

# What's New in Core Data

Session 207

**Nick Gillett**

Core Data Software Engineer

These are confidential sessions—please refrain from streaming, blogging, or taking pictures

# Road Map

# Road Map

- iCloud

# Road Map

- iCloud
  - Fallback Store

# Road Map

- iCloud
  - Fallback Store
  - Asynchronous Setup

# Road Map

- iCloud
  - Fallback Store
  - Asynchronous Setup
  - Account Changes

# Road Map

- iCloud
  - Fallback Store
  - Asynchronous Setup
  - Account Changes
  - New API

# Road Map

- iCloud
  - Fallback Store
  - Asynchronous Setup
  - Account Changes
  - New API
  - Living on iCloud



# Road Map

- iCloud
  - Fallback Store
  - Asynchronous Setup
  - Account Changes
  - New API
  - Living on iCloud
- Demos

# Road Map

- iCloud
  - Fallback Store
  - Asynchronous Setup
  - Account Changes
  - New API
  - Living on iCloud
- Demos
- SQLite Store Enhancements

iCloud

# iCloud

- Speed

# iCloud

- Speed
- Simplicity

# iCloud

- Speed
- Simplicity
- Consistency

# Fallback Store

# Fallback Store





# Fallback Store



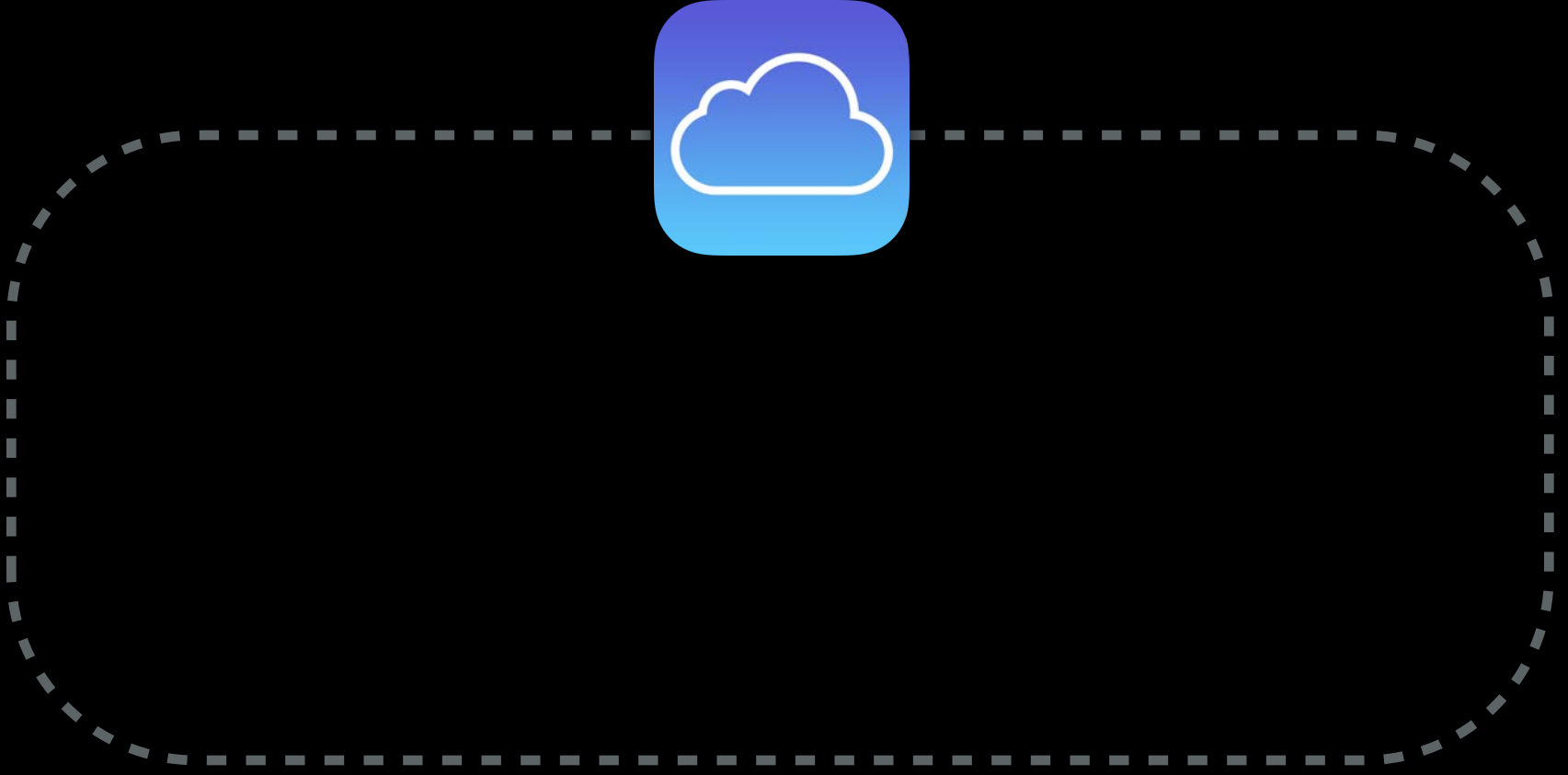
# Fallback Store



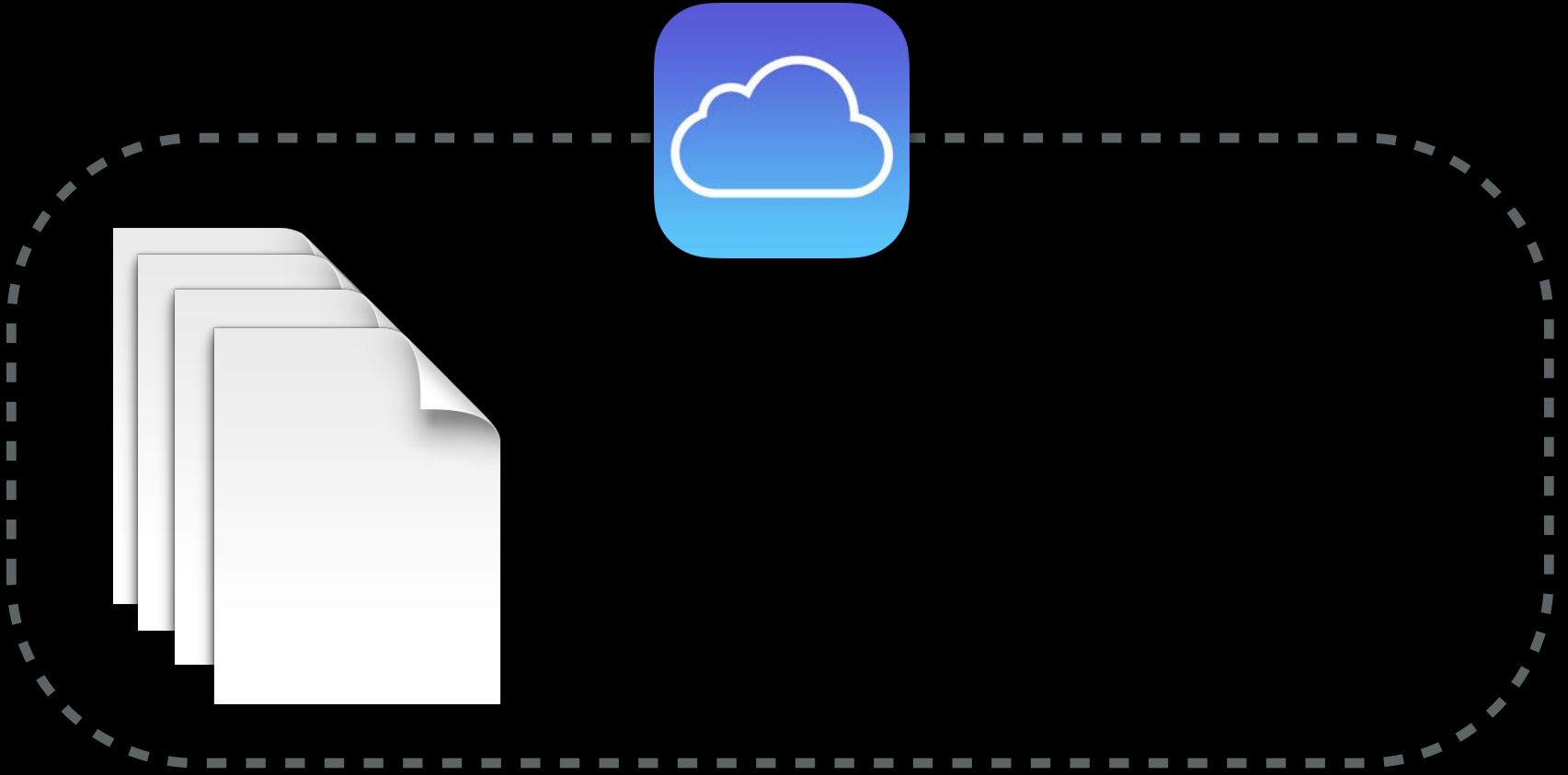
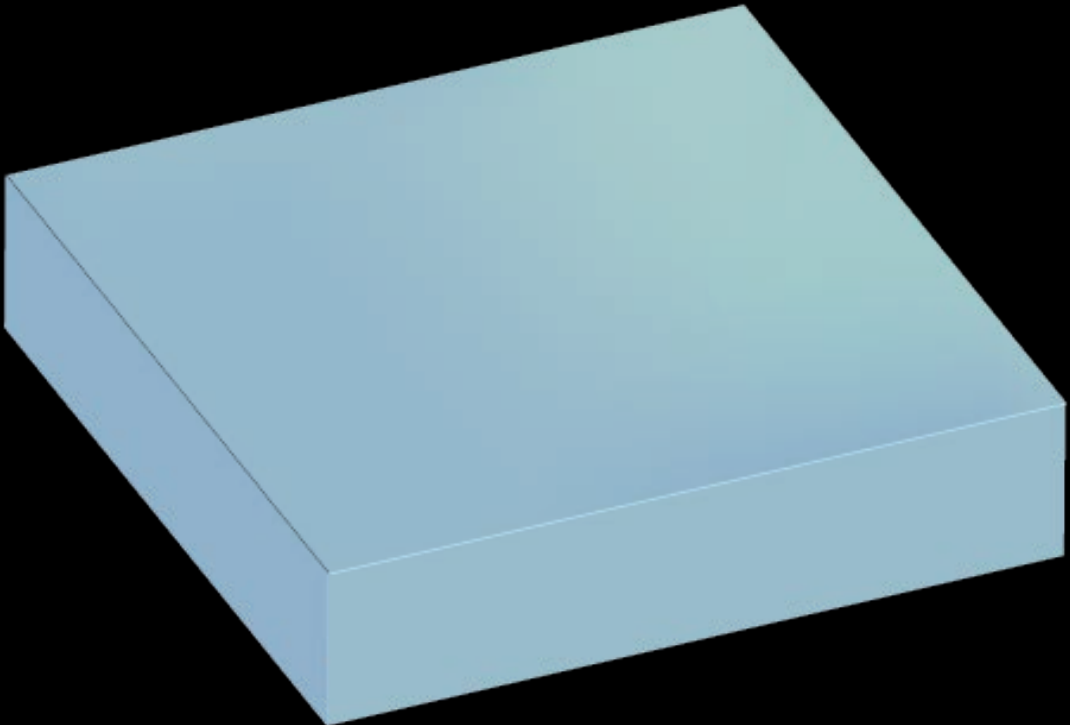
# Fallback Store



