Using Core Data with CloudKit

Nick Gillett, Core Data
Using Core Data with CloudKit
Using Core Data with CloudKit

Everything should sync
Using Core Data with CloudKit

Everything should sync

Sync should be easy
Existing Technologies
Existing Technologies

Core Data provides local persistence
Existing Technologies

Core Data provides local persistence

CloudKit provides distributed persistence
Existing Technologies

Core Data provides local persistence

CloudKit provides distributed persistence

Both exist on all platforms
Existing Technologies

Core Data provides local persistence
CloudKit provides distributed persistence
Both exist on all platforms
Both support a wide variety of applications
Existing Technologies
Existing Technologies

Objects

Models

Stores
## Existing Technologies

<table>
<thead>
<tr>
<th>Objects</th>
<th>Core Data</th>
<th>CloudKit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Models</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stores</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Existing Technologies

<table>
<thead>
<tr>
<th>Objects</th>
<th>Core Data</th>
<th>CloudKit</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSManagedObject</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Models</th>
<th>Core Data</th>
<th>CloudKit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stores</th>
<th>Core Data</th>
<th>CloudKit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Existing Technologies

<table>
<thead>
<tr>
<th></th>
<th>Core Data</th>
<th>CloudKit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objects</td>
<td>NSManagedObject</td>
<td>CKRecord</td>
</tr>
<tr>
<td>Models</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stores</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Existing Technologies

<table>
<thead>
<tr>
<th></th>
<th>Core Data</th>
<th>CloudKit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Objects</strong></td>
<td>NSManagedObject</td>
<td>CKRecord</td>
</tr>
<tr>
<td><strong>Models</strong></td>
<td>NSManagedObjectModel</td>
<td></td>
</tr>
<tr>
<td><strong>Stores</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Existing Technologies

<table>
<thead>
<tr>
<th></th>
<th>Core Data</th>
<th>CloudKit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objects</td>
<td><code>NSManagedObject</code></td>
<td><code>CKRecord</code></td>
</tr>
<tr>
<td>Models</td>
<td><code>NSManagedObjectModel</code></td>
<td><code>Schema</code></td>
</tr>
<tr>
<td>Stores</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Existing Technologies

<table>
<thead>
<tr>
<th>Objects</th>
<th>Core Data</th>
<th>CloudKit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NSManagedObject</td>
<td>CKRecord</td>
</tr>
<tr>
<td>Models</td>
<td>NSManagedObjectModel</td>
<td>Schema</td>
</tr>
<tr>
<td>Stores</td>
<td>NSPersistentStore</td>
<td></td>
</tr>
</tbody>
</table>
## Existing Technologies

<table>
<thead>
<tr>
<th></th>
<th>Core Data</th>
<th>CloudKit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Objects</strong></td>
<td>NSManagedObject</td>
<td>CKRecord</td>
</tr>
<tr>
<td><strong>Models</strong></td>
<td>NSManagedObjectModel</td>
<td>Schema</td>
</tr>
<tr>
<td><strong>Stores</strong></td>
<td>NSPersistentStore</td>
<td>CKRecordZone / CKDatabase</td>
</tr>
</tbody>
</table>
Welcome to Xcode

Version 11.0 beta (11i.335v)

Get started with a playground
Explore new ideas quickly and easily.

Create a new Xcode project
Create an app for iPhone, iPad, Mac, Apple Watch, or Apple TV.

Clone an existing project
Start working on something from a Git repository.

No Recent Projects

Open another project...
Welcome to Xcode

Version 11.0 beta (11l.335v)

Get started with a playground
Explore new ideas quickly and easily.

Create a new Xcode project
Create an app for iPhone, iPad, Mac, Apple Watch, or Apple TV.

Clone an existing project
Start working on something from a Git repository.

No Recent Projects

Open another project...
 WWDCDemo

PROJECT WWDCDemo

TARGETS WWDCDemo

WWDCDemoTests

WWDCDemoUITests

WWDCDemoTests.swift

WWDCDemoUITests.swift

Info.plist

Products

IDENTITY AND TYPE
Name WWDCDemo
Location Absolute
Full Path /Users/dm/Demo/WWDCDemo/WWDCDemo.xcodeproj

PROJECT DOCUMENT
Project Format Xcode 9.3-compatible
Organization Demo
Class Prefix

TEXT SETTINGS
Indent Using Spaces
Widths 4 4
Tab Wrap lines

SIGNING & CAPABILITIES

+ Capability All Debug Release

SIGNING

Automatically manage signing
Xcode will create and update profiles, app IDs, and certificates.

Team Apple Inc. - Cocoa

Bundle Identifier com.apple.wwdc:WWDCDemo

Provisioning Profile Xcode Managed Profile

Signing Certificate Apple Development: Nick Gillett (3SU9J9Q5764)

Add capabilities by clicking the "+" button above.
Signing & Capabilities

+ Capability  All  Debug  Release

- Signing

  - Automatically manage signing
  - Team: Apple Inc. – Cocoa
  - Bundle Identifier: com.apple.wwdc:WWDCDemo
  - Provisioning Profile: Xcode Managed Profile
  - Signing Certificate: Apple Development: Nick Gillett (3J5Q9B5764)

Add capabilities by clicking the "+" button above.
WWDCDemo

General

Signing & Capabilities

Resource Tags Info Build Settings Build Phases Build Rules

PROJECT WWDCDemo

TARGETS WWDCDemo

+ Capability All Debug Release

Signing

Automatically manage signing
Xcode will create and update profiles, app IDs, and certificates.

