THE UNIVERSITY OF CHICAGO

ADVANCED iOS APPLICATION DEVELOPMENT

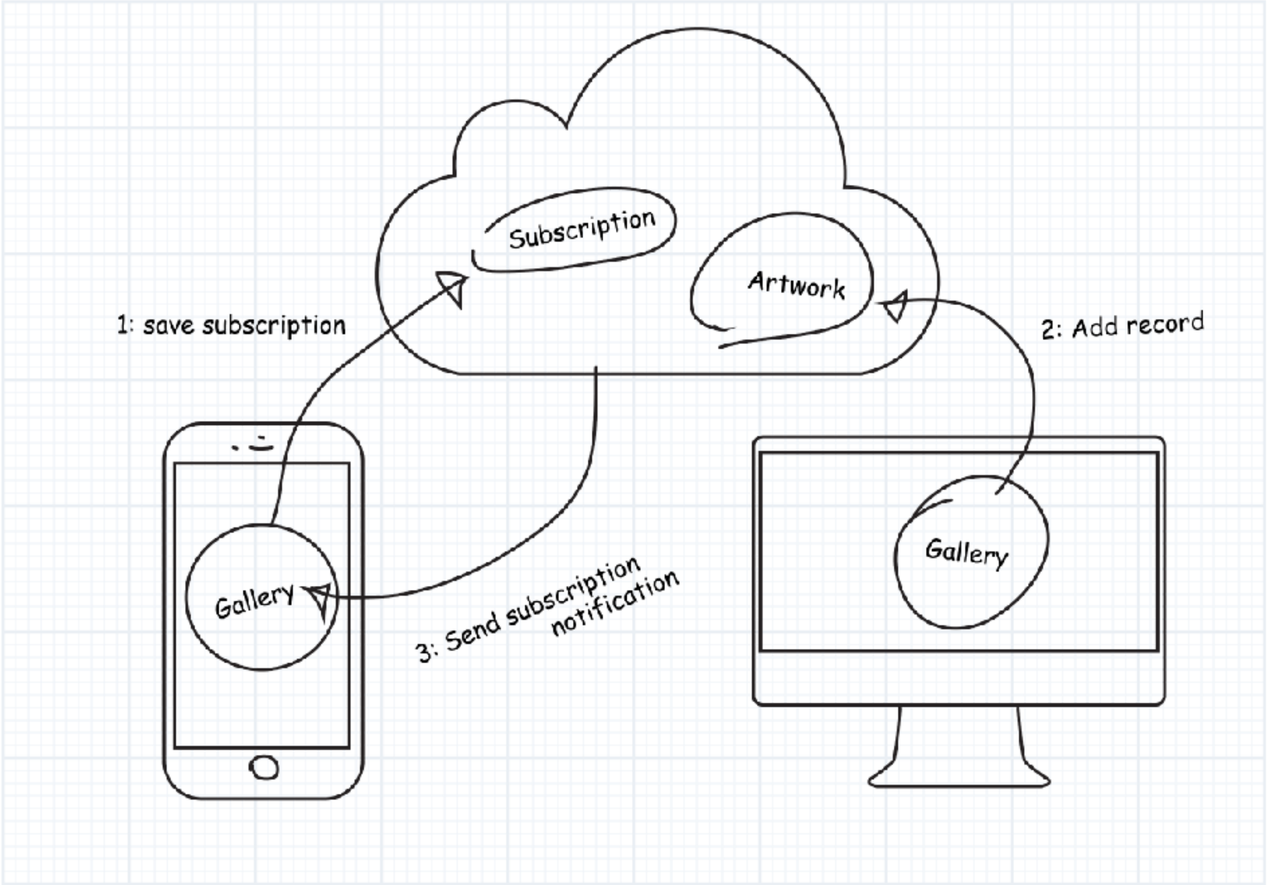
MPCS 51032 • SPRING 2020 • SESSION 3

SUBSCRIPTIONS

SUBSCRIPTIONS

SUBTITLE

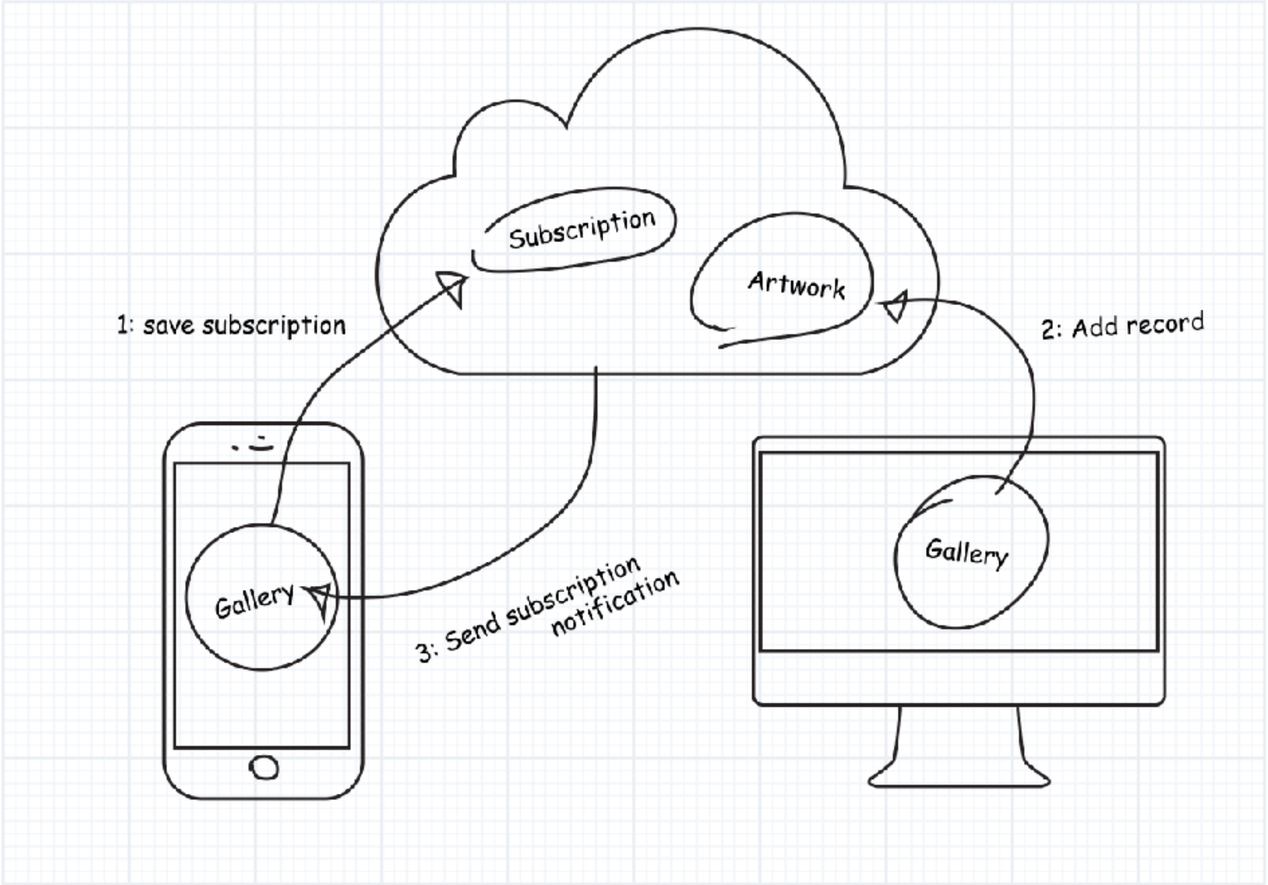
- Queries are polls
- Great for slicing through large server data
- Bad for large, mostly static data set
  - Battery life
  - Networking traffic
  - User experience