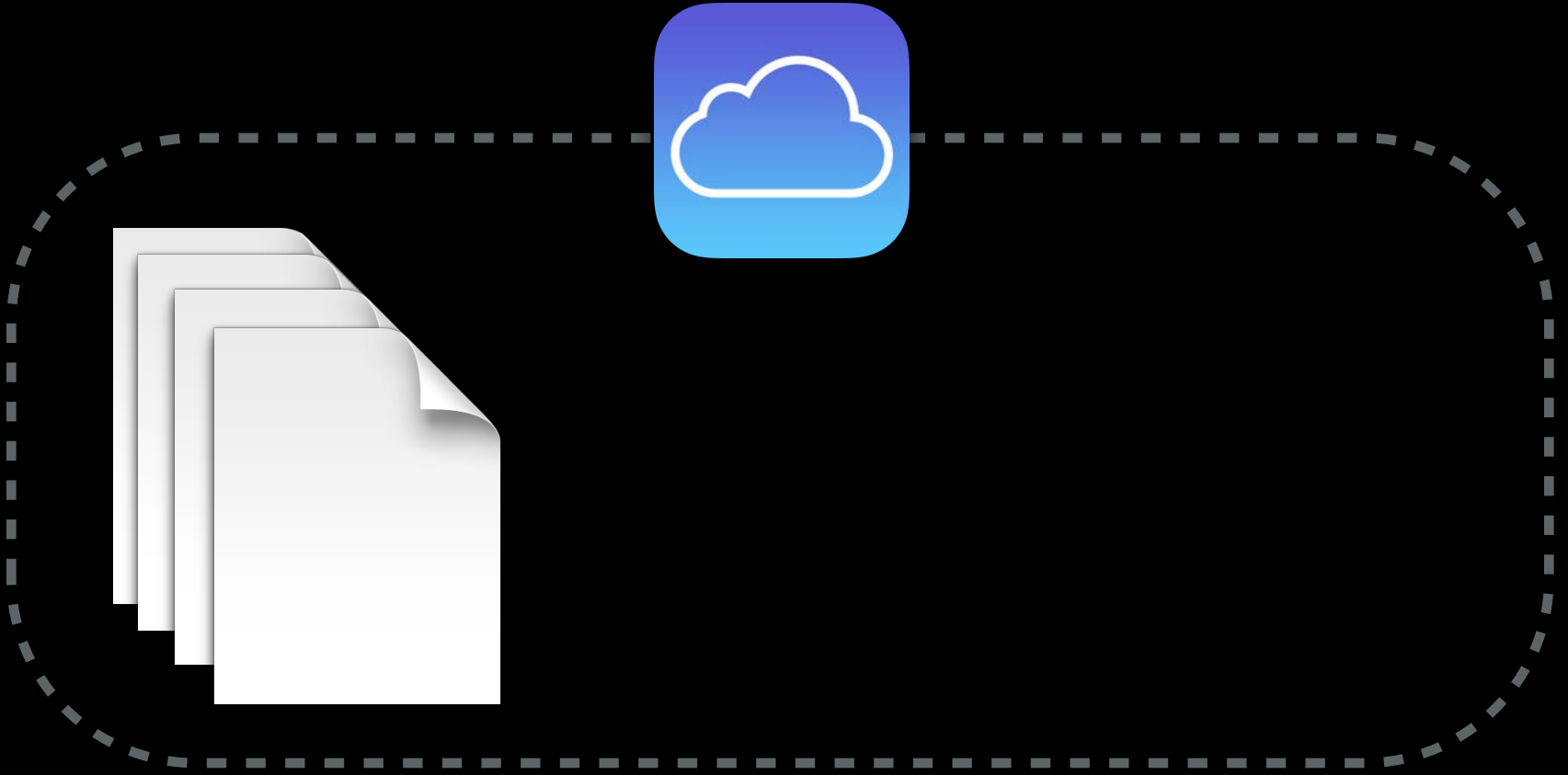
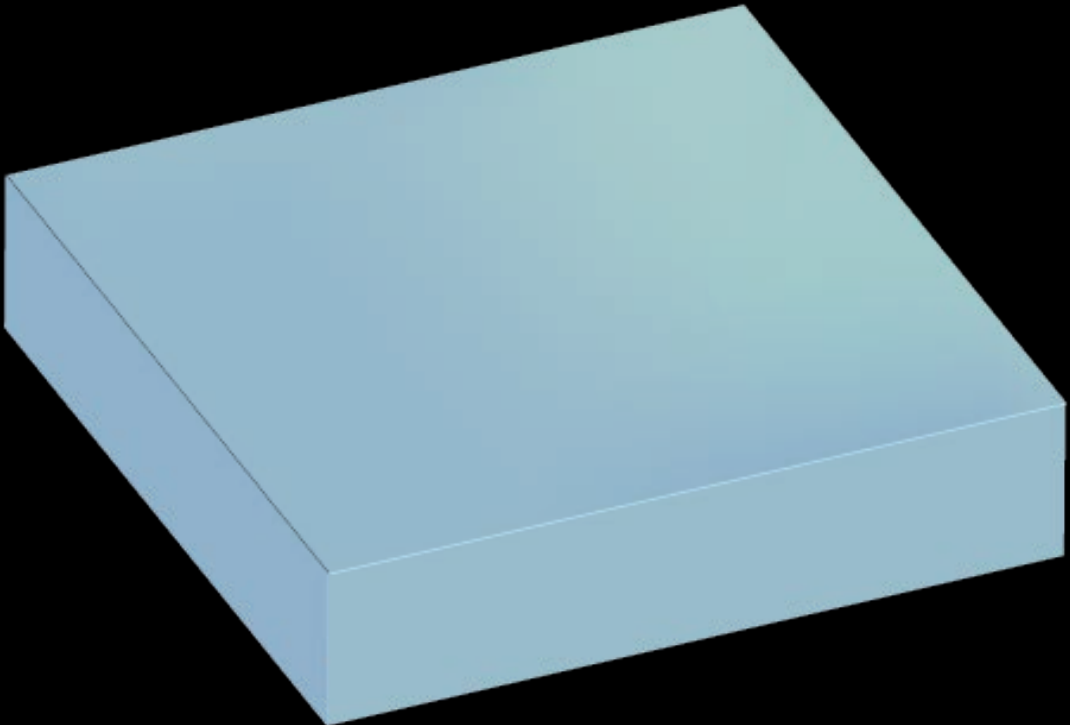
# Fallback Store