Team Apple Inc. – Cocoa

Bundle Identifier com.apple.wwdc.WWDCDemo

Provisioning Profile Xcode Managed Profile

Signing Certificate Apple Development: Nick Gillett (3U5Q9857E4)

iCloud

Services

- Key-value storage
- iCloud Documents
- CloudKit

Containers

- Use default containers
- Specify custom containers
  - iCloud.com.apple.wwdc.WWDCDemo iCloud
  - com.apple.coredata.cuds
  - com.apple.sqlite NOTES

- CloudKit Dashboard

Push Notifications

Add capabilities by clicking the “+” button above.
Detail view content goes here
Detail view content goes here
<table>
<thead>
<tr>
<th>Edit</th>
<th>Master</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2019-06-03 16:41:50 +0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2019-06-03 16:41:49 +0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2019-06-03 16:41:49 +0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2019-06-03 16:41:48 +0000</td>
<td></td>
</tr>
</tbody>
</table>

Detail view content goes here
<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019-06-03</td>
<td>16:42:34</td>
<td></td>
</tr>
<tr>
<td>2019-06-03</td>
<td>16:42:34</td>
<td></td>
</tr>
<tr>
<td>2019-06-03</td>
<td>16:42:33</td>
<td></td>
</tr>
<tr>
<td>2019-06-03</td>
<td>16:42:33</td>
<td></td>
</tr>
<tr>
<td>2019-06-03</td>
<td>16:41:50</td>
<td></td>
</tr>
<tr>
<td>2019-06-03</td>
<td>16:41:49</td>
<td></td>
</tr>
<tr>
<td>2019-06-03</td>
<td>16:41:49</td>
<td></td>
</tr>
<tr>
<td>2019-06-03</td>
<td>16:41:48</td>
<td></td>
</tr>
</tbody>
</table>

Detail view content goes here
<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019-06-03</td>
<td>16:42:34</td>
<td>+0000</td>
</tr>
<tr>
<td>2019-06-03</td>
<td>16:42:34</td>
<td>+0000</td>
</tr>
<tr>
<td>2019-06-03</td>
<td>16:42:33</td>
<td>+0000</td>
</tr>
<tr>
<td>2019-06-03</td>
<td>16:42:33</td>
<td>+0000</td>
</tr>
<tr>
<td>2019-06-03</td>
<td>16:41:50</td>
<td>+0000</td>
</tr>
<tr>
<td>2019-06-03</td>
<td>16:41:49</td>
<td>+0000</td>
</tr>
<tr>
<td>2019-06-03</td>
<td>16:41:49</td>
<td>+0000</td>
</tr>
<tr>
<td>2019-06-03</td>
<td>16:41:49</td>
<td>+0000</td>
</tr>
<tr>
<td>2019-06-03</td>
<td>16:41:48</td>
<td>+0000</td>
</tr>
</tbody>
</table>
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
    // Override point for customization after application launch.
    let splitViewController = self.window!.rootViewController as! UISplitViewController
    let navigationController = 
        splitViewController.viewControllers[splitViewController.viewControllers.count-1] as! UINavigationController
    navigationController.topViewController!.navigationBar.leftBarButtonItem = 
        splitViewController.displayModeButtonItem
    splitViewController.delegate = self

    let masterNavigationController = splitViewController.viewControllers[0] as! UINavigationController
    let controller = masterNavigationController.topViewController as! MasterViewController
    controller.managedObjectContext = self.persistentContainer.viewContext
    self.persistentContainer.viewContext.automaticallyMergesChangesFromParent = true
    return true
}

func applicationWillResignActive(_ application: UIApplication) {
    // Sent when the application is about to move from active to inactive state. This can occur for certain types of temporary interruptions (such as an incoming phone call or SMS message) or when the user quits the application and it begins the transition to the background state.
    // Use this method to pause ongoing tasks, disable timers, and invalidate graphics rendering callbacks. Games should use this method to pause the game.
}

func applicationDidEnterBackground(_ application: UIApplication) {
    // Use this method to release shared resources, save user data, invalidate timers, and store enough application state information to restore your application to its current state before it was suspended.
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
    // Override point for customization after application launch.
    let splitViewController = self.window!.rootViewController as! UISplitViewController
    let navigationController = 
        splitViewController.viewControllers[splitViewController.viewControllers.count-1] as! UINavigationController
    navigationController.topViewController!.navigationBar.items[0].buttonItems = 
        splitViewController.displayModeButtonItem
    splitViewController.delegate = self

    let masterNavigationController = splitViewController.viewControllers[0] as! UINavigationController
    let controller = masterNavigationController.topViewController as! MasterViewController
    controller.managedObjectContext = self.persistentContainer.viewContext
    self.persistentContainer.viewContext.automaticallyMergesChangesFromParent = true
    return true
}

func applicationWillResignActive(_ application: UIApplication) {
    // Sent when the application is about to move from active to inactive state. This can occur for certain types of temporary interruptions (such as an incoming phone call or SMS message) or when the user quits the application and it begins the transition to the background state.
    // Use this method to pause ongoing tasks, disable timers, and invalidate graphics rendering callbacks. Games should use this method to pause the game.
}

func applicationDidEnterBackground(_ application: UIApplication) {
    // Use this method to release shared resources, save user data, invalidate timers, and store enough application state information to restore your application to its current state before the next launch.
}
return false

// MARK: - Core Data stack

lazy var persistentContainer: NSPersistentContainer = {
    /*
    The persistent container for the application. This implementation
    creates and returns a container, having loaded the store for the
    application to it. This property is optional since there are legitimate
    error conditions that could cause the creation of the store to fail.
    */
    let container = NSPersistentContainer(name: "WWDCDemo")
    container.loadPersistentStores(completionHandler: { (storeDescription, error) in
        if let error = error as NSError? {
            // Replace this implementation with code to handle the error appropriately.
            // fatalError() causes the application to generate a crash log and terminate. You
            // should not use this function in a shipping application, although it may be
            // useful during development.

