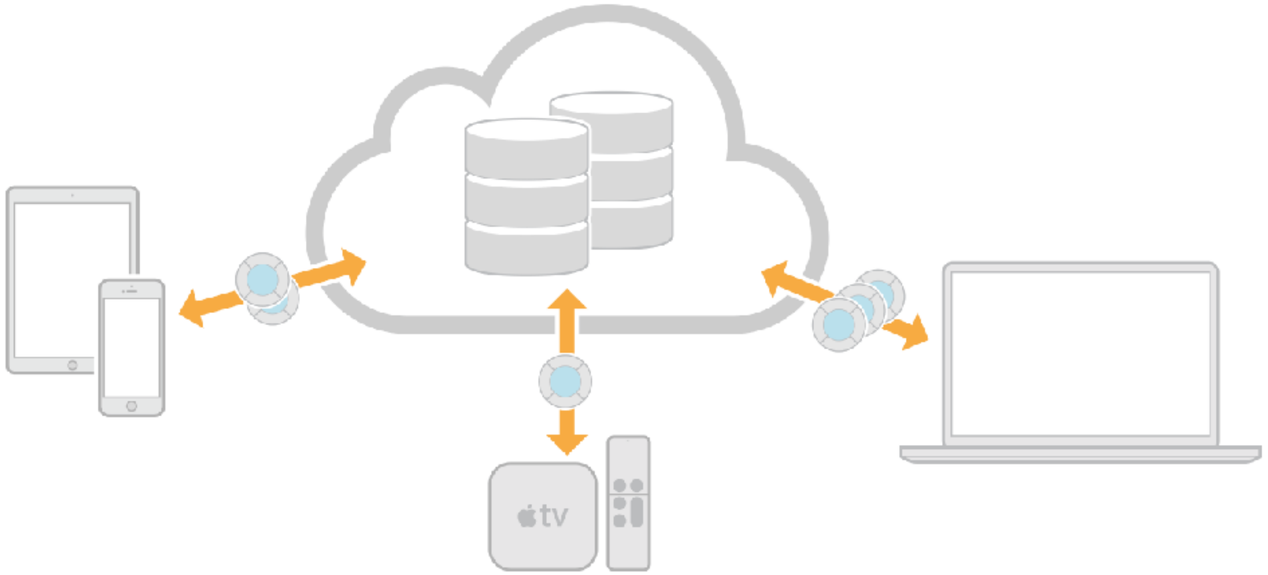


CLOUDKIT

SUBTITLE

- 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 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