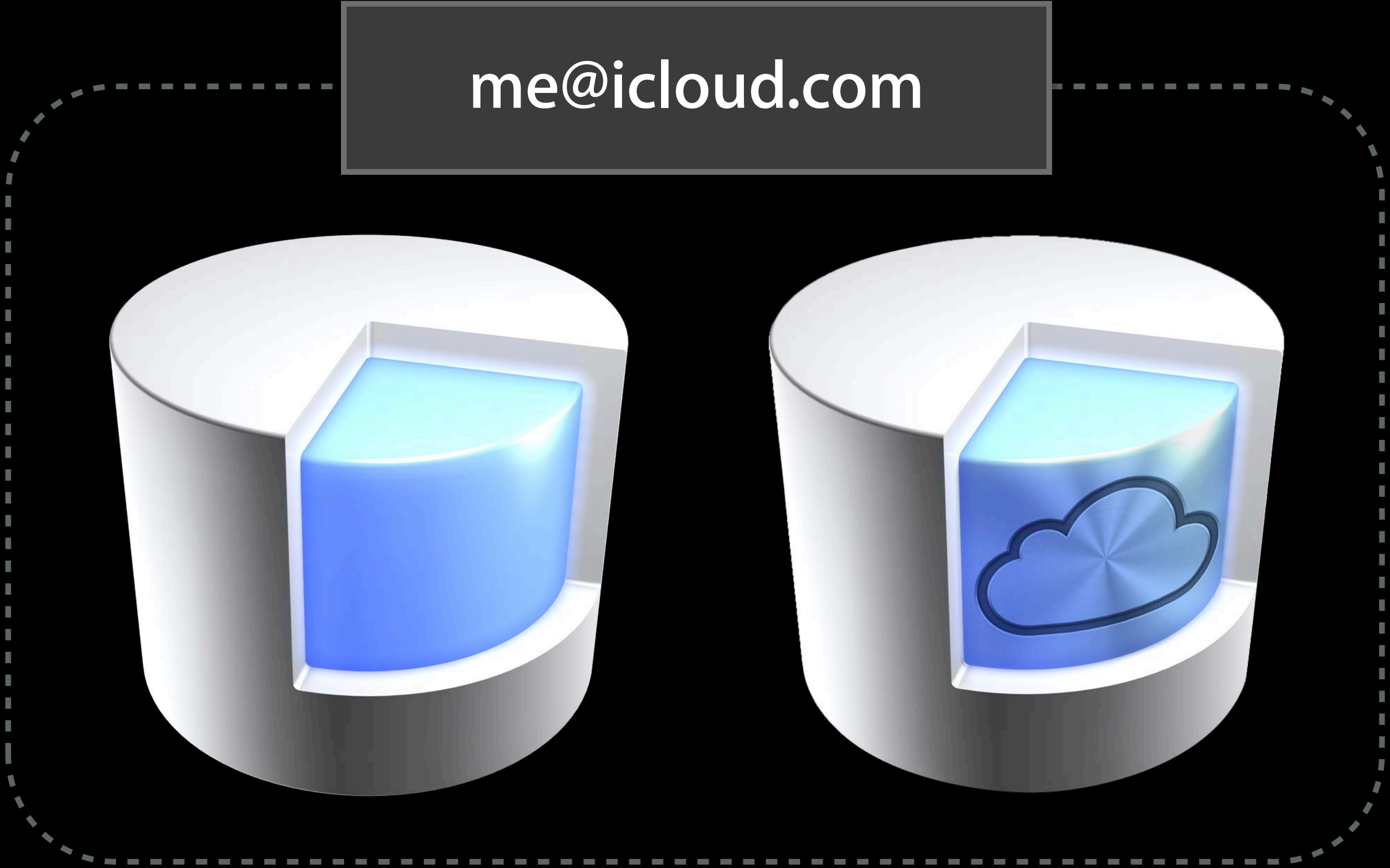
# Fallback Store



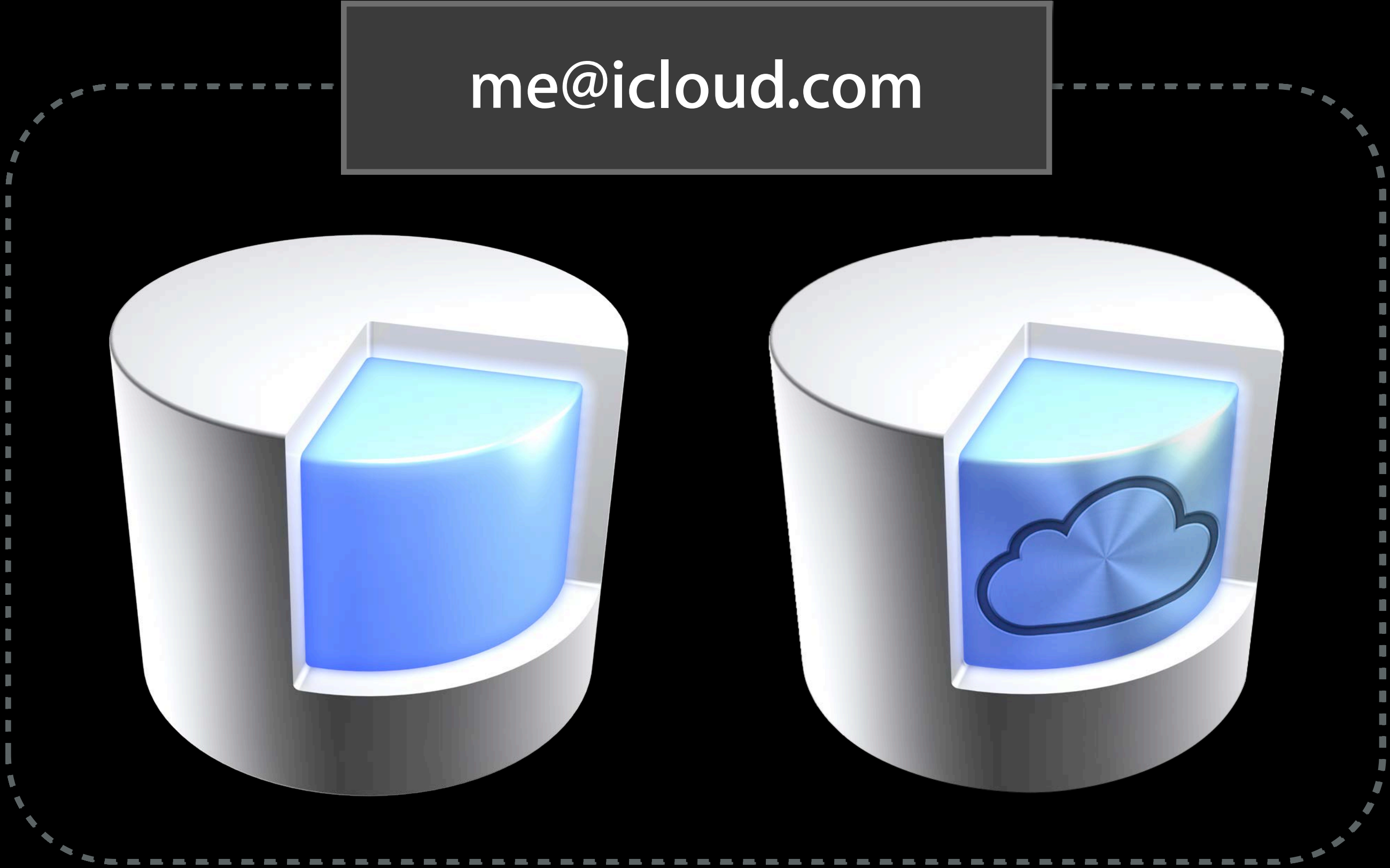