            /*
            Typical reasons for an error here include:
            * The parent directory does not exist, cannot be created, or disallows writing.
            * The persistent store is not accessible, due to permissions or data protection
            * when the device is locked.
            * The device is out of space.
            * The store could not be migrated to the current model version.
            Check the error message to determine what the actual problem was.
            */
            fatalError("Unresolved error \(error), \(error.userInfo)")
        }
    })
}
}
return false

// MARK: - Core Data stack

lazy var persistentContainer: NSPersistentContainer = {

    /*
    The persistent container for the application. This implementation
    creates and returns a container, having loaded the store for the
    application to it. This property is optional since there are legitimate
    error conditions that could cause the creation of the store to fail.
    */

    let container = NSPersistentContainer(name: "WWCDemo"
    container.loadPersistentStores(completionHandler: { (storeDescription, error) in
        if let error = error as NSError? {
            // Replace this implementation with code to handle the error appropriately.
            // fatalError() causes the application to generate a crash log and terminate. You
            // should not use this function in a shipping application, although it may be
            // useful during development.

            /*
            Typical reasons for an error here include:
            * The parent directory does not exist, cannot be created, or disallows writing.
            * The persistent store is not accessible, due to permissions or data protection
            when the device is locked.
            * The device is out of space.
            * The store could not be migrated to the current model version.
            Check the error message to determine what the actual problem was.
            */
            fatalError("Unresolved error \(error), \error.userInfo")
        }
    })
}
}
What Is NSPersistentCloudKitContainer?
What Is NSPersistentCloudKitContainer?

Encapsulation of common patterns
What Is NSPersistentCloudKitContainer?

Encapsulation of common patterns

Save thousands of lines of code
What Is NSPersistentCloudKitContainer?

Encapsulation of common patterns
Save thousands of lines of code
Foundation we can build on
What Is NSPersistentCloudKitContainer?

Encapsulation of common patterns

Save thousands of lines of code

Foundation we can build on

Help us help you! Submit feedback!
NSPersistentCloudKitContainer
NSPersistentCloudKitContainer

A local replica of all CloudKit data
NSPersistentCloudKitContainer

A local replica of all CloudKit data

Robust scheduling and error recovery
NSPersistentCloudKitContainer

A local replica of all CloudKit data

Robust scheduling and error recovery

Transformation of NSManagedObject to CKRecord
Local Replica
Local Replica
Local Replica
Local Replica
Local Replica
Local Replica
Local Replica

milliseconds

seconds – minutes
Local Replica

GB/s

KB/s – MB/s
Local Replica
Scheduling
Scheduling
Scheduling
Scheduling
Scheduling
Scheduling
Scheduling
Scheduling
Scheduling
Scheduling
Scheduling
Scheduling
Scheduling
Scheduling
NSManagedObject to CKRecord
NSManagedObject to CKRecord
NSManagedObject to CKRecord
NSManagedObject to CKRecord
NSManagedObject to CKRecord
NSManagedObject to CKRecord
NSManagedObject to CKRecord
NSManagedObject to CKRecord
NSManagedObject to CKRecord
NSManagedObject to CKRecord
NSManagedObject to CKRecord
NSManagedObject to CKRecord
NSManagedObject to CKRecord
NSManagedObject to CKRecord
NSManagedObject to CKRecord
NSManagedObject to CKRecord
NSPersistentCloudKitContainer
NSPersistentCloudKitContainer

Complete local replica
NSPersistentCloudKitContainer

Complete local replica

Automatic scheduling
NSPersistentCloudKitContainer

Complete local replica

Automatic scheduling

Serialization from NSManagedObject to CKRecord
Life After Adopting NSPersistentCloudKitContainer
Life After Adopting NSPersistentCloudKitContainer

Build great applications with Core Data
Life After Adopting NSPersistentCloudKitContainer

Build great applications with Core Data

Add on to our foundation
Learning About NSPersistentCloudKitContainer
Learning About NSPersistentCloudKitContainer

Mirroring a Core Data Store with CloudKit
Setting Up Core Data with CloudKit
Creating a Core Data Model for CloudKit
Syncing a Core Data Store with CloudKit
Reading CloudKit Records for Core Data
Build Great Apps with Core Data
Build Great Apps with Core Data

Responsive user interfaces with NSFetchResultsController
Build Great Apps with Core Data

Responsive user interfaces with NSFetchResultsController

Stable views with query generations
Build Great Apps with Core Data

Responsive user interfaces with NSFetchResultsController
Stable views with query generations
Change processing with history tracking
Build Great Apps with Core Data