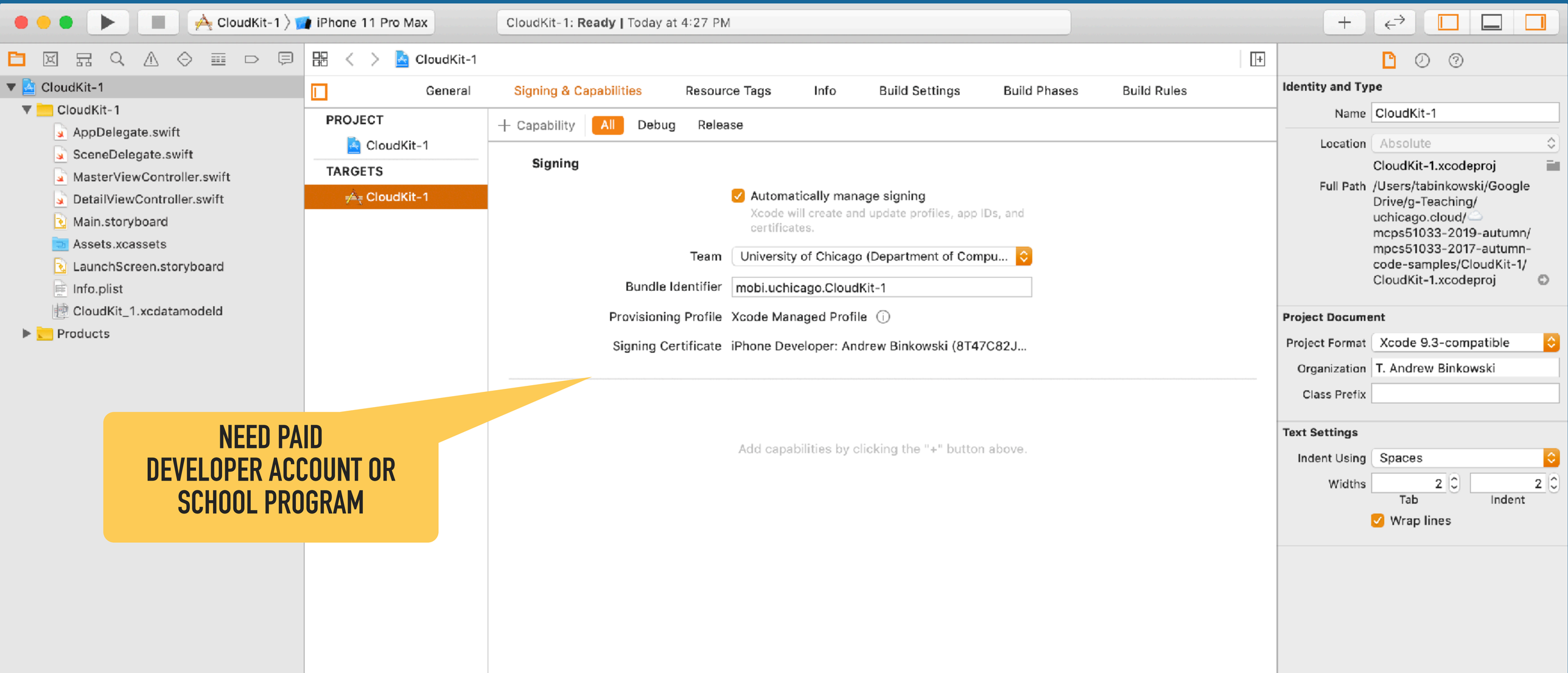
SUBTITLE

- 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

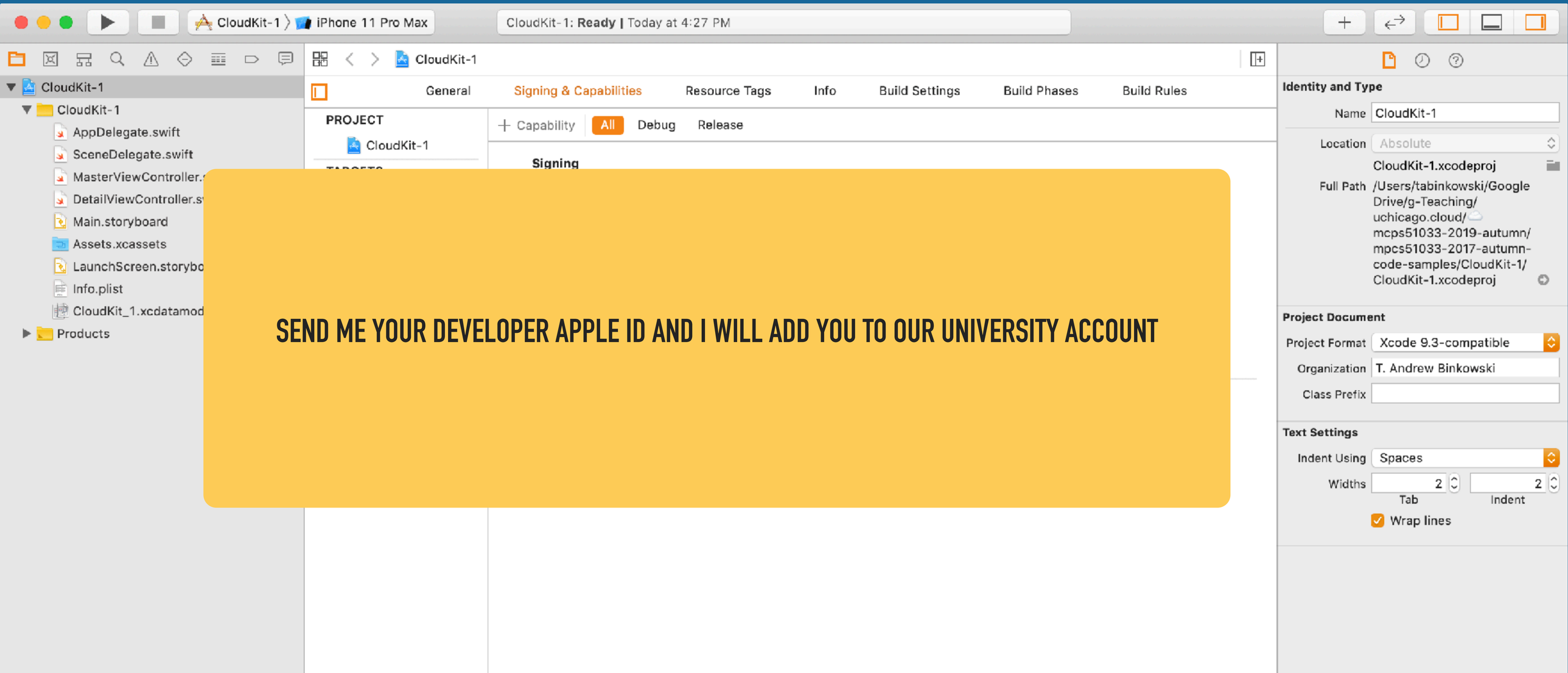
ENABLING CLOUDKIT IN YOUR APPLICATION

ENABLING CLOUDKIT IN YOUR APPLICATION



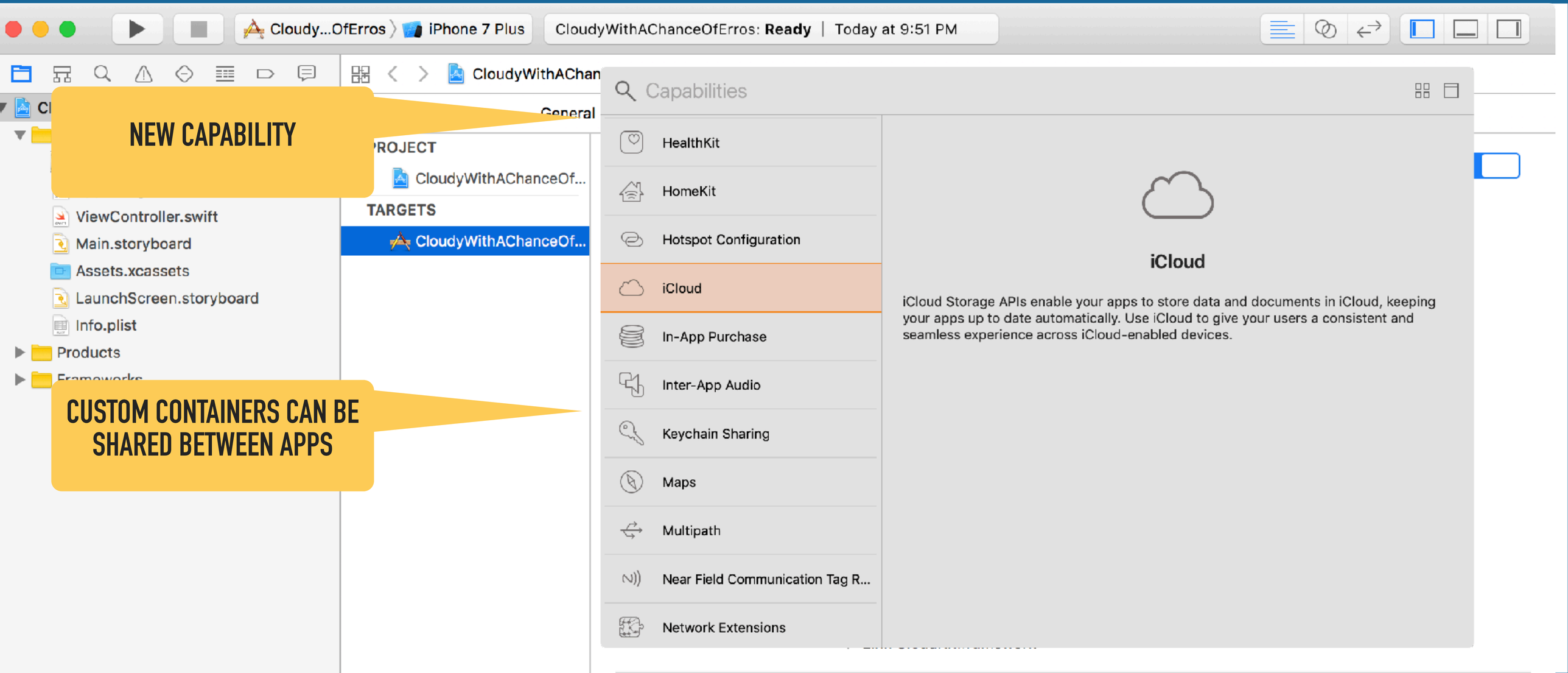
NEED PAID DEVELOPER ACCOUNT OR SCHOOL PROGRAM