# Fallback Store



# Fallback Store



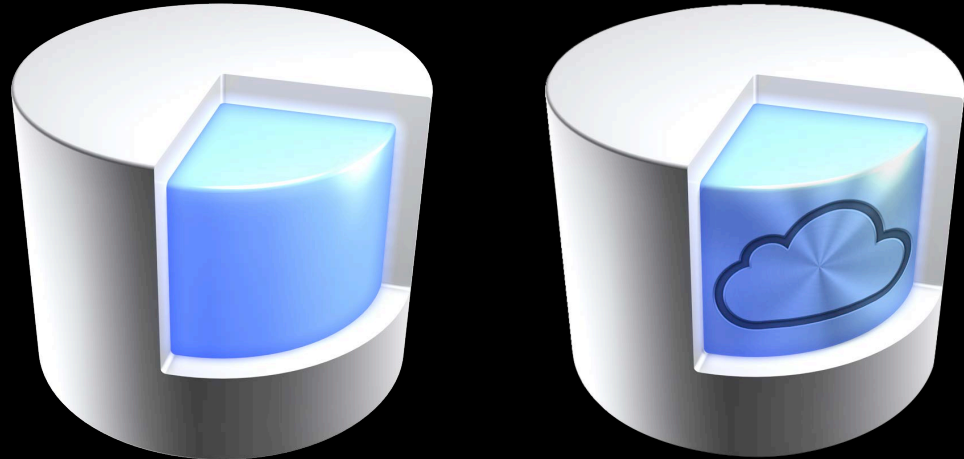
# Fallback Store