Responsive user interfaces with NSFetchResultsController

Stable views with query generations

Change processing with history tracking
<table>
<thead>
<tr>
<th>Title</th>
<th>Content</th>
<th>Tags</th>
<th>Attachments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Another great demo!</td>
<td><img src="demo.png" alt="image" /></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hi WWDC!</td>
<td><img src="demo.png" alt="image" /></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Posts</td>
<td>Detail</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TITLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CONTENT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TAGS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ATTACHMENTS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Life After Adopting NSPersistentCloudKitContainer

Build great applications with Core Data

Add on to our foundation
Extending NSPersistentCloudKitContainer
Extending NSPersistentCloudKitContainer

Working with multiple stores
Extending NSPersistentCloudKitContainer

Working with multiple stores

Working with the CloudKit Schema
Extending NSPersistentCloudKitContainer

- Working with multiple stores
- Working with the CloudKit Schema
- Data Modeling for collaboration
Multiple Store Use Cases
Multiple Store Use Cases

Data Segregation
Multiple Store Use Cases

Data Segregation

Enforcement of different types of constraints
Multiple Store Use Cases

Data Segregation

Enforcement of different types of constraints

Throttling/Coalescing
NSPersistentCloudKitContainer — Multiple Stores
NSPersistentCloudKitContainer — Multiple Stores
NSPersistentCloudKitContainer — Multiple Stores
NSPersistentCloudKitContainer — Multiple Stores
NSPersistentCloudKitKitContainer — Multiple Stores
NSPersistentCloudKitContainer — Multiple Stores
NSPersistentCloudKitContainer — Multiple Stores
NSPersistentCloudKitContainer — Multiple Stores
NSPersistentCloudKitContainer — Multiple Stores
NSPersistentCloudKitContainer — Multiple Stores
NSPersistentCloudKitContainer — Multiple Stores
NSPersistentCloudKitContainer — Multiple Stores
NSPersistentCloudKitContainer — Multiple Stores
NSPersistentCloudKitContainer — Multiple Stores
NSPersistentCloudKitContainer — Multiple Stores
### Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
</tr>
</thead>
</table>

### Relationships

<table>
<thead>
<tr>
<th>Relationship</th>
<th>Destination</th>
<th>Inverse</th>
</tr>
</thead>
</table>

### Fetched Properties

<table>
<thead>
<tr>
<th>Fetched Property</th>
<th>Predicate</th>
</tr>
</thead>
</table>
let container = NSPersistentCloudKitContainer(name: "CloudKitContainer")

let local = NSPersistentStoreDescription(url: URL(fileURLWithPath: "/files/local.sqlite"))
local.configuration = "Local"

let cloud = NSPersistentStoreDescription(url: URL(fileURLWithPath: "/files/cloud.sqlite"))
cloud.configuration = "Cloud"
cloud.cloudKitContainerOptions = NSPersistentCloudKitContainerOptions(containerIdentifier: "iCloud.com.wwdc.demo")

container.persistentStoreDescriptions = [ local, cloud ]
let container = NSPersistentCloudKitContainer(name: "CloudKitContainer")

let local = NSPersistentStoreDescription(url: URL(fileURLWithPath: "/files/local.sqlite"))
local.configuration = "Local"

let cloud = NSPersistentStoreDescription(url: URL(fileURLWithPath: "/files/cloud.sqlite"))
cloud.configuration = "Cloud"
cloud.cloudKitContainerOptions = NSPersistentCloudKitContainerOptions(containerIdentifier: "iCloud.com.wwdc.demo")

container.persistentStoreDescriptions = [ local, cloud ]
let container = NSPersistentCloudKitContainer(name: "CloudKitContainer")

let local = NSPersistentStoreDescription(url: URL(fileURLWithPath: "/files/local.sqlite"))
local.configuration = "Local"

let cloud = NSPersistentStoreDescription(url: URL(fileURLWithPath: "/files/cloud.sqlite"))
cloud.configuration = "Cloud"
cloud.cloudKitContainerOptions = NSPersistentCloudKitContainerOptions(containerIdentifier: "iCloud.com.wwdc.demo")

container.persistentStoreDescriptions = [ local, cloud ]
let container = NSPersistentCloudKitContainer(name: "CloudKitContainer")

let local = NSPersistentStoreDescription(url: URL(fileURLWithPath: "/files/local.sqlite"))
local.configuration = "Local"

let cloud = NSPersistentStoreDescription(url: URL(fileURLWithPath: "/files/cloud.sqlite"))
cloud.configuration = "Cloud"
cloud.cloudKitContainerOptions = NSPersistentCloudKitContainerOptions(containerIdentifier: "iCloud.com.wwdc.demo")

container.persistentStoreDescriptions = [ local, cloud ]
let container = NSPersistentCloudKitContainer(name: "CloudKitContainer")

let local = NSPersistentStoreDescription(url: URL(fileURLWithPath: "/files/local.sqlite"))
local.configuration = "Local"

let cloud = NSPersistentStoreDescription(url: URL(fileURLWithPath: "/files/cloud.sqlite"))
cloud.configuration = "Cloud"
cloud.cloudKitContainerOptions = NSPersistentCloudKitContainerOptions(containerIdentifier: "iCloud.com.wwdc.demo")

container.persistentStoreDescriptions = [ local, cloud ]
NSPersistentCloudKitContainer — Multiple Stores
NSPersistentCloudKitContainer — Multiple Stores
NSPersistentCloudKitContainer — Multiple Stores
NSPersistentCloudKitContainer — Multiple Stores
NSPersistentCloudKitContainer — Multiple Stores
let container = NSPersistentCloudKitContainer(name: "CloudKitContainer")