ENABLING CLOUDKIT IN YOUR APPLICATION

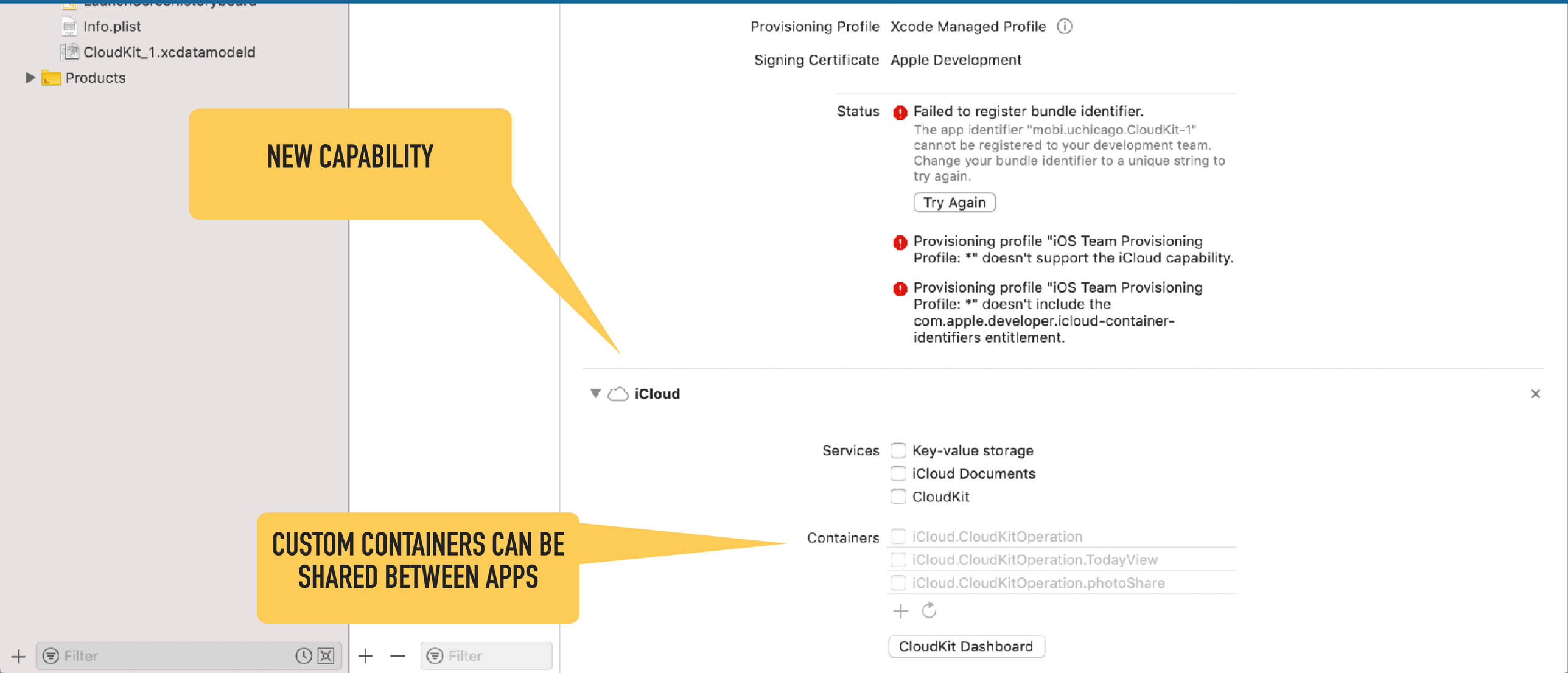


SEND ME YOUR DEVELOPER APPLE ID AND I WILL ADD YOU TO OUR UNIVERSITY ACCOUNT

ENABLING CLOUDKIT IN YOUR APPLICATION



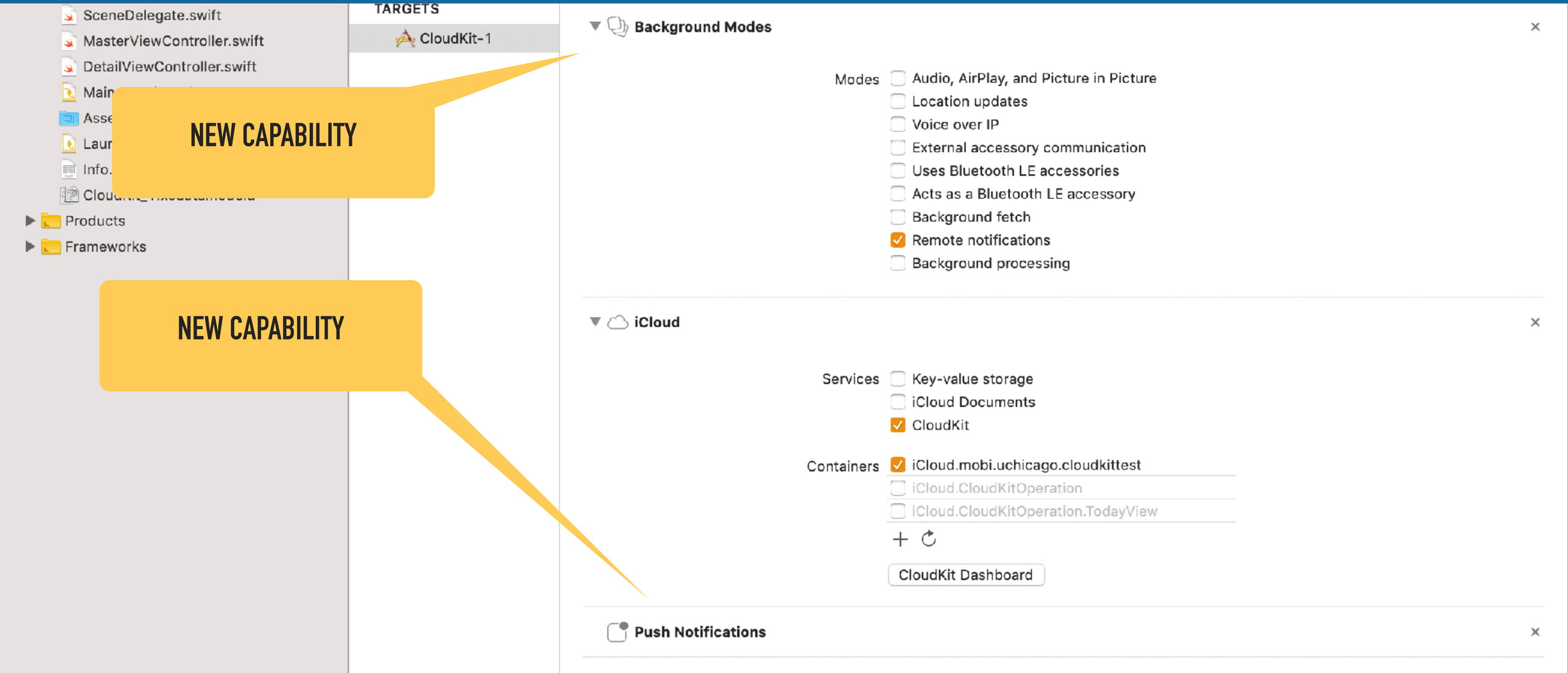
ENABLING CLOUDKIT IN YOUR APPLICATION



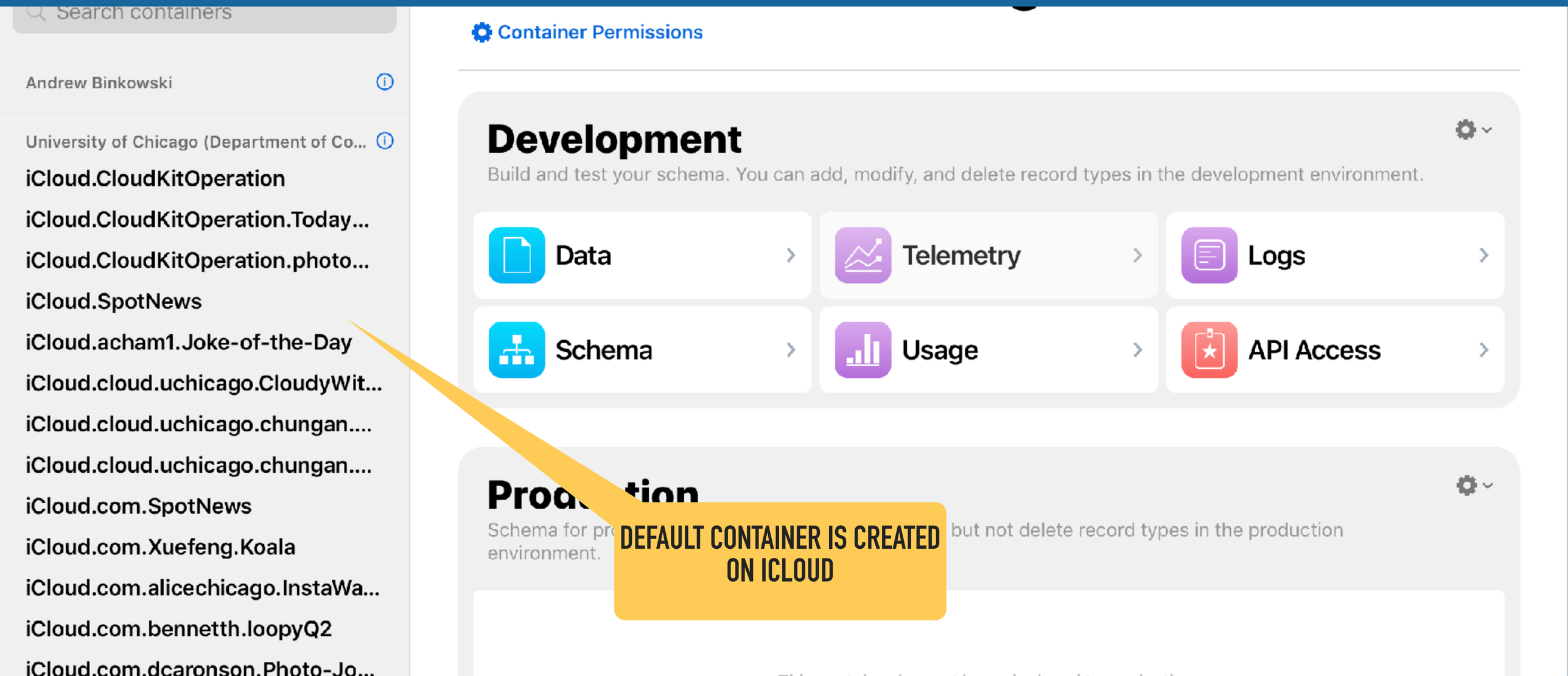
NEW CAPABILITY

CUSTOM CONTAINERS CAN BE SHARED BETWEEN APPS

ENABLING CLOUDKIT IN YOUR APPLICATION

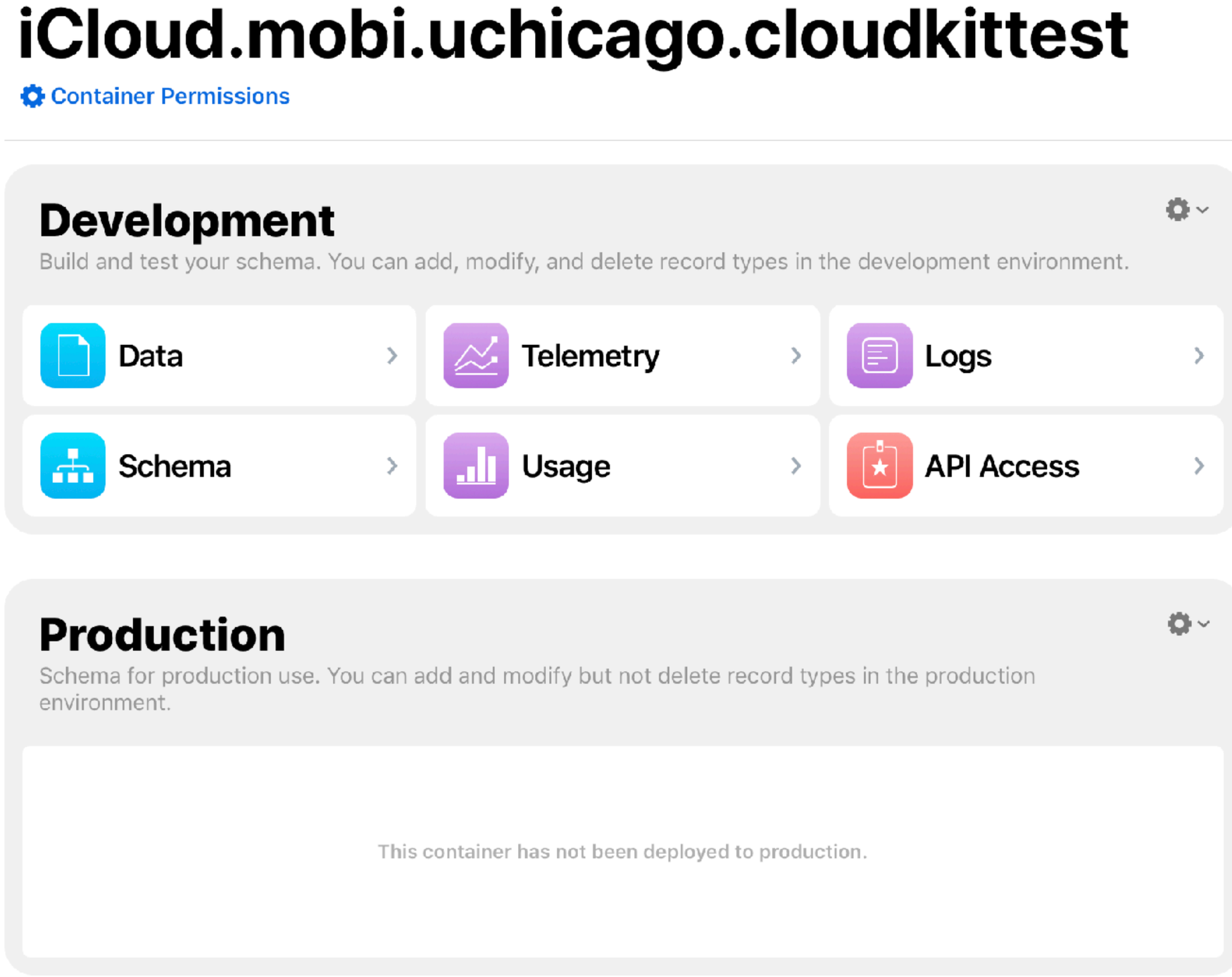


ENABLING CLOUDKIT IN YOUR APPLICATION



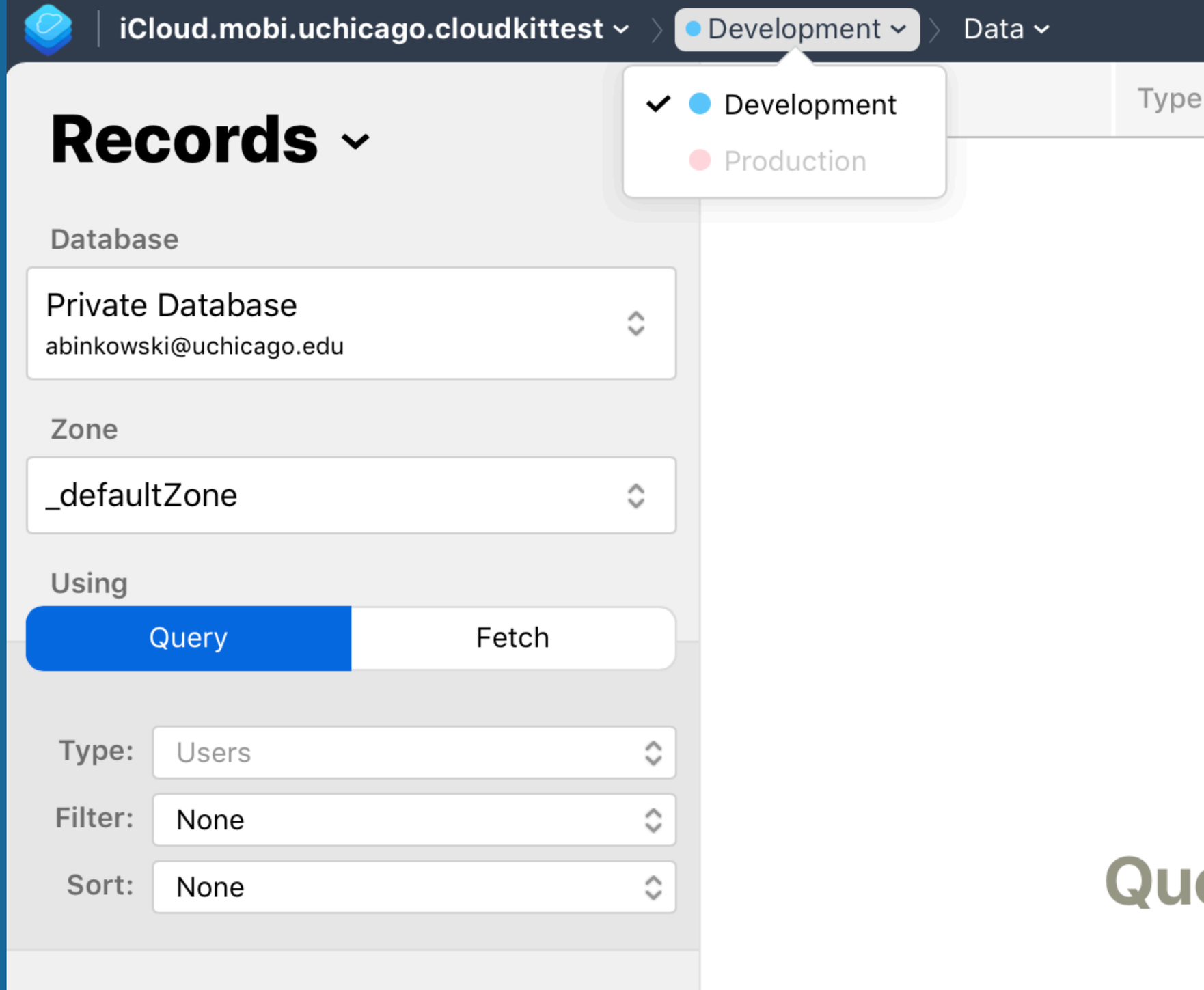
ENABLING CLOUDKIT IN YOUR APPLICATION

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

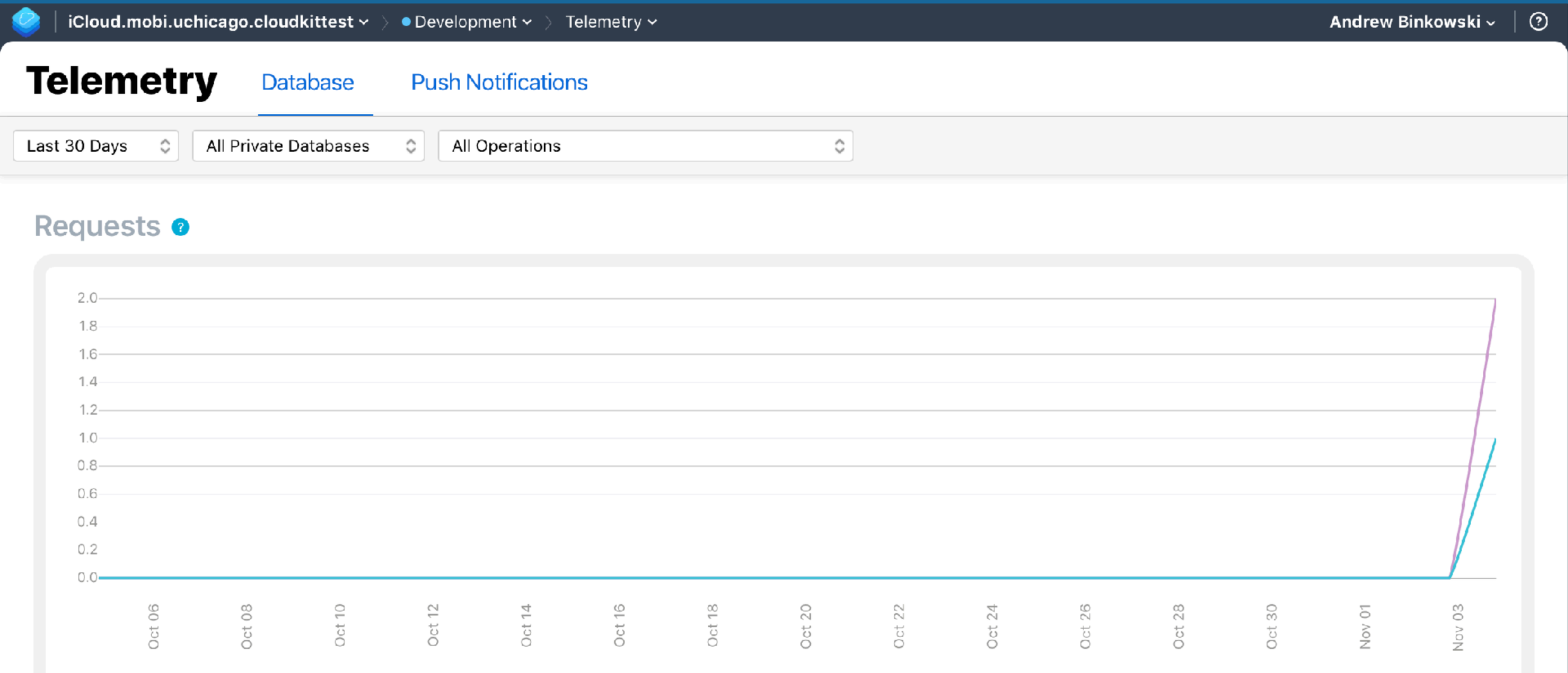


ENABLING CLOUDKIT IN YOUR APPLICATION

- "Flip a switch" to go to production mode



ENABLING CLOUDKIT IN YOUR APPLICATION



WHEN TO USE CLOUDKIT

WHEN TO USE CLOUDKIT

- Whenever possible
- When you have a tightly defined problem to solve