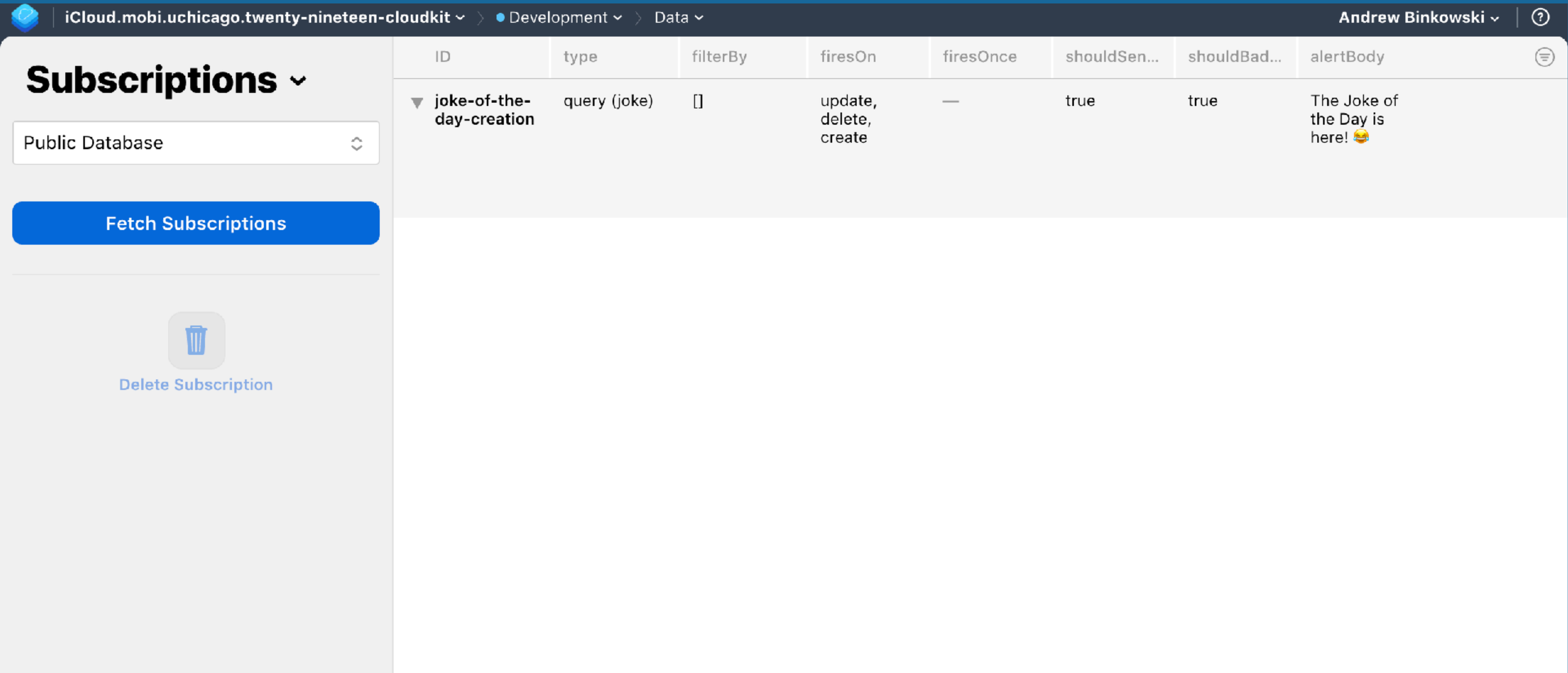
SUBSCRIPTIONS

SUBTITLE

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

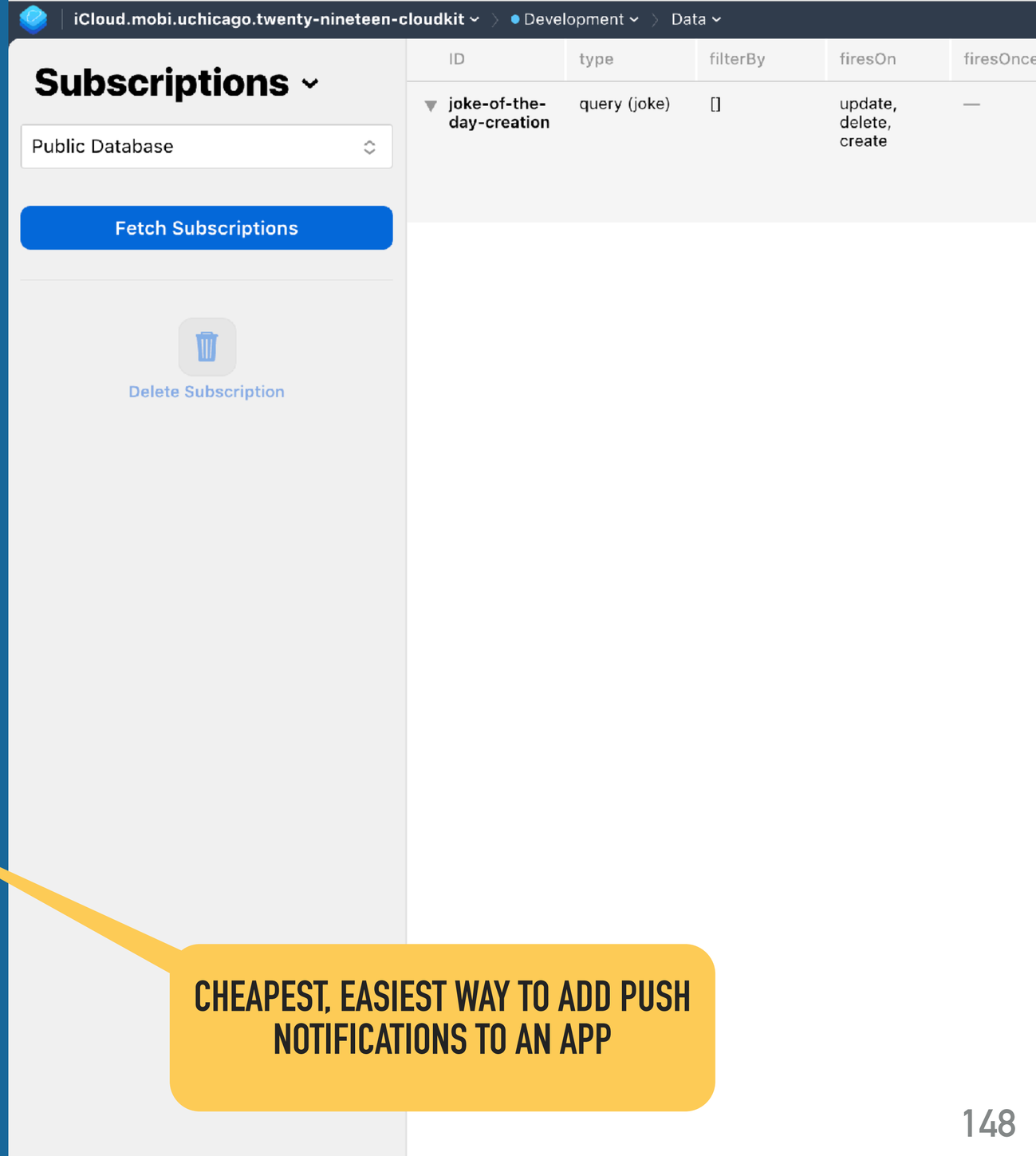


SUBSCRIPTIONS



SUBSCRIPTIONS

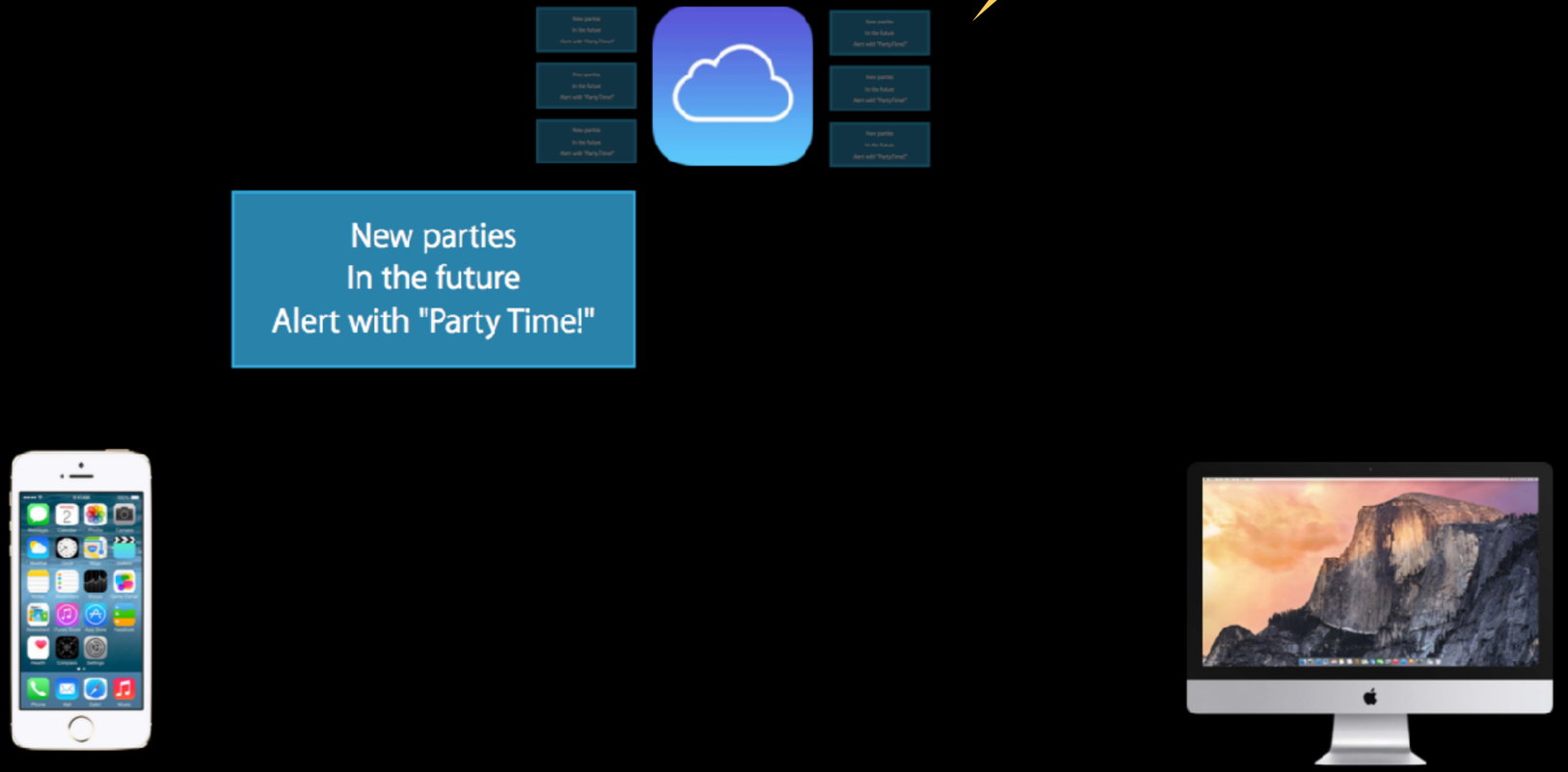
- CKQuerySubscription
  - RecordType
  - NSPredicate
  - Push
- Push delivered via Apple Push Notification Service (APNS) augmented payload
  - JSON