let local = NSPersistentStoreDescription(url: URL(fileURLWithPath: "/files/local.sqlite"))
local.configuration = "Local"

let cloud = NSPersistentStoreDescription(url: URL(fileURLWithPath: "/files/cloud.sqlite"))
cloud.configuration = "Cloud"
cloud.cloudKitContainerOptions = NSPersistentCloudKitContainerOptions(containerIdentifier: "iCloud.com.wwdc.demo")

let shared = NSPersistentStoreDescription(url: URL(fileURLWithPath: "/files/shared.sqlite"))
shared.configuration = "Shared"
shared.cloudKitContainerOptions = NSPersistentCloudKitContainerOptions(containerIdentifier: "iCloud.com.wwdc.shared")

container.persistentStoreDescriptions = [ local, cloud, shared ]
let container = NSPersistentCloudKitContainer(name: "CloudKitContainer")

let local = NSPersistentStoreDescription(url: URL(fileURLWithPath: "/files/local.sqlite"))
local.configuration = "Local"

let cloud = NSPersistentStoreDescription(url: URL(fileURLWithPath: "/files/cloud.sqlite"))
cloud.configuration = "Cloud"
cloud.cloudKitContainerOptions = NSPersistentCloudKitContainerOptions(containerIdentifier: "iCloud.com.wwdc.demo")

let shared = NSPersistentStoreDescription(url: URL(fileURLWithPath: "/files/shared.sqlite"))
shared.configuration = "Shared"
shared.cloudKitContainerOptions = NSPersistentCloudKitContainerOptions(containerIdentifier: "iCloud.com.wwdc.shared")

container.persistentStoreDescriptions = [ local, cloud, shared ]
let container = NSPersistentCloudKitContainer(name: "CloudKitContainer")

let local = NSPersistentStoreDescription(url: URL(fileURLWithPath: "/files/local.sqlite"))
local.configuration = "Local"

let cloud = NSPersistentStoreDescription(url: URL(fileURLWithPath: "/files/cloud.sqlite"))
cloud.configuration = "Cloud"
cloud.cloudKitContainerOptions = NSPersistentCloudKitContainerOptions(containerIdentifier: "iCloud.com.wwdc.demo")

let shared = NSPersistentStoreDescription(url: URL(fileURLWithPath: "/files/shared.sqlite"))
shared.configuration = "Shared"
shared.cloudKitContainerOptions = NSPersistentCloudKitContainerOptions(containerIdentifier: "iCloud.com.wwdc.shared")

container.persistentStoreDescriptions = [ local, cloud, shared ]
NSPersistentCloudKitContainer's Schema
NSPersistentCloudKitContainer's Schema

Record types and entity names
NSPersistentCloudKitContainer's Schema

Record types and entity names

Asset externalization
NSPersistentCloudKitContainer's Schema

Record types and entity names

Asset externalization

Relationships
Sample Application Data Model

- **Post**
  - Attributes: content, title
  - Relationships: attachments, tags

- **Attachment**
  - Attributes: thumbnail, uuid
  - Relationships: imageData, post

- **ImageData**
  - Attributes: data
  - Relationships: attachment

- **Tag**
  - Attributes: color, name, uuid
  - Relationships: posts

- **Attachment**
  - Attributes: thumbnail, uuid
  - Relationships: imageData, post

- **ImageData**
  - Attributes: data
  - Relationships: attachment

- **Tag**
  - Attributes: color, name, uuid
  - Relationships: posts
@objc(Post)
public class Post: NSManagedObject {
    @NSManaged public var title: String?
    @NSManaged public var content: String?
    @NSManaged public var attachments: NSSet?
    @NSManaged public var tags: NSSet?
}
@objc(Post)

public class Post: NSManagedObject {

    @NSManaged public var title: String?
    @NSManaged public var content: String?
    @NSManaged public var attachments: NSSet?
    @NSManaged public var tags: NSSet?

}
@objc(Post)
public class Post: NSManagedObject {
    @NSManaged public var title: String?
    @NSManaged public var content: String?
    @NSManaged public var attachments: NSSet?
    @NSManaged public var tags: NSSet?
}

<CKRecord: 0x7fd0b2539220; recordID=Post_UUID, values={
    "CD_content" = "An example core data string";
    "CD_content_ckAsset" = "<CKAsset: 0x7fd0b2515c20;...>";
    "CD_title" = "An example core data string";
    "CD_title_ckAsset" = "<CKAsset: 0x7fd0b240bd40;...>";
    "CD_entityName" = Post;
}, recordType=CD_Post>
@objc(Post)
public class Post: NSManagedObject {
    @NSManaged public var title: String?
    @NSManaged public var content: String?
    @NSManaged public var attachments: NSSet?
    @NSManaged public var tags: NSSet?
}

<CKRecord: 0x7fd0b2539220; recordID=Post_UUID, values={
    "CD_content" = "An example core data string";
    "CD_content_ckAsset" = "<CKAsset: 0x7fd0b2515c20;...>";
    "CD_title" = "An example core data string";
    "CD_title_ckAsset" = "<CKAsset: 0x7fd0b40bd40;...>";
    "CD_entityName" = Post;
}, recordType=CD_Post>
@objc(Post)
public class Post: NSManagedObject {
    @NSManaged public var title: String?
    @NSManaged public var content: String?
    @NSManaged public var attachments: NSSet?
    @NSManaged public var tags: NSSet?
}