- You're developing exclusively for Apple platforms (or willing to do a lot of work)

WHEN TO USE CLOUDKIT

- Sync is hard
- Lot's of approaches

malhal

iOS developer, since the beginning

Apple WWDC app

CloudKit Sync Nightmare

malhal

13th January 2017

CloudKit

So after previous post on analysing the web requests, where we found out the classes involved and sort of when they should be used, we are still clueless as to how to structure our app to enable CloudKit syncing. We know we shouldn't subclass fetch operation but then how should we gather our information to pass to the ope the ger cor ret Clo lon

First of all we notice 336 files. 336 files just for the networking and caching, not the UI. Unbelievable, but lets carry on anyway.

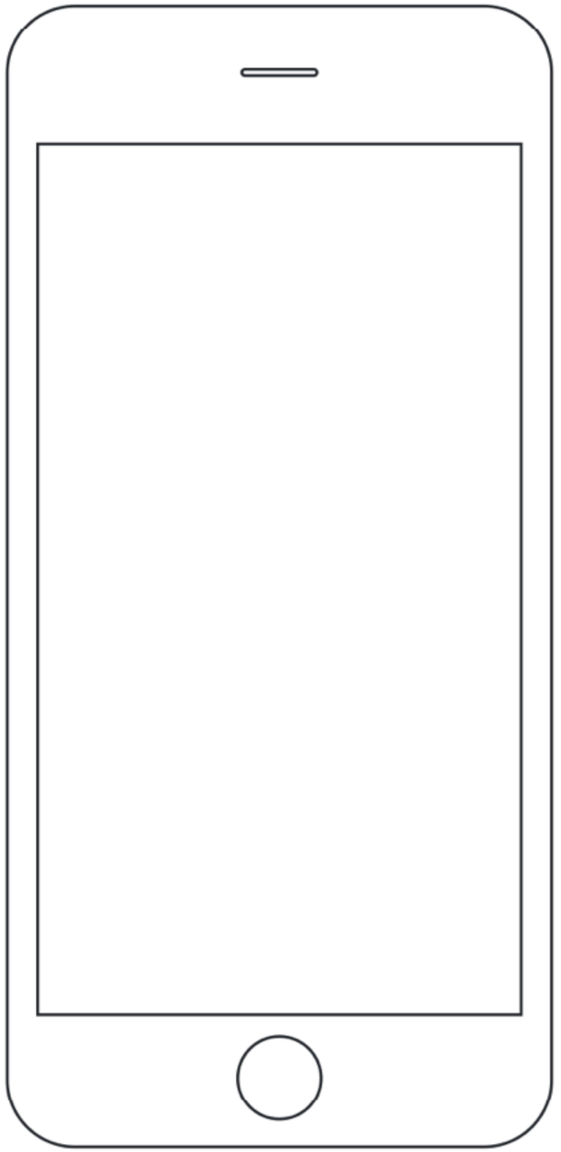
A lot of the CloudKit related files are prefixed with FCCK, 27 of them! And there are many more like controllers etc. that use these objects. Maybe half the files have a reference to some aspect of CloudKit one way or another.