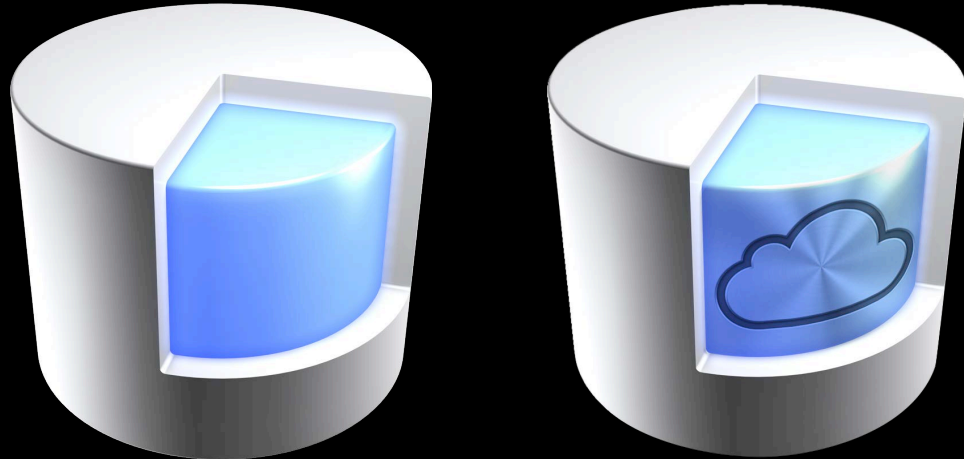


# Fallback Store

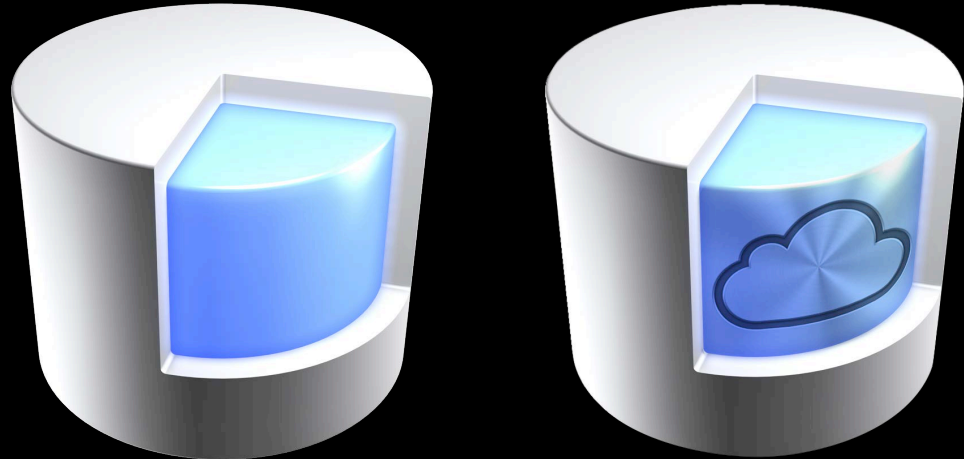
me@icloud.com