CHEAPEST, EASIEST WAY TO ADD PUSH NOTIFICATIONS TO AN APP

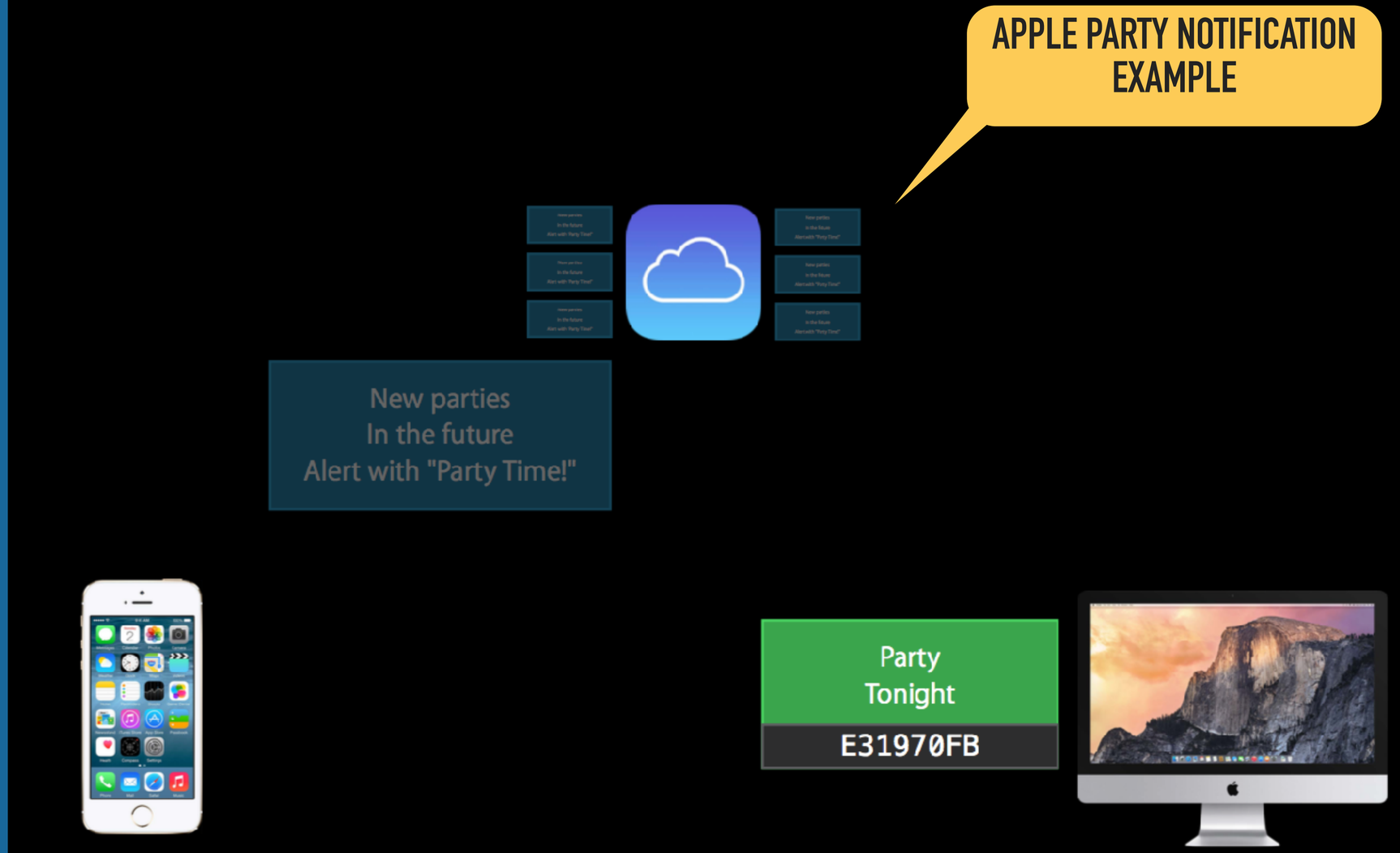
SUBSCRIPTIONS

- Subscription in action



SUBSCRIPTIONS

- Subscription in action

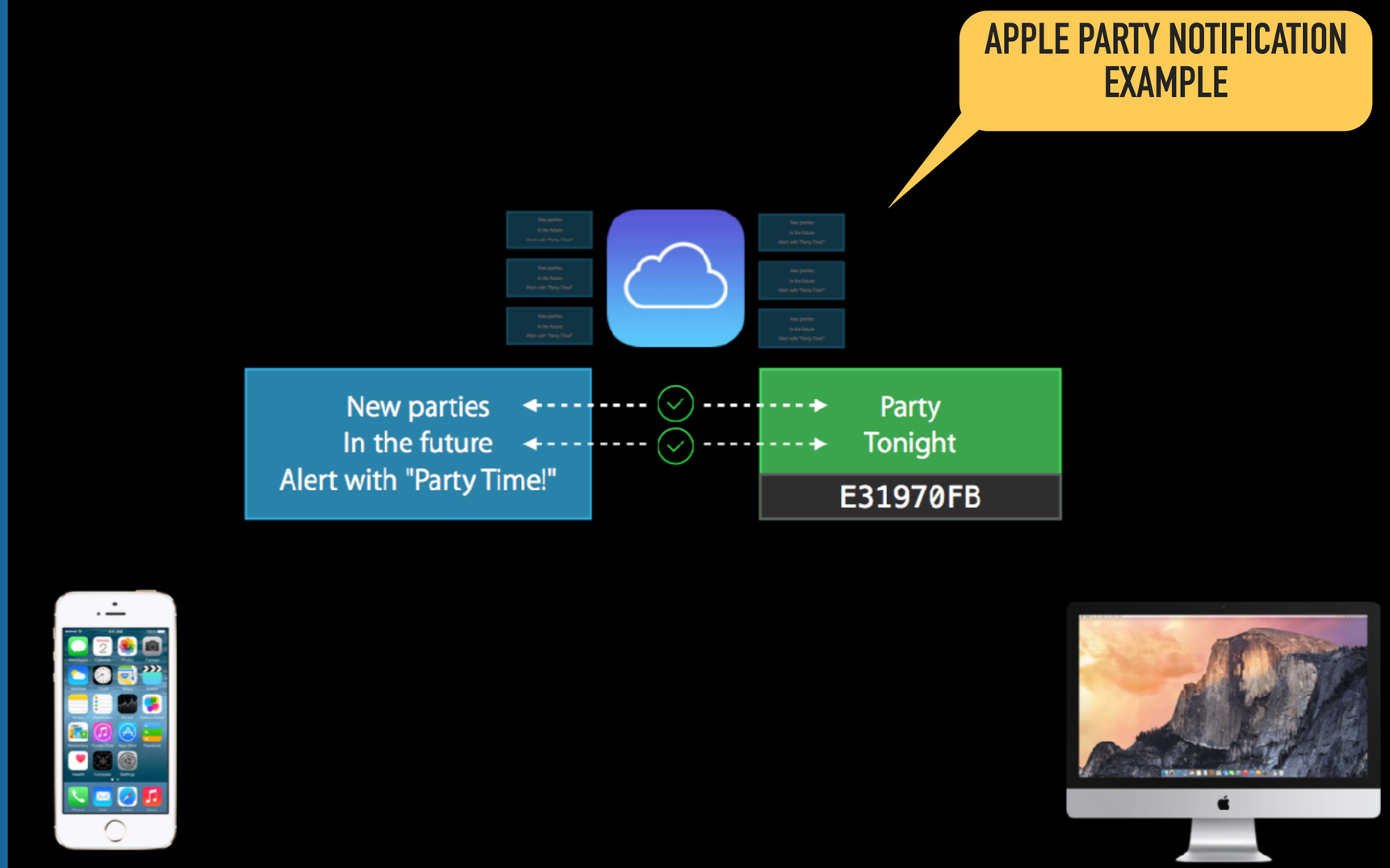


SUBSCRIPTIONS

- Subscription in action

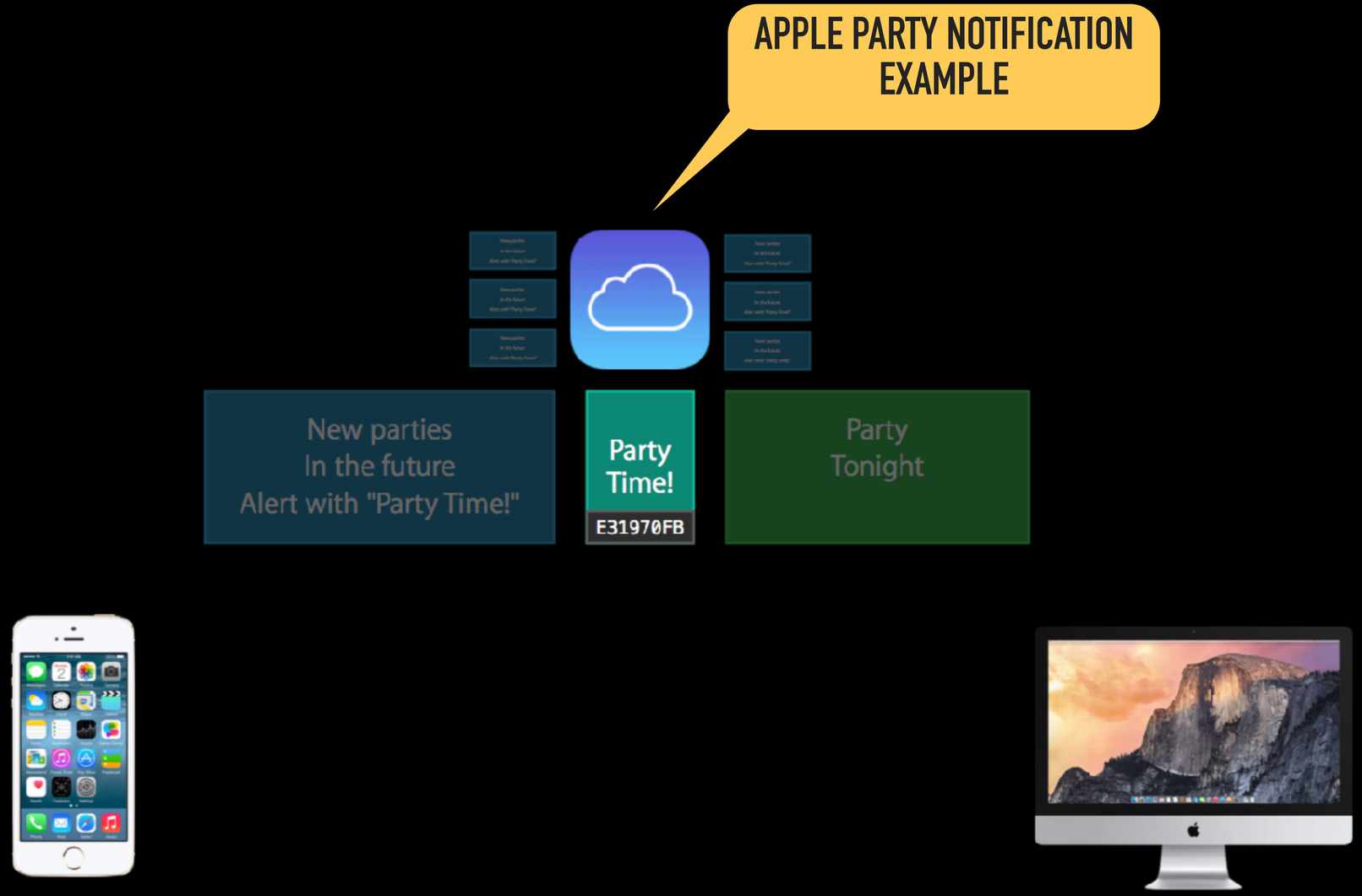
SUBSCRIPTIONS

- Subscription in action



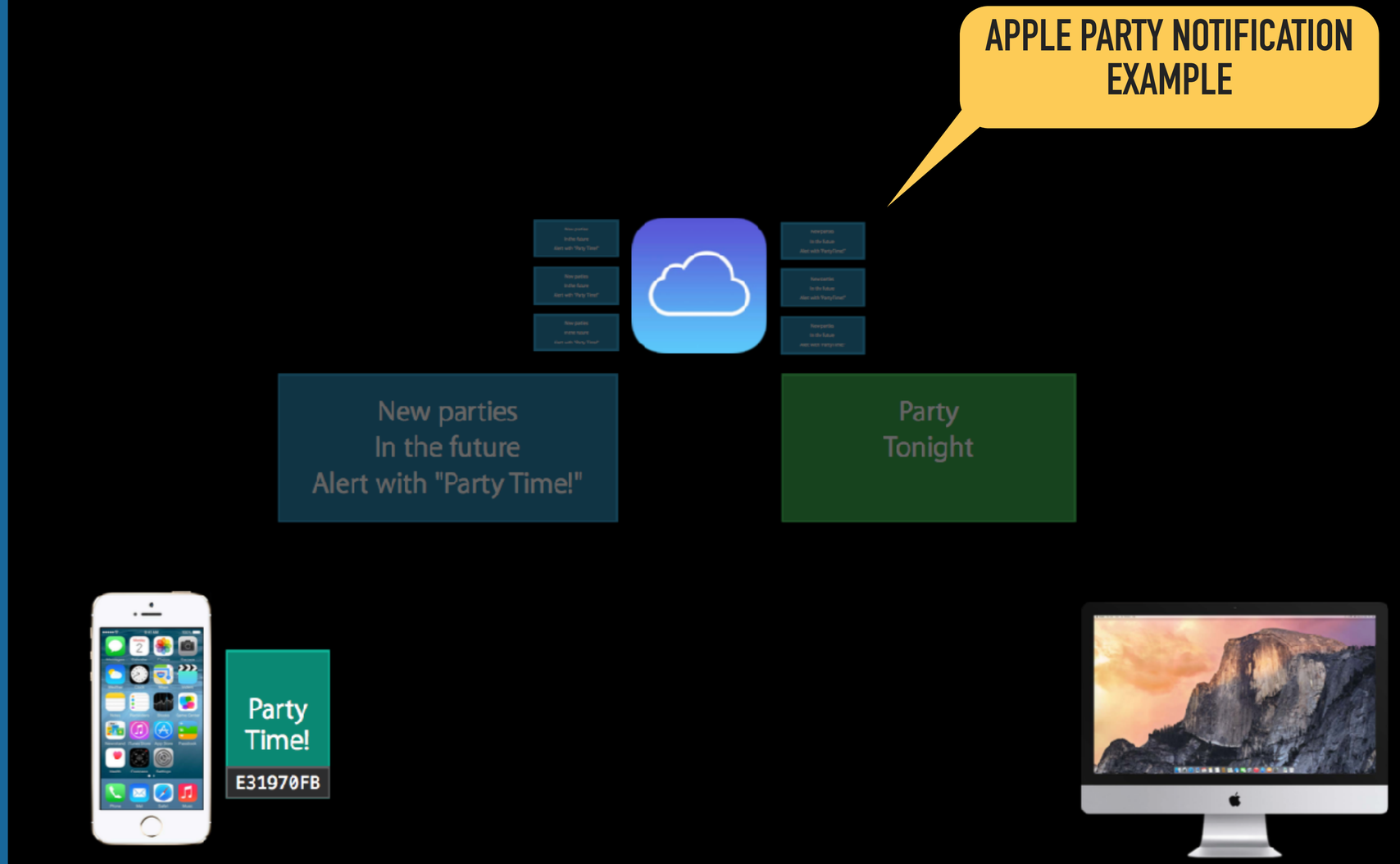
SUBSCRIPTIONS

- Subscription in action



SUBSCRIPTIONS

- Subscription in action



SUBSCRIPTIONS

- Subscriptions are user based
- Require a unique id

```
func registerJokeOfTheDaySubscriptions() {
  // Unique identifier for the subscription
  let identifier = "joke-of-the-day-creation"

  // Create the notification that will be delivered
  let notificationInfo = CKSubscription.NotificationInfo()
  notificationInfo.alertBody = "The Joke of the Day is here! 😂"
  notificationInfo.shouldBadge = true
  notificationInfo.shouldSendContentAvailable = true
  //notificationInfo.desiredKeys = ["joke"]

  // Create the subscription object
  let subscription = CKQuerySubscription(recordType: "joke",
                                         predicate: NSPredicate(value: true),
                                         subscriptionID: identifier,
                                         options: [
                                          CKQuerySubscription.Options.firesOnRecordCre
                                          CKQuerySubscription.Options.firesOnRecordUpd
                                          CKQuerySubscription.Options.firesOnRecordDel