To be honest I wasn't expecting this complicated an app, this looks like at least a year of coding, maybe even more. Remember the app only syncs a few record types, reading list (bookmarks), history (read articles) and personalisation data like favourite feeds. Implementing syncing is going to be no easy task at all; no wonder we are all clueless after a few hours of WWDC talks! Hopefully some of my research has helped you though.

ASSIGNMENT 3

ASSIGNMENT 3

- Picture Journal App
  - Users select a picture and add a caption
  - Meta data generated from picture
  - Share and Today Extensions
  - Sync using CloudKit subscriptions



ASSIGNMENT 3

Day One

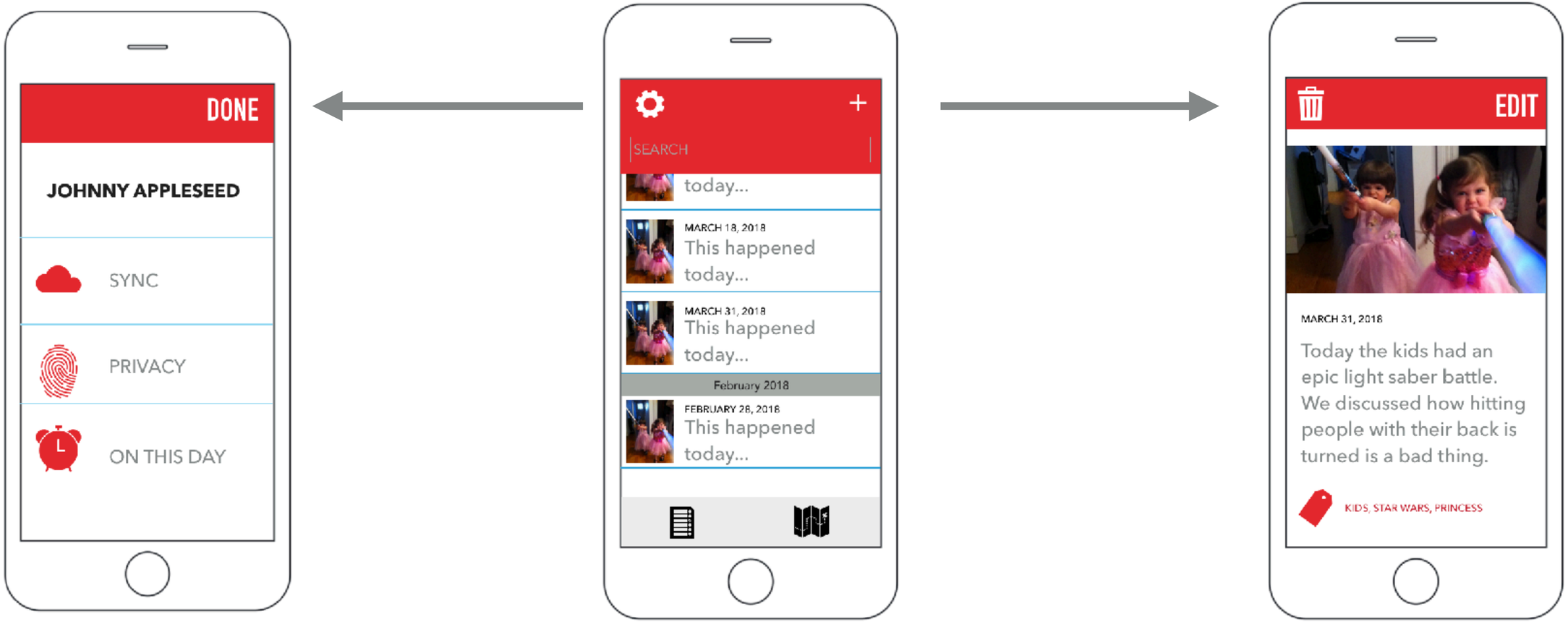
ASSIGNMENT 3

- Day One
  - Apple Design Award winner
  - Success as a paid app (for a while)
  - Moved to subscription model
  - Move from using Dropbox flat files to iCloud Drive to custom sync solution