<CKRecord: 0x7fd0b2539220; recordID=Post_UUID, values={
    "CD_content" = "An example core data string";
    "CD_content_ckAsset" = "<CKAsset: 0x7fd0b2515c20;...>";
    "CD_title" = "An example core data string";
    "CD_title_ckAsset" = "<CKAsset: 0x7fd0b240bd40;...>";
    "CD_entityName" = Post;
}, recordType=CD_Post>
@objc(Post)

public class Post: NSManagedObject {
    @NSManaged public var title: String?
    @NSManaged public var content: String?
    @NSManaged public var attachments: NSSet?
    @NSManaged public var tags: NSSet?
}

<CKRecord: 0x7fd0b2539220; recordID=Post_UUID, values={
    "CD_content" = "An example core data string";
    "CD_content_ckAsset" = "<CKAsset: 0x7fd0b2515c20;...>";
    "CD_title" = "An example core data string";
    "CD_title_ckAsset" = "<CKAsset: 0x7fd0b240bd40;...>";
    "CD_entityName" = ImagePost;
}, recordType=CD_Post>
@objc(Post)
public class Post: NSManagedObject {
    @NSManaged public var title: String?
    @NSManaged public var content: String?
    @NSManaged public var attachments: NSSet?
    @NSManaged public var tags: NSSet?
}

<CKRecord: 0x7fd0b2539220; recordID=Post_UUID, values={
    "CD_content" = "An example core data string";
    "CD_content_ckAsset" = "<CKAsset: 0x7fd0b2515c20;...>";
    "CD_title" = "An example core data string";
    "CD_title_ckAsset" = "<CKAsset: 0x7fd0b40bd40;...>";
    "CD_entityName" = ImagePost;
}, recordType=CD_Post>
@objc(Post)

public class Post: NSManagedObject {
    @NSManaged public var title: String?
    @NSManaged public var content: String?
    @NSManaged public var attachments: NSSet?
    @NSManaged public var tags: NSSet?
}

<CKRecord: 0x7fd0b2539220; recordID=Post_UUID, values={
    "CD_content" = "An example core data string";
    "CD_content_ckAsset" = "<CKAsset: 0x7fd0b2515c20;...>";
    "CD_title" = "An example core data string";
    "CD_title_ckAsset" = "<CKAsset: 0x7fd0b40bd40;...>";
    "CD_entityName" = VideoPost;
}, recordType=CD_Post>
@objc(Post)
public class Post: NSManagedObject {
    @NSManaged public var title: String?
    @NSManaged public var content: String?
    @NSManaged public var attachments: NSSet?
    @NSManaged public var tags: NSSet?
}

<CKRecord: 0x7fd0b2539220; recordID=Post_UUID, values={
    "CD_content" = "An example core data string";
    "CD_content_ckAsset" = "<CKAsset: 0x7fd0b2515c20;...>";
    "CD_title" = "An example core data string";
    "CD_title_ckAsset" = "<CKAsset: 0x7fd0b40bd40;...>";
    "CD_entityName" = VideoPost;
}, recordType=CD_Post>
public class Post: NSManagedObject {
    @NSManaged public var title: String?
    @NSManaged public var content: String?
    @NSManaged public var attachments: NSSet?
    @NSManaged public var tags: NSSet?
}

<CKRecord: 0x7fd0b2539220; recordID=Post_UUID, values={
    "CD_content" = "An example core data string";
    "CD_content_ckAsset" = "<CKAsset: 0x7fd0b2515c20;...>";
    "CD_title" = "An example core data string";
    "CD_title_ckAsset" = "<CKAsset: 0x7fd0b240bd40;...>";
    "CD_entityName" = Post;
}, recordType=CD_Post>
@objc(Post)
public class Post: NSManagedObject {
    @NSManaged public var title: String?
    @NSManaged public var content: String?
    @NSManaged public var attachments: NSSet?
    @NSManaged public var tags: NSSet?
}

<CKRecord: 0x7fd0b2539220; recordID=Post_UUID, values={["CD_content" = "An example core data string";
    "CD_content_ckAsset" = "<CKAsset: 0x7fd0b2515c20;...>";
    "CD_title" = "An example core data string";
    "CD_title_ckAsset" = "<CKAsset: 0x7fd0b40bd40;...>";
    "CD_entityName" = Post;]
}, recordType=CD_Post>
@objc(Post)
public class Post: NSManagedObject {
    @NSManaged public var title: String?
    @NSManaged public var content: String?
    @NSManaged public var attachments: NSSet?
    @NSManaged public var tags: NSSet?
}

<CKRecord: 0x7fd0b2539220; recordID=Post_UUID, values={
    "CD_content" = "An example core data string";
    "CD_content_ckAsset" = "<CKAsset: 0x7fd0b2515c20;...>";
    "CD_title" = "An example core data string";
    "CD_title_ckAsset" = "<CKAsset: 0x7fd0b240bd40;...>";
    "CD_entityName" = Post;
}, recordType=CD_Post>
@objc(Post)
public class Post: NSManagedObject {
    @NSManaged public var title: String?
    @NSManaged public var content: String?
    @NSManaged public var attachments: NSSet?
    @NSManaged public var tags: NSSet?
}