  subscription.notificationInfo = notificationInfo

  // Save subscription
  CKContainer.default().publicCloudDatabase.save(subscription, completionHandler: ({re
    if let err = error {
      print("JOTD: subscription failed \(err.localizedDescription)")
    } else {
      print("JOTD: subscription set up")
    }
  }))
}
```

SUBSCRIPTIONS

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

```
func registerJokeOfTheDaySubscriptions() {
  // Unique identifier for the subscription
  let uuid: UUID = UUID()
  let identifier = "\(uuid)-joke-of-the-day-any-key"

  // Create the notification that will be delivered
  let notificationInfo = CKSubscription.NotificationInfo()
  notificationInfo.alertBody = "The Joke of the Day is here! 😂"
  notificationInfo.shouldBadge = true
  notificationInfo.shouldSendContentAvailable = true
  notificationInfo.desiredKeys = ["question"]

  // Create the subscription object
  let subscription = CKQuerySubscription(recordType: "joke",
                                         predicate: NSPredicate(value: true),
                                         subscriptionID: identifier,
                                         options: [
                                          CKQuerySubscription.Options.firesOnRecordCr

  subscription.notificationInfo = notificationInfo

  // Save subscription
  currentDB.save(subscription, completionHandler: ({returnRecord, error in
    if let err = error {
      print("JOTD: subscription failed \(err.localizedDescription)")
    } else {
      print("JOTD: subscription set up")
    }
  }))
}
```