ASSIGNMENT 3

OVERVIEW

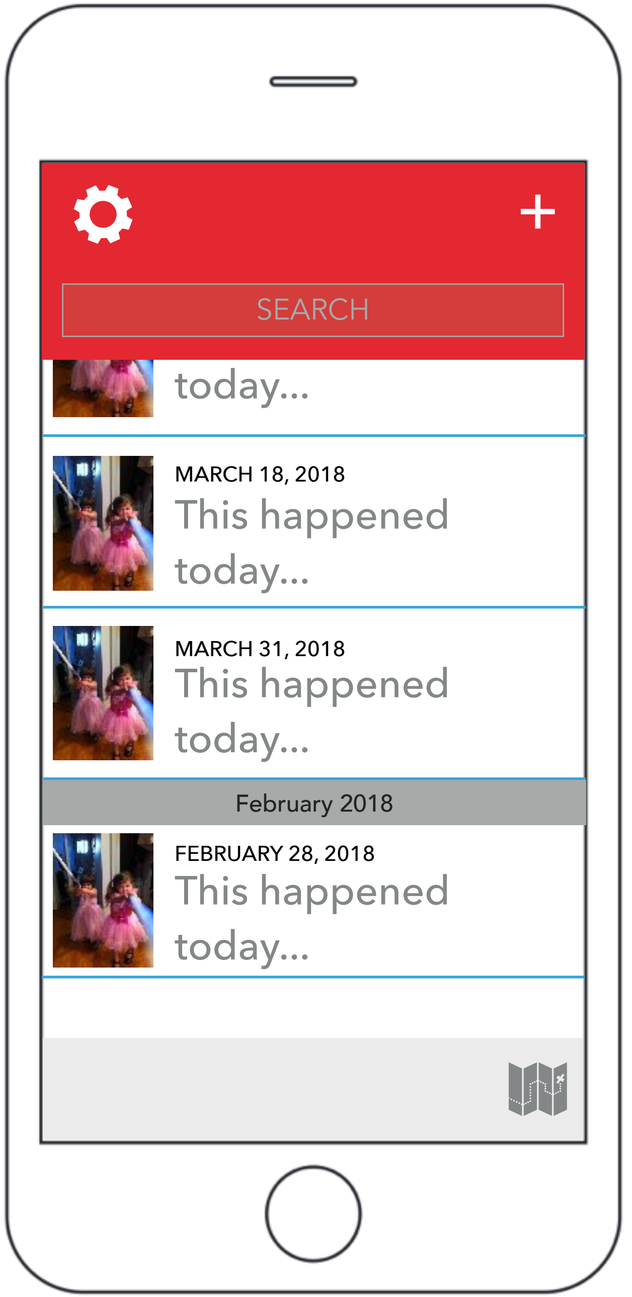
CLOUDKIT



ASSIGNMENT 3

HOME SCREEN

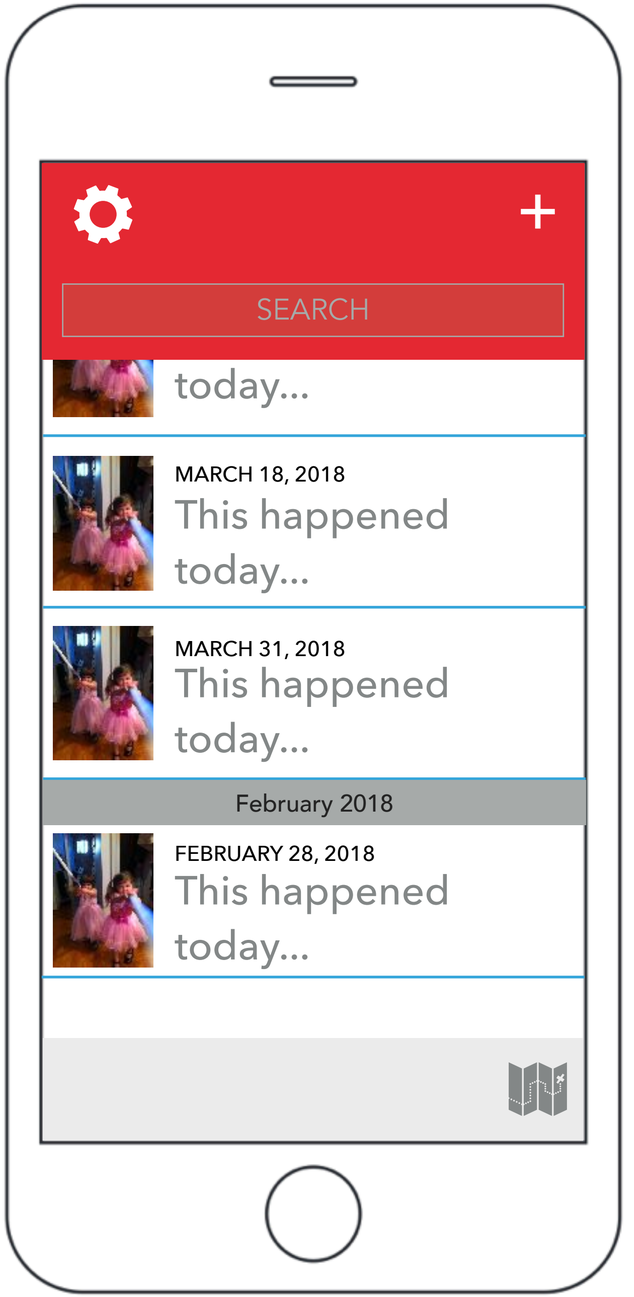
- Home screen is list
  - TableView (or CollectionView) should be segmented by month
  - Cell should display thumbnail, text blurb, and date (can include more if like)
  - Search bar should search by text and tags



ASSIGNMENT 3

HOME SCREEN

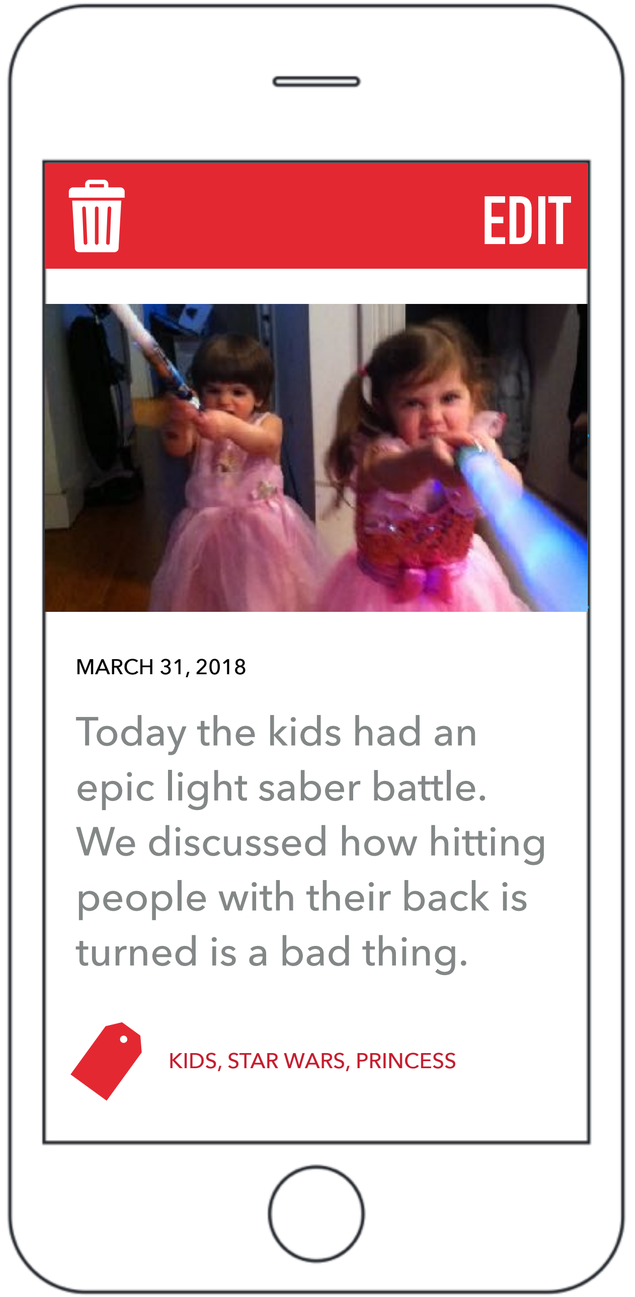
- Map screen will show locations from entries
  - Tapping on entry will show all entries in that location via location based query
  - Show back on main home screen