<CKRecord: 0x7fd0b2539220; recordID=Post_UUID, values={
    "CD_content" = "Amazing... but brief!";
    "CD_title" = "An amazing post!";
    "CD_entityName" = Post;
}, recordType=CD_Post>
@objc(Post)
public class Post: NSManagedObject {
    @NSManaged public var title: String?
    @NSManaged public var content: String?
    @NSManaged public var attachments: NSSet?
    @NSManaged public var tags: NSSet?
}

<CKRecord: 0x7fd0b2539220; recordID=Post_UUID, values=
    "CD_content_ckAsset" = "<CKAsset: 0x7fd0b2515c20;...>";
    "CD_title" = "An amazing post!";
    "CD_entityName" = Post;
}, recordType=CD_Post>
@objc(Post)
public class Post: NSManagedObject {
    @NSManaged public var title: String?
    @NSManaged public var content: String?
    @NSManaged public var attachments: NSSet?
    @NSManaged public var tags: NSSet?
}

<CKRecord: 0x7fd0b2539220; recordID=Post_UUID, values={
    "CD_content_ckAsset" = "<CKAsset: 0x7fd0b2515c20;...>";
    "CD_title" = "An amazing post!";
    "CD_entityName" = Post;
}, recordType=CD_Post>
@objc(Post)

public class Post: NSManagedObject {
    @NSManaged public var title: String?
    @NSManaged public var content: String?
    @NSManaged public var attachments: NSSet?
    @NSManaged public var tags: NSSet?
}

@objc(Post)
public class Post: NSManagedObject {
  @NSManaged public var title: String?
  @NSManaged public var content: String?
  @NSManaged public var attachments: NSSet?
  @NSManaged public var tags: NSSet?
}

Sample Application Data Model

- **Post**
  - **Attributes**
    - content
    - title
  - **Relationships**
    - attachments
    - tags

- **Attachment**
  - **Attributes**
    - thumbnail
    - uuid
  - **Relationships**
    - imageData
    - post

- **ImageData**
  - **Attributes**
    - data
  - **Relationships**
    - attachment

- **Tag**
  - **Attributes**
    - color
    - name
    - uuid
  - **Relationships**
    - posts
@objc(Post)
public class Post: NSManagedObject {
    @NSManaged public var attachments: NSSet?
}

@objc(Attachment)
public class Attachment: NSManagedObject {
    @NSManaged public var post: Post?
}
@objc(Post)
public class Post: NSManagedObject {
    @NSManaged public var attachments: NSSet?
}

@objc(Attachment)
public class Attachment: NSManagedObject {
    @NSManaged public var post: Post?
}
@objc(Post)
public class Post: NSManagedObject {
    @NSManaged public var attachments: NSSet?
}

@objc(Attachment)
public class Attachment: NSManagedObject {
    @NSManaged public var post: Post?
}
@objc(Post)
public class Post: NSManagedObject {
    @NSManaged public var attachments: NSSet?
}

@objc(Attachment)
public class Attachment: NSManagedObject {
    @NSManaged public var post: Post?
}

<CKRecord: 0x7fd0b4011bf0; recordID=Attachment_UUID, values={
    "CD_entityName" = Attachment;
    "CD_imageData" = "ImageData_UUID";
    "CD_post" = "Post_UUID";
    "CD_uuid" = "B7841A69-868D-49A9-851D-3BB7030DA73A";
}, recordType=CD_Attachment>
@objc(Post)
public class Post: NSManagedObject {
    @NSManaged public var attachments: NSSet?
}

@objc(Attachment)
public class Attachment: NSManagedObject {
    @NSManaged public var post: Post?
}

<CKRecord: 0x7fd0b4011bf0; recordID=Attachment_UUID, values={
    "CD_entityName" = Attachment;
    "CD_imageData" = "ImageData_UUID";
    "CD_post" = "Post_UUID";
    "CD_uuid" = "B7841A69-868D-49A9-851D-3BB7030DA73A";
}, recordType=CD_Attachment>
@objc(Post)
public class Post: NSManagedObject {
    @NSManaged public var attachments: NSSet?
}

@objc(Attachment)
public class Attachment: NSManagedObject {
    @NSManaged public var post: Post?
}

<CKRecord: 0x7fd0b4011bf0; recordID=Attachment_UUID, values={
    "CD_entityName" = Attachment;
    "CD_imageData" = "ImageData_UUID";
    "CD_post" = "Post_UUID";
    "CD_uuid" = "B7841A69-868D-49A9-851D-3BB7030DA73A";
}, recordType=CD_Attachment>
Sample Application Data Model

**Post**
- **Attributes**
  - content
  - title
- **Relationships**
  - attachments
  - tags

**Attachment**
- **Attributes**
  - thumbnail
  - uuid
- **Relationships**
  - imageData
  - post

**ImageData**
- **Attributes**
  - data
- **Relationships**
  - attachment

**Tag**
- **Attributes**
  - color
  - name
  - uuid
- **Relationships**
  - posts
@objc(Post)
public class Post: NSManagedObject {
    @NSManaged public var tags: NSSet?
}

@objc(Tag)
public class Tag: NSManagedObject {
    @NSManaged public var posts: NSSet?
}
@objc(Post)
public class Post: NSManagedObject {
    @NSManaged public var tags: NSSet?
}

@objc(Tag)
public class Tag: NSManagedObject {
    @NSManaged public var posts: NSSet?
}
<CKRecord: 0x7fd0b2539220; recordID=Post_UUID, values={
    "CD_content" = "An example core data string";
    "CD_content_ckAsset" = "<CKAsset: 0x7fd0b2515c20;...>";
    "CD_title" = "An example core data string";
    "CD_title_ckAsset" = "<CKAsset: 0x7fd0b240bd40;...>";
    "CD_entityName" = Post;
}, recordType=CD_Post>