SUBSCRIPTIONS

- Create a subscription

```
func registerJokeOfTheDaySubscriptions() {
  // Unique identifier for the subscription
  let uuid: UUID = UUID()
  let identifier = "\(uuid)-joke-of-the-day-any-key"

  // Create the notification that will be delivered
  let notificationInfo = CKSubscription.NotificationInfo()
  notificationInfo.alertBody = "The Joke of the Day is here! 😂"
  notificationInfo.shouldBadge = true
  notificationInfo.shouldSendContentAvailable = true
  notificationInfo.desiredKeys = ["question"]

  // Create the subscription object
  let subscription = CKQuerySubscription(recordType: "joke",
                                         predicate: NSPredicate(value: true),
                                         subscriptionID: identifier,
                                         options: [
                                          CKQuerySubscription.Options.firesOnRecordCre

  subscription.notificationInfo = notificationInfo

  // Save subscription
  currentDB.save(subscription, completionHandler: ({returnRecord, error in
    if let err = error {
      print("JOTD: subscription failed \(err.localizedDescription)")
    } else {
      print("JOTD: subscription set up")
    }
  }))
}
```

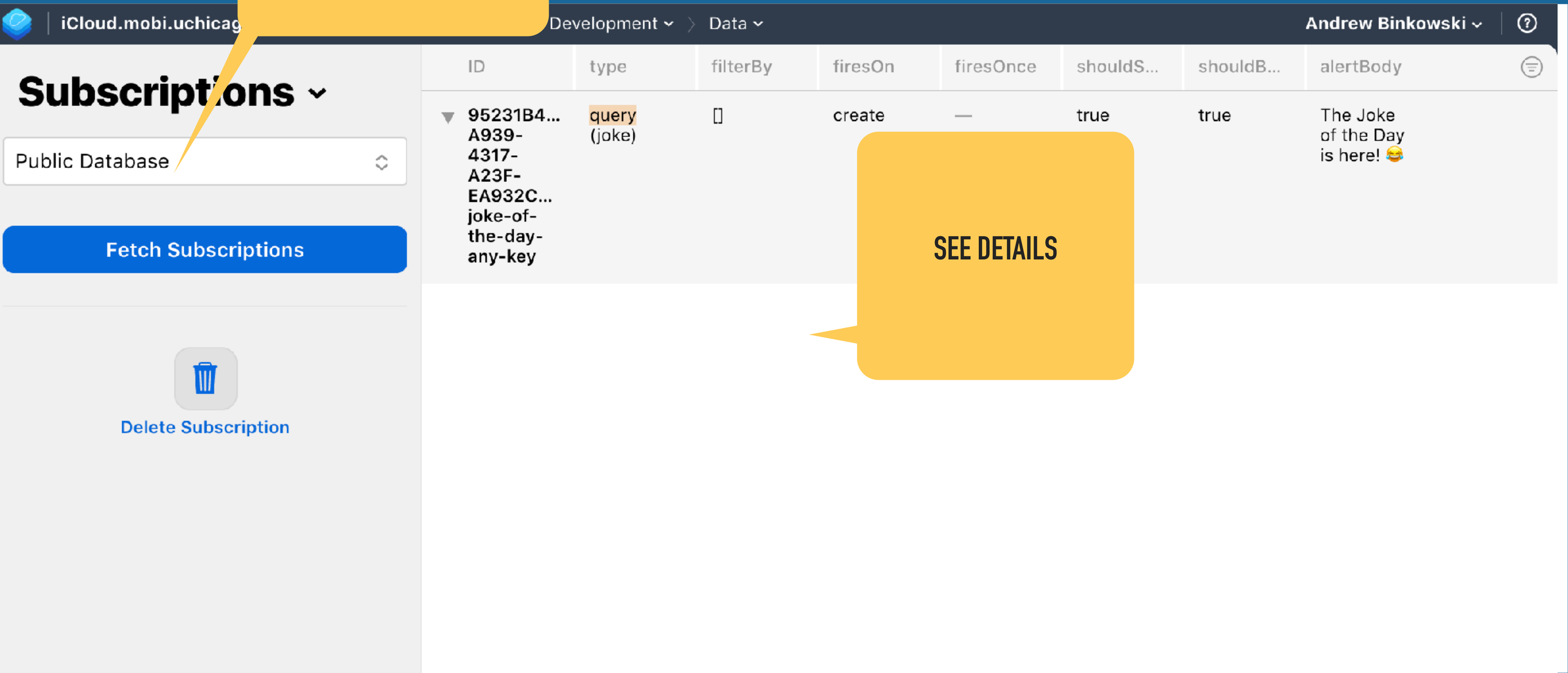
SUBSCRIPTIONS

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

```
func registerJokeOfTheDaySubscriptions() {
  // Unique identifier for the subscription
  let uuid: UUID = UUID()
  let identifier = "\(uuid)-joke-of-the-day-any-key"

  // Create the notification that will be delivered
  let notificationInfo = CKSubscription.NotificationInfo()
  notificationInfo.alertBody = "The Joke of the Day is here! 😂"
  notificationInfo.shouldBadge = true
  notificationInfo.shouldSendContentAvailable = true
  notificationInfo.desiredKeys = ["question"]

  // Create the subscription object
  let subscription = CKQuerySubscription(recordType: "joke",
                                         predicate: NSPredicate(value: true),
                                         subscriptionID: identifier,
                                         options: [
                                          CKQuerySubscription.Options.firesOnRecordCre

  subscription.notificationInfo = notificationInfo

  // Save subscription
  currentDB.save(subscription, completionHandler: ({returnRecord, error in
    if let err = error {
      print("JOTD: subscription failed \(err.localizedDescription)")
    } else {
      print("JOTD: subscription set up")
    }
  }))
}
```