ASSIGNMENT 3

ADD/EDIT SCREEN

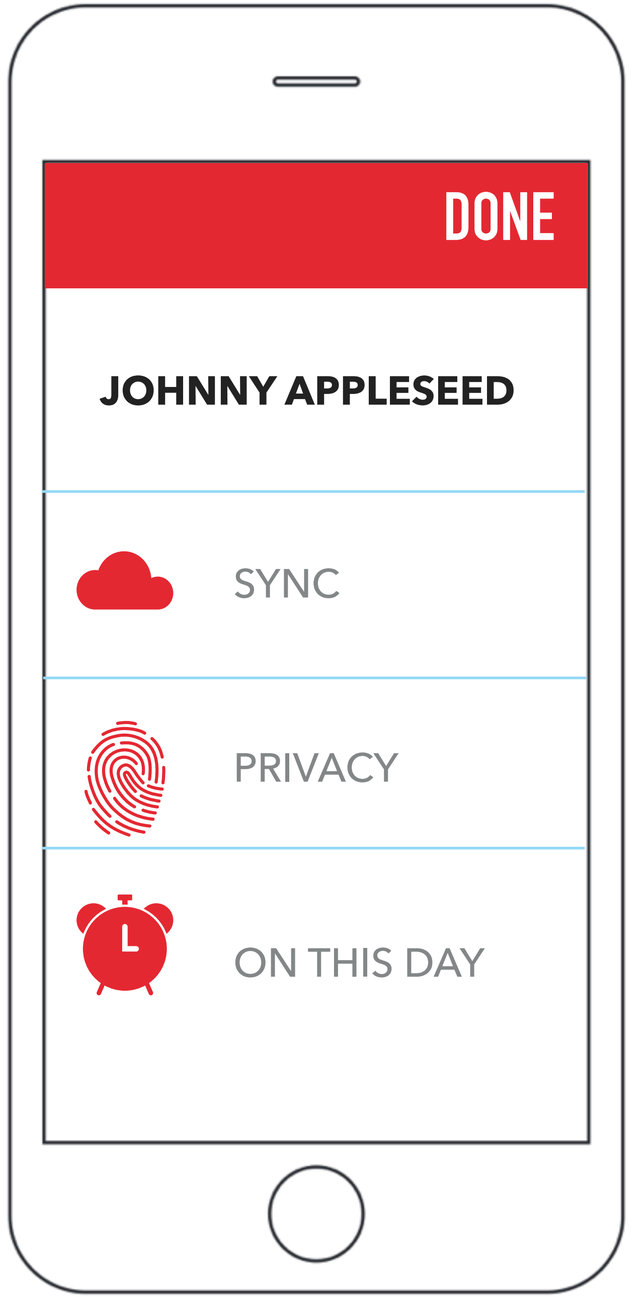
- User is in app and selects add a new entry
  - Entry has a photo and caption text, tags (user generated)
  - Meta-data is automatically generated
    - GPS, Date: From Photos framework
    - Weather: https://darksky.net/dev
    - Tags: Vision, NLP framework
  - Stored as a custom struct and serialized to disk as JSON
  - Everything can be edited (including date)



ASSIGNMENT 3

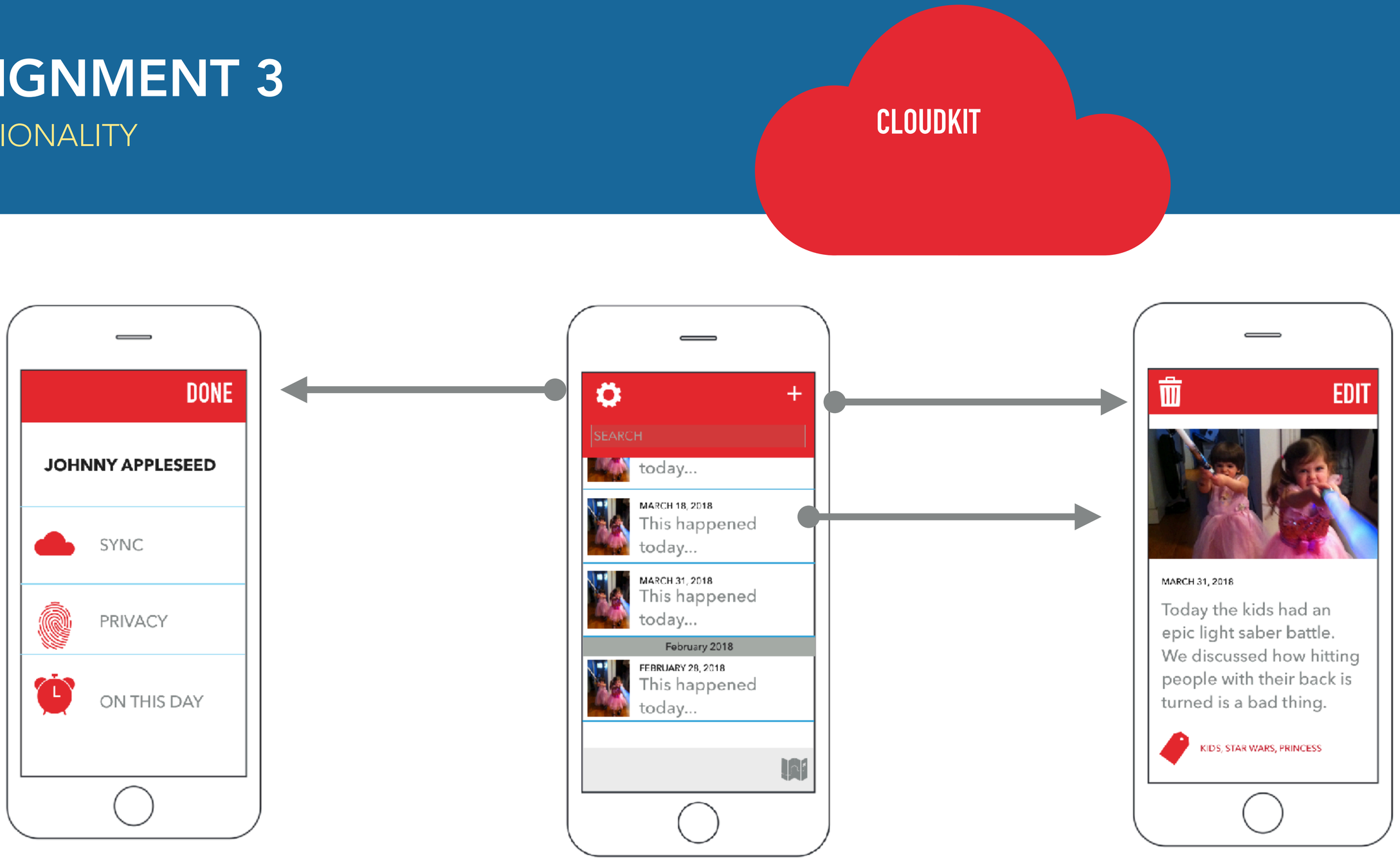
SETTINGS SCREEN

- Settings screen
  - Show iCloud user name
  - Allow user Sync (reset local cache); provide good feedback
  - Set a passcode
  - Bonus Point (on this day notification)