<CKRecord: 0x7fd0b2419da0; recordID=Tag_UUID, values={
    "CD_color" = {length = 17, bytes = 0x536f6d65206578616d706c652064617461};
    "CD_color_ckAsset" = "<CKAsset: 0x7fd0b2532de0;...>";
    "CD_name" = "An example core data string";
    "CD_name_ckAsset" = "<CKAsset: 0x7fd0b24215d0;...>";
    "CD_uuid" = "DEB9D10A-55C1-42EC-BA15-77A41028AF9B";
    "CD_entityName" = Tag;
}, recordType=CD_Tag>
<CKRecord: 0x7fd0b24202f0; recordID=CDMR_UUID, values={
    "CD_entityNames" = "Post:Tag";
    "CD_recordNames" = "Post_UUID:Tag_UUID";
    "CD_relationships" = "tags:posts";
}, recordType=CDMR>
<CKRecord: 0x7fd0b24202f0; recordID=CDMR_UUID, values={
    "CD_entityNames" = "Post:Tag";
    "CD_recordNames" = "Post_UUID:Tag_UUID";
    "CD_relationships" = "tags:posts";
}, recordType=CDMR>
<CKRecord: 0x7fd0b24202f0; recordID=CDMR_UUID, values={
    "CD_entityNames" = "Post:Tag";
    "CD_recordNames" = "Post_UUID:Tag_UUID";
    "CD_relationships" = "tags:posts";
}, recordType=CDMR>
<CKRecord: 0x7fd0b24202f0; recordID=CDMR_UUID, values={
    "CD_entityNames" = "Post:Tag";
    "CD_recordNames" = "Post_UUID:Tag_UUID";
    "CD_relationships" = "tags:posts";
}, recordType=CDMR>
<CKRecord: 0x7fd0b24202f0; recordID=CDMR_UUID, values={
    "CD_entityNames" = "Post:Tag";
    "CD_recordNames" = "Post_UUID:Tag_UUID";
    "CD_relationships" = "tags:posts";
}, recordType=CDMR>
Data Modeling for Collaboration
Data Modeling for Collaboration

Collaboration is not conflict resolution
Data Modeling for Collaboration

Collaboration is not conflict resolution

NSPersistentCloudKitContainer resolves conflicts automatically
Data Modeling for Collaboration

Collaboration is not conflict resolution

NSPersistentCloudKitContainer resolves conflicts automatically

Get better merge behavior with relationships
Another great demo!

Hi WWDC!
Another great demo!

Hi WWDC!

Let's talk about collaboration
Another great demo!

Hi WWDC!

Let’s talk about collaboration
Another great demo!

Hi WWDC!

Let's talk about collaboration
Another great demo!

Hi WWDC!

Let's talk about collaboration
Let's talk about collaboration

Collaboration is great!

Everyone should do it!
<table>
<thead>
<tr>
<th>TITLE</th>
<th>CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Let's talk about collaboration</td>
<td>Collaboration is great!</td>
</tr>
<tr>
<td></td>
<td>Everyone should do it!</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Let’s talk about collaboration

Collaboration is great!
Let's talk about collaboration

Everyone should do it!
<table>
<thead>
<tr>
<th>TITLE</th>
<th>Let's talk about collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTENT</td>
<td>Collaboration is great! Everyone should do it!</td>
</tr>
<tr>
<td>TAGS</td>
<td></td>
</tr>
<tr>
<td>ATTACHMENTS</td>
<td></td>
</tr>
</tbody>
</table>
Let's talk about collaboration

Everyone Collaboration is should great! do it!
Data Modeling for Collaboration
Data Modeling for Collaboration

Post

✓ Attributes
  content
title

✓ Relationships
Data Modeling for Collaboration

Avoid collisions on "flat" values

- Post
  - Attributes
    - content
    - title
  - Relationships
Data Modeling for Collaboration

Avoid collisions on "flat" values
Instead model values as contributions

Post
- Attributes
  - title
- Relationships
  - PostContent
    - Attributes
      - contribution
    - Relationships
      - post
Data Modeling for Collaboration

Avoid collisions on "flat" values
Instead model values as contributions
Leverage relationships for eventual consistency
Avoid collisions on "flat" values

Instead model values as contributions

Leverage relationships for eventual consistency

Order contributions
Data Modeling for Collaboration

Avoid collisions on "flat" values

Instead model values as contributions

Leverage relationships for eventual consistency

Order contributions

Iterate as necessary
Summary

Easy sync with NSPersistentCloudKitContainer
Summary

Easy sync with NSPersistentCloudKitContainer

New sample code and documentation!
Summary

Easy sync with NSPersistentCloudKitContainer

New sample code and documentation!

Build on NSPersistentCloudKitContainer
More Information

developer.apple.com/wwdc19/202

<table>
<thead>
<tr>
<th>Lab</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core Data and CloudKit Lab</td>
<td>Tuesday, 3:00</td>
</tr>
<tr>
<td>CloudKit Lab</td>
<td>Wednesday, 9:00</td>
</tr>
<tr>
<td>Core Data Lab</td>
<td>Wednesday, 2:00</td>
</tr>
</tbody>
</table>