SUBSCRIPTIONS

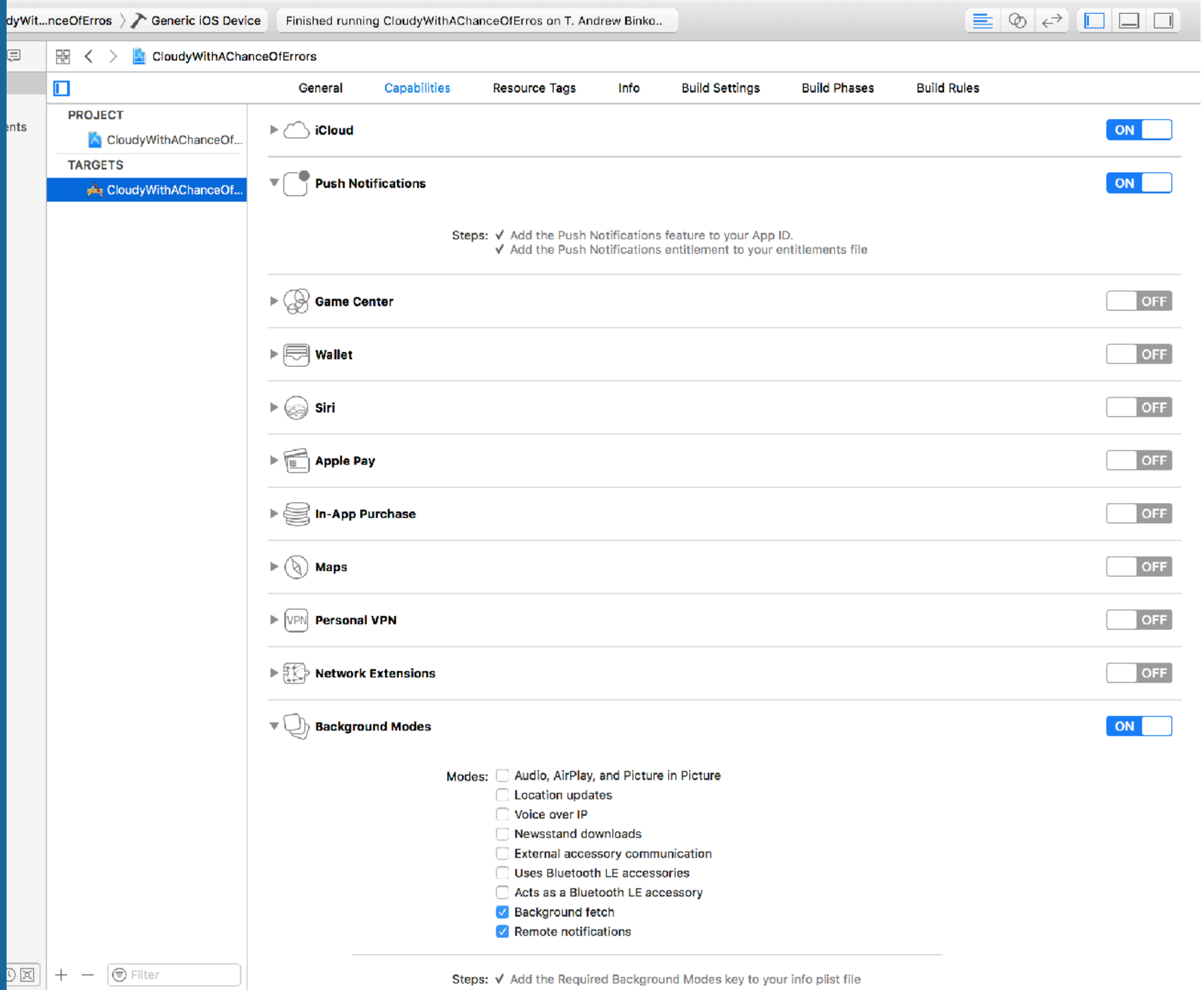
SHOW ALL SUBSCRIPTIONS



SETUP YOUR APP TO HANDLE NOTIFICATIONS

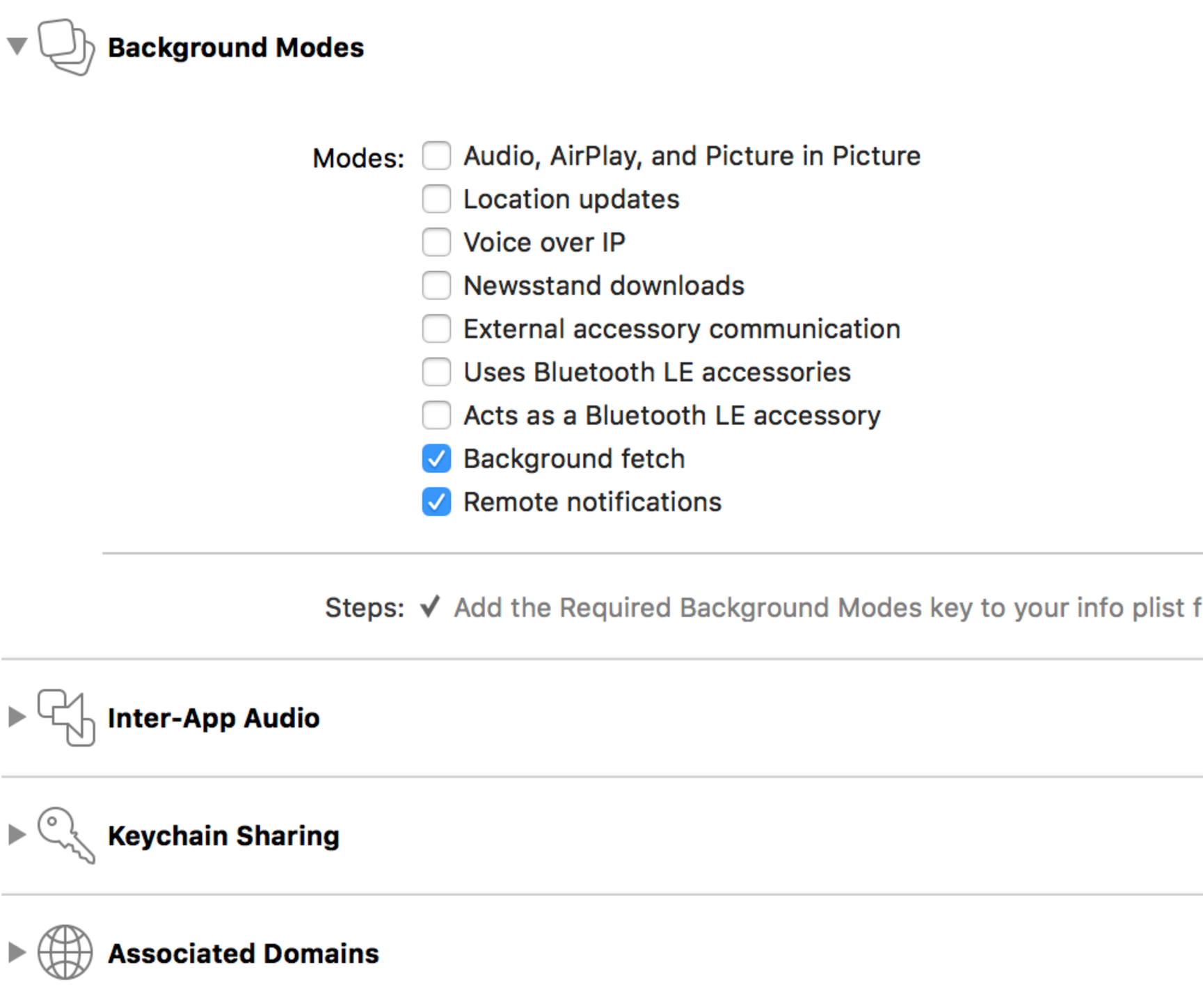
SUBSCRIPTIONS

- Required capabilities
  - Push notifications



SUBSCRIPTIONS

- Required capabilities
  - Remote notification
  - Background fetch



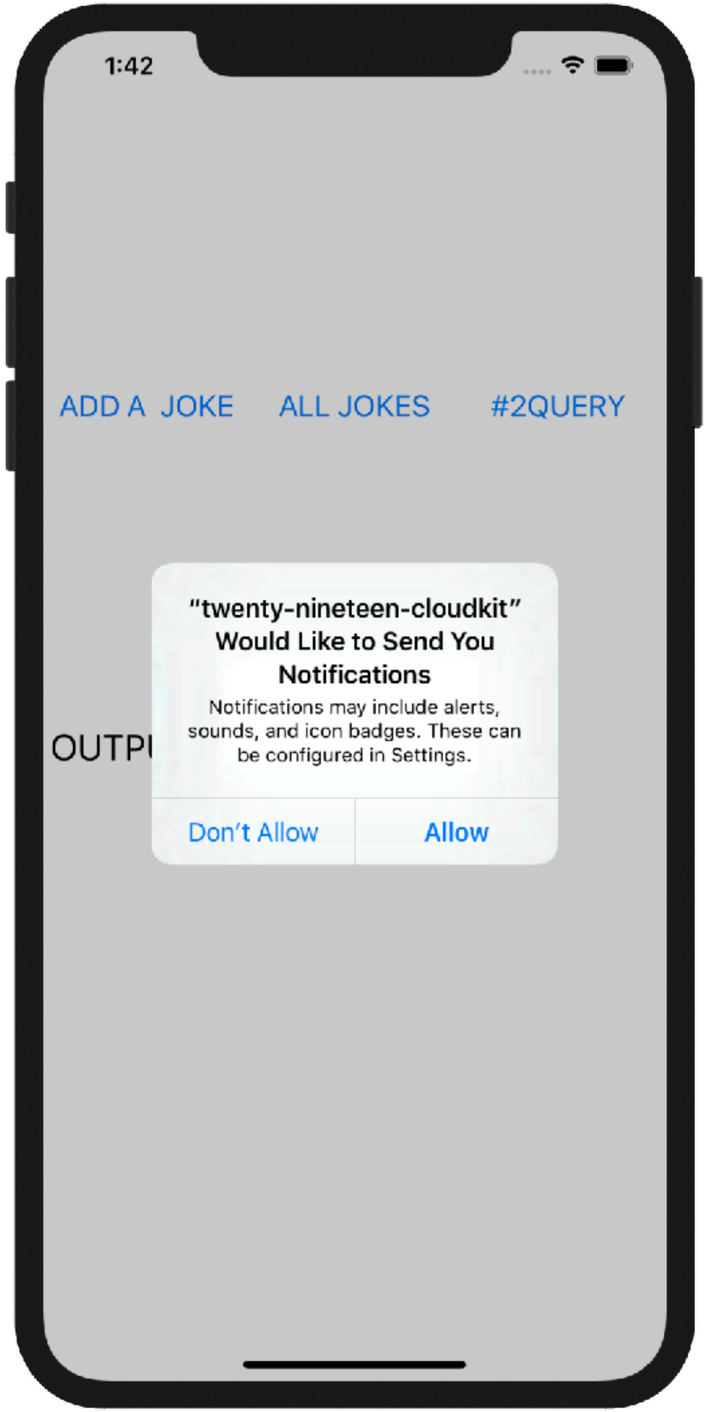
SUBSCRIPTIONS

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                    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
}
```

REGISTER SUBSCRIPTIONS

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")
    }
  }))
}
```

HANDLE NOTIFICATIONS (THERE ARE 3 WAYS)

SUBSCRIPTIONS

- Handle notifications depending on what you want to do
  - Do nothing (just show notifications)
  - Receive information about what has changed

```
func application(_ applicati
                 didFinishLa

  // Set the notification de
  UNUserNotificationCenter.c
    if let error = error {
      print("Error: \(error
    } else {
      application.registerFo
    }
  }
  UNUserNotificationCenter.c

  // Register subscriptions
  CloudKitManager.sharedInst
  CloudKitManager.sharedInst
  //configureUserNotificatio

  if let notification = laun
    let aps = notification['
    print("APS: \(aps)")
  }
  return true
}
```

SUBSCRIPTIONS

- Do nothing
- Handle CloudKit notifications in two places (depending on the state of the app)
  - applicationsDidFinishLaunching
  - didReceiveRemoteNotification

```
func application(_ applicati
                 didFinishLa

  // Set the notification de
  UNUserNotificationCenter.c
    if let error = error {
      print("Error: \(error
    } else {
      application.registerFo
    }
  }
  UNUserNotificationCenter.c

  // Register subscriptions
  CloudKitManager.sharedInst
  CloudKitManager.sharedInst
  //configureUserNotificatio

  if let notification = laun
    let aps = notification['
    print("APS: \(aps)")
  }
  return true
}
```

SUBSCRIPTIONS

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

FRESH LAUNCH
(RARELY HAPPENS)

```
func application(_ applicati
                 didFinishLa

                 notification de
                 icationCenter.
                 ror = error {
                 Error: \(error

                 tion.registerFo

  }
  UNUserNotificationCenter.

  // Register subscriptions
  CloudKitManager.sharedInst
  CloudKitManager.sharedInst
  //configureUserNotificatio

  if let notification = laun
    let aps = notification['
    print("APS: \(aps)")
  }
  return true
}
```

SUBSCRIPTIONS

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


```
func application(_ applicati
                 didFinishLa

  // Set the notification de
  UNUserNotificationCenter.c
    if let error = error {
      print("Error: \(error
    } else {
      lication.registerFo


      otificationCenter.c

      ster subscriptions
  CloudKitManager.sharedInst
  CloudKitManager.sharedInst
  //configureUserNotificatio

  if let notification = laun
    let aps = notification['
    print("APS: \(aps)")
  }
  return true
}
```

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 when receiving notifications