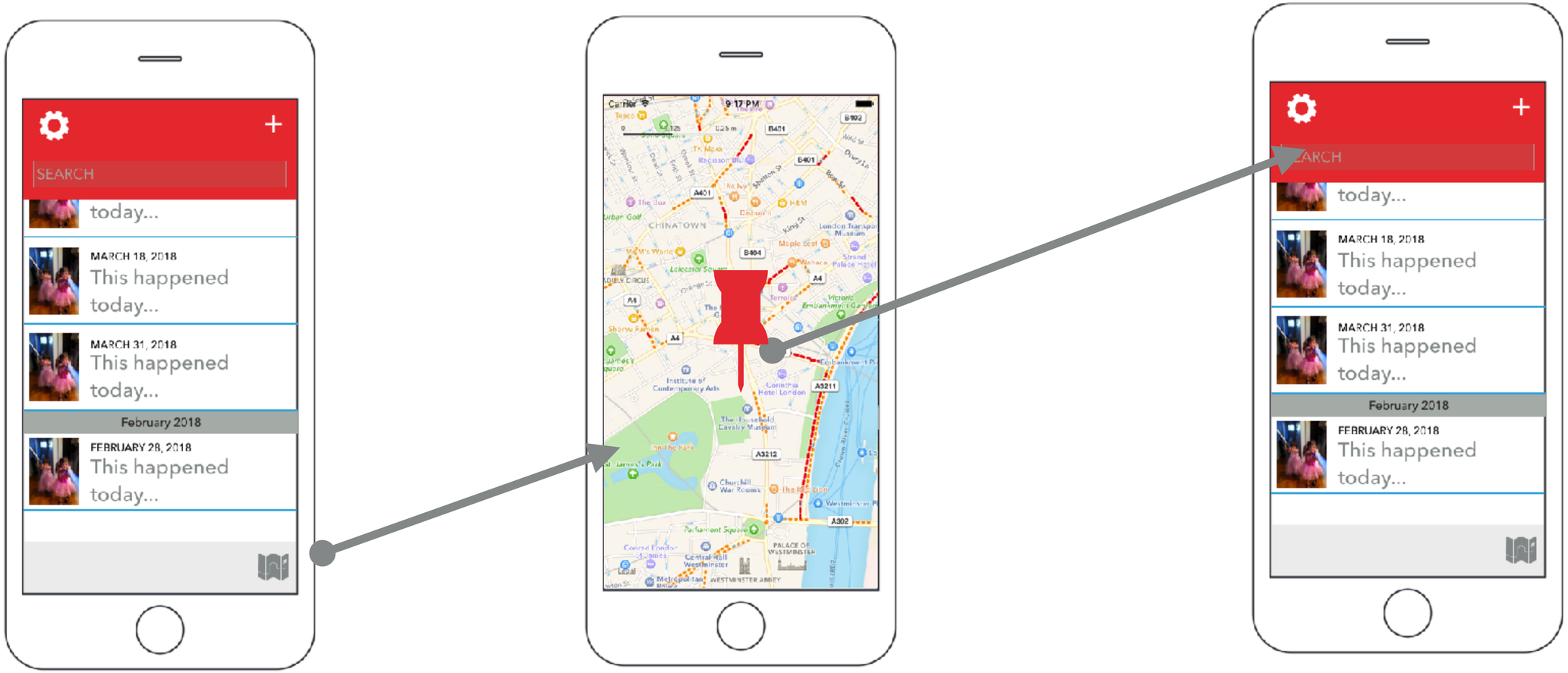
ASSIGNMENT 3

FUNCTIONALITY



ASSIGNMENT 3

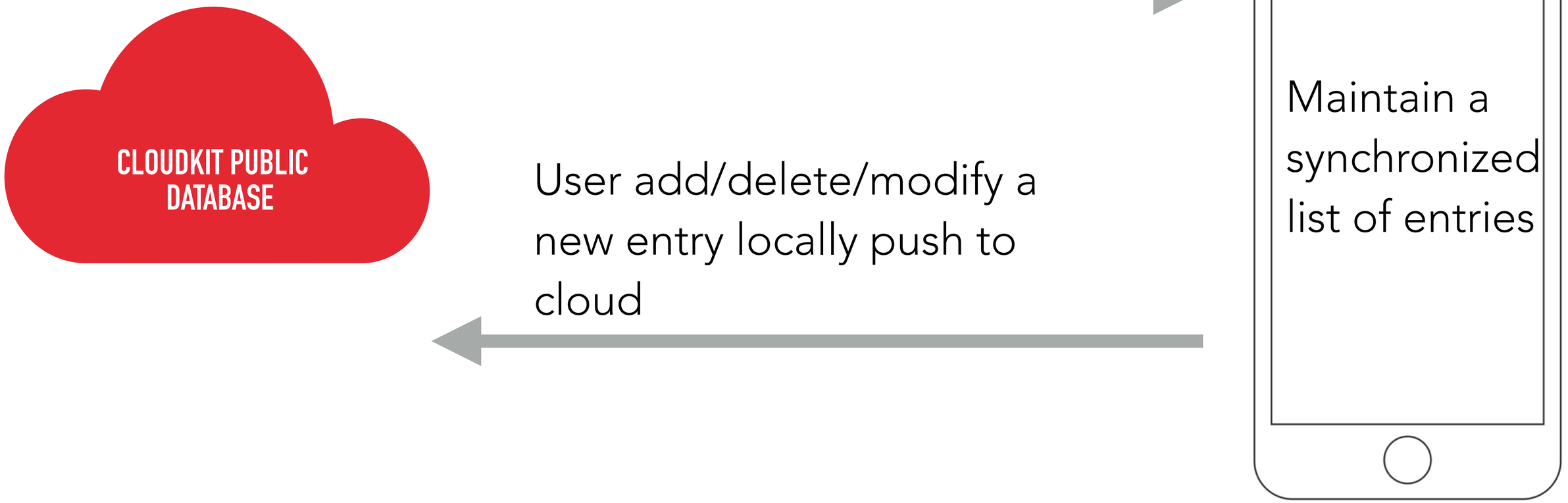
FUNCTIONALITY



ASSIGNMENT 3

- Sync is a hard problem
  - Local caching
  - Mirroring
  - Platform services (Firebase, Realm, not CloudKit)
- We are going to use a simple and free strategy using CloudKit public database

ASSIGNMENT 3



ASSIGNMENT 3

On app launch

- Fetch changes
- Subscribe to future changes

On push from CloudKit

- Fetch changes

ASSIGNMENT 3

OFFLINE-FIRST APPROACH

- Local change
  - Save to local data
  - Mark `synced = false`
  - Update UI (for user's sake)
  - Push to CloudKit
  - On success `synced = true`
  - Save the cloudKit `recordId` to local object

If not successful try again or put in queue to try later

ASSIGNMENT 3

OFFLINE-FIRST APPROACH

- Remote change received via subscription
  - Find local object that matches (recordId)
  - Mark `synced = false`
  - Update local object to match changes
  - Update UI (for user's sake)
  - On success `synced = true`

If not successful try again or put in queue to try later

ASSIGNMENT 3

SYNC FLOW

- Assumptions that make our simple sync approach plausible
  - Journal is single user (not many potential conflicts)
  - Sync data is just text (don't request images until we need them)
  - We can restore the database in a reasonable amount of time
  - Probably not that much data (~1 entry a day)

ASSIGNMENT 3

IMAGES

- Cache images
  - NSCache or Cache directory
  - They will be on iCloud if needed
- Ideas for performance
  - Resize images
  - Create thumbnail cache for table

Only request assets when you need them and then always cache

HTTPS://BLOG.FROZENFIRESTUDIOS.COM/WORKING-WITH-IMAGES-IN-CLOUDKIT-1E3579C67558

RECOMMENDED WORK FOR THIS WEEK

- Entry input, persist, and edit
  - Select photo(s); metadata extraction
  - Input text
  - Input tags
- Sync with CloudKit
  - Subscriptions
  - Dirty flag
- Notifications
- TableView screen
  - Searchbar with query
  - Section headers
  - Map point to search

GOING OVER NEXT WEEK

- Extensions
  - Today
  - Share
- Embedded Frameworks
- App Groups

GOING OVER NEXT WEEK

- 3D Touch
- Vision Framework
- Background Tasks

THE UNIVERSITY OF CHICAGO

ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 3