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
    [UIApplication.LaunchOptionsKey: Any]?) -> Bool {

  // Register a subscription for this device
  CloudKitManager.sharedInstance.registerJokeOfTheDaySubscriptions()

  // Request authorization for notifications
  UNUserNotificationCenter.current().requestAuthorization(options: [.alert, .sound, .badge])
      granted, error in
    if let error = error {
      print("D'oh: \(error.localizedDescription)")
    } else {
      DispatchQueue.main.async {
        application.registerForRemoteNotifications()
      }
    }
  }

  // Set delegate to receive notifications in all cases
  UNUserNotificationCenter.current().delegate = self


  // Parse the launch notification so that we can get the payload
  if let notification = launchOptions?[UIApplication.LaunchOptionsKey.remoteNotification] as
      [String: AnyObject] {
    let aps = notification["aps"] as! [String: AnyObject]
    print("APS: \(aps)")
  }


  return true
}
```

SUBSCRIPTIONS

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

```
func registerJokeOfTheDaySubscriptions() {
  // Unique identifier for the subscription
  let uuid: UUID = UUID()
  let identifier = "\(uuid)-joke-of-the-day-any-key"

  // Create the notification that will be delivered
  let notificationInfo = CKSubscription.NotificationInfo()
  notificationInfo.alertBody = "The Joke of the Day is here! 😂"
  notificationInfo.shouldBadge = true
  notificationInfo.shouldSendContentAvailable = true
  notificationInfo.desiredKeys = ["question"]

  // Create the subscription object
  let subscription = CKQuerySubscription(recordType: "joke",
                                         predicate: NSPredicate(value: true),
                                         subscriptionID: identifier,
                                         options: [
                                          CKQuerySubscription.Options.firesOnRecordCre

  subscription.notificationInfo = notificationInfo

  // Save subscription
  currentDB.save(subscription, completionHandler: ({returnRecord, error in
    if let err = error {
      print("JOTD: subscription failed \(err.localizedDescription)")
    } else {
      print("JOTD: subscription set up")
    }
  }))
}
```

SUBSCRIPTIONS

```
[AnyHashable("aps"): {
    alert = "The Joke of the Day is here! \Ud83d\Ude02";
    badge = 39;
    "content-available" = 1;
}, AnyHashable("ck"): {
    ce = 2;
    cid = "iCloud.mobi.uchicago.twenty-nineteen-cloudkit";
    ckuserid = "_c364d6b1fb322f17cd2346d1b82667a2";
    nid = "90a87393-1eb3-403c-bff9-73ad00c44820";
    qry =     {
        dbs = 2;
        fo = 1;
        rid = "DCE4A6EF-A6A0-47C9-AE60-4022E13D511B";
        sid = "joke-of-the-day-creation";
        zid = "_defaultZone";
        zoid = "_defaultOwner";
    };
}]
```

SUBSCRIPTIONS

```
[AnyHashable("aps"): {
    alert = "The Joke of the Day is here! \Ud83d\Ude02";
    badge = 39;
    "content-available" = 1;
}, AnyHashable("ck"): {
    ce = 2;
    cid = "iCloud.mobi.uchicago.twenty-nineteen-cloudkit";
    ckuserid = "_c364d6b1fb322f17cd2346d1b82667a2";
    nid = "90a87393-1eb3-403c-bff9-73ad00c44820";
    qry =     {
        dbs = 2;
        fo = 1;
        rid = "DCE4A6EF-A6A0-47C9-AE60-4022E13D511B";
        sid = "joke-of-the-day-creation";
        zid = "_defaultZone";
        zoid = "_defaultOwner";
    };
}]
```

SILENT NOTIFCATIONS

DEBUGGING NOTIFICATION HANDLING

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…nus can retrieve the unhandled ones with `CKFetchNotificationChangesOperation` and process them from there.

HTTPS://DEVELOPER.APPLE.COM/LIBRARY/CONTENT/QA/QA1917/_INDEX.HTML

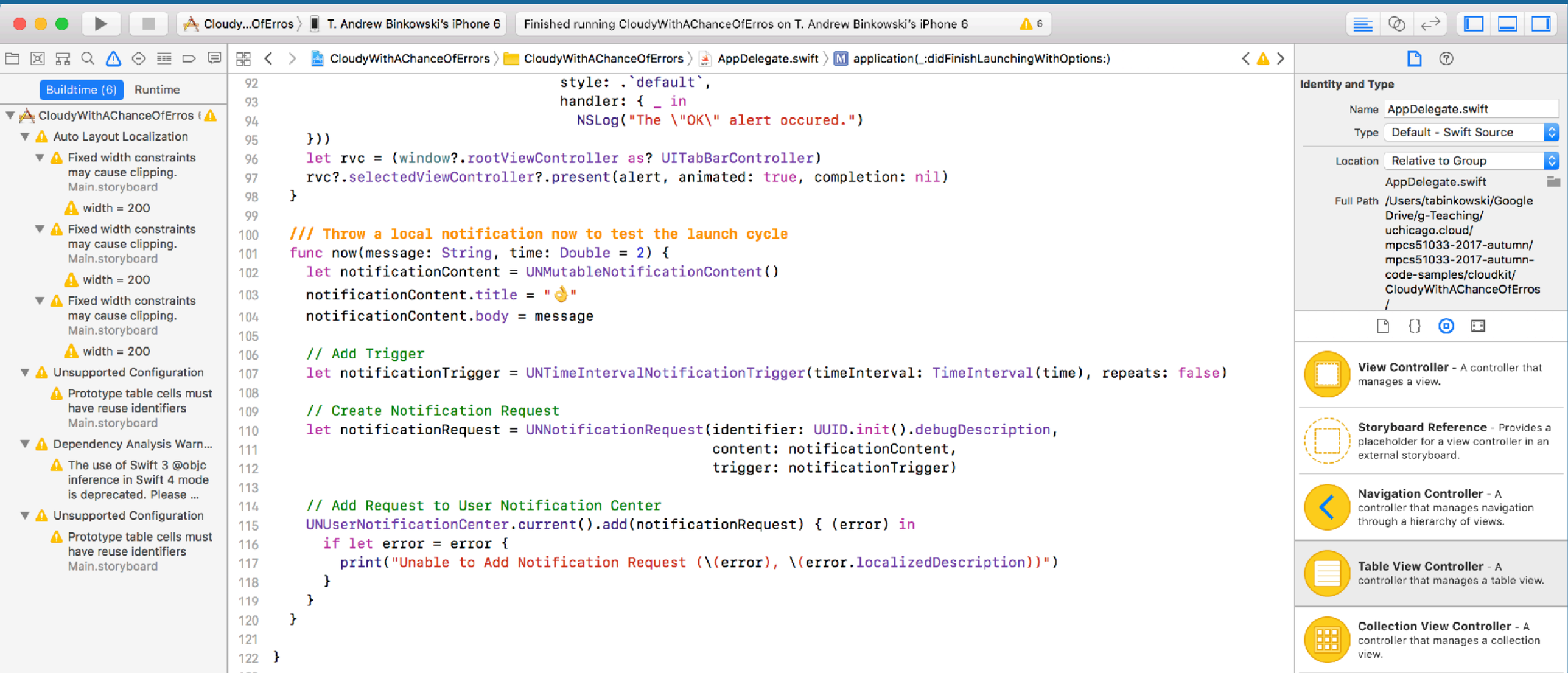
SUBSCRIPTIONS

```
/// 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 occured.")
  }))
  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



WEB SERVICES

WEB SERVICE API

SUBTITLE

- 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" />
```

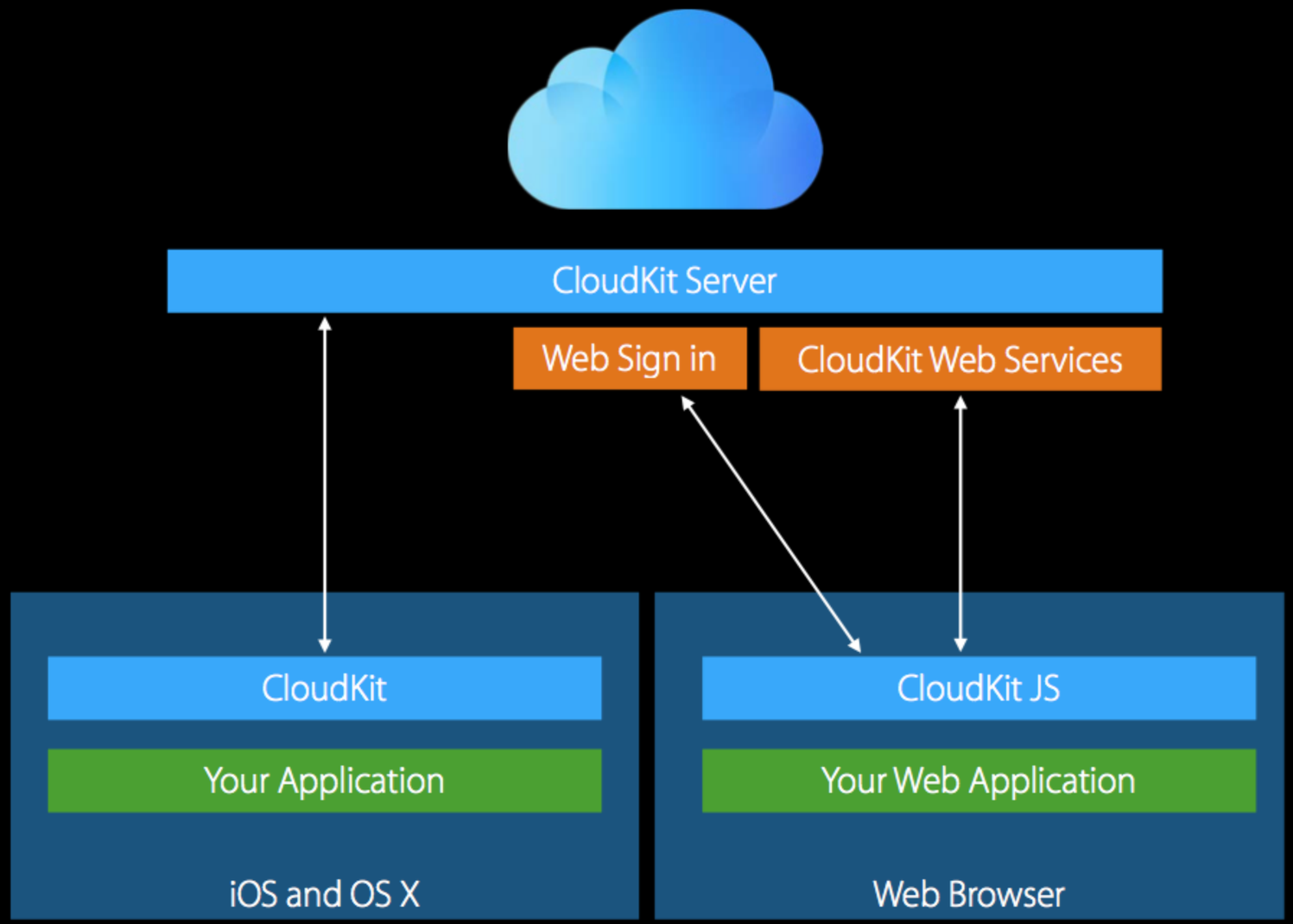
WEB SERVICE API

SUBTITLE

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

WEB SERVICES

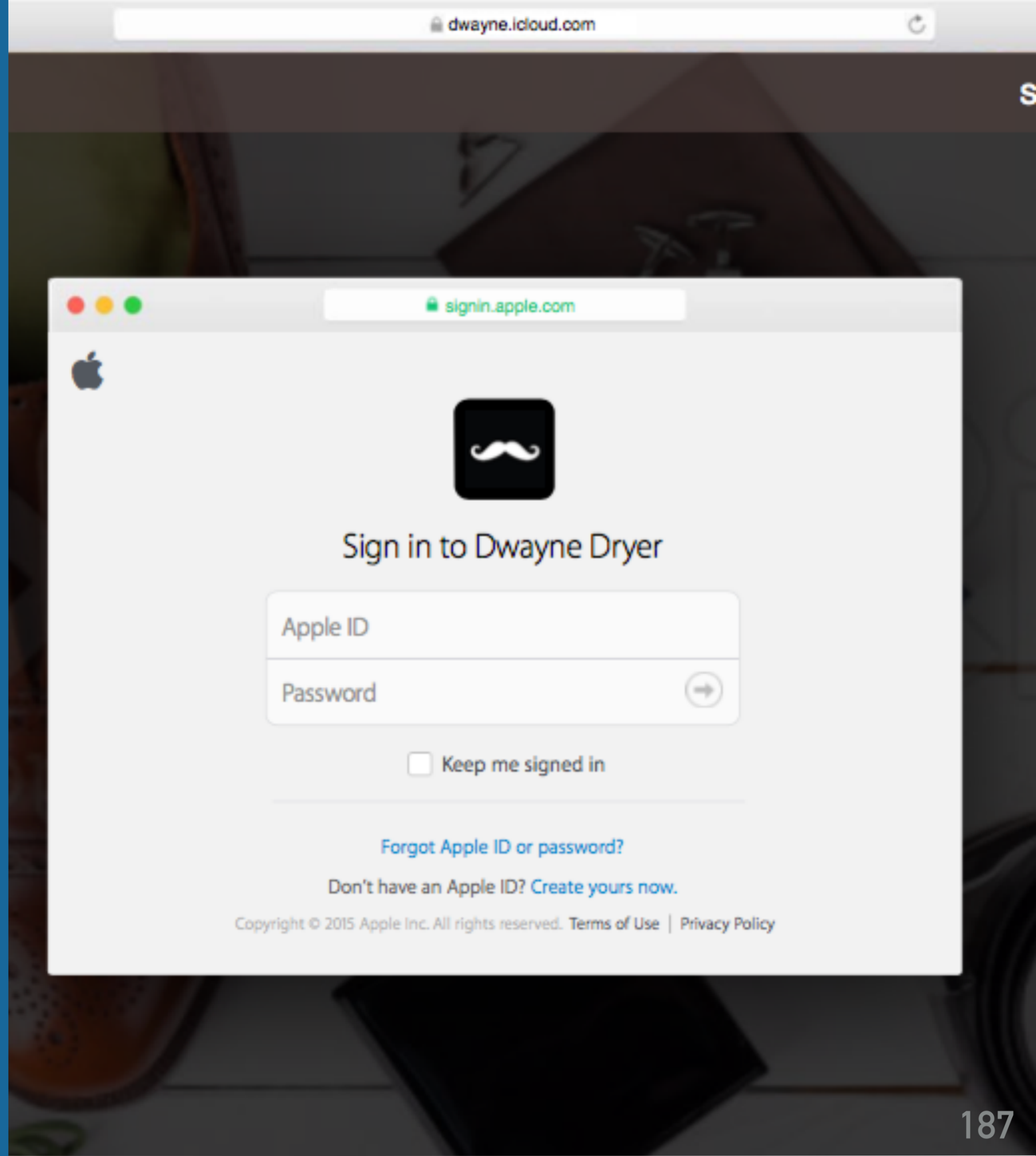
- New server-to-server capabilities
- Add servers to cover the areas that cloud kit doesn't cover



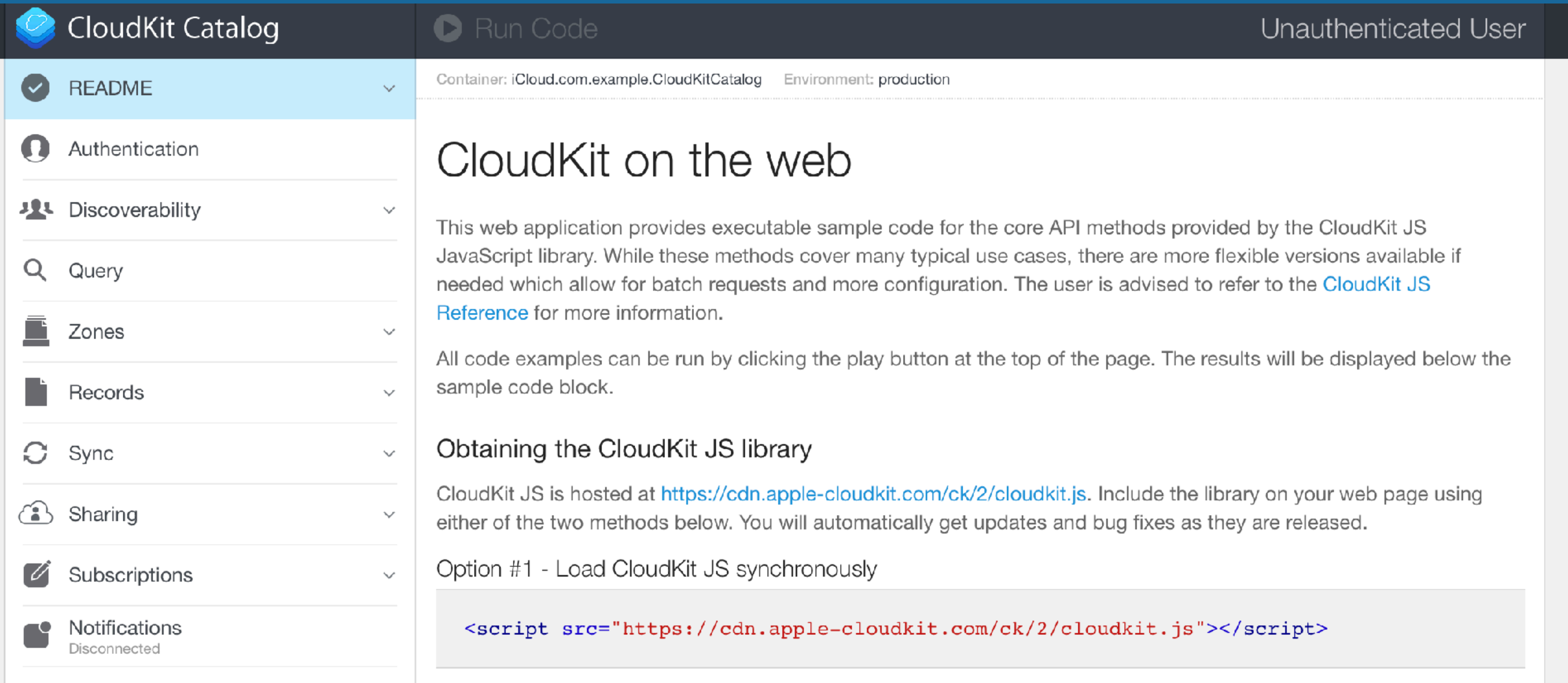
WEB SERVICE API

SUBTITLE

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



WEB SERVICE API



THE UNIVERSITY OF CHICAGO

ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 3