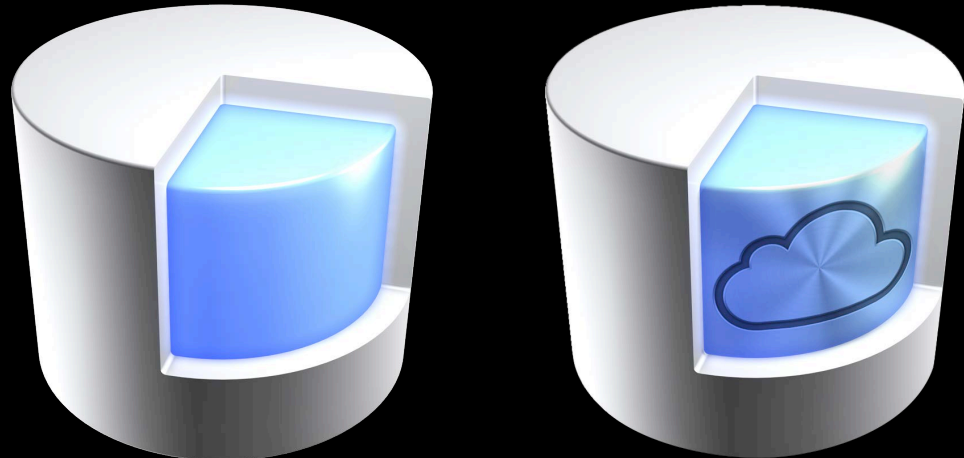
me2@icloud.com



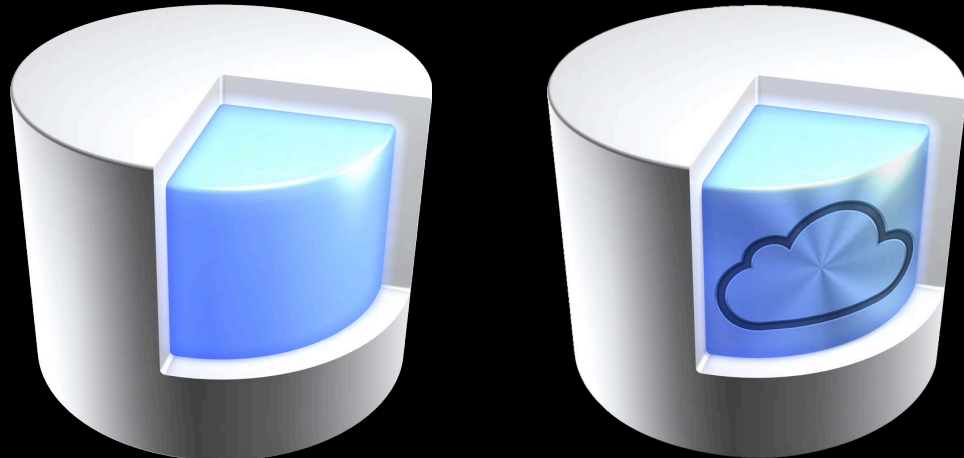
me3@icloud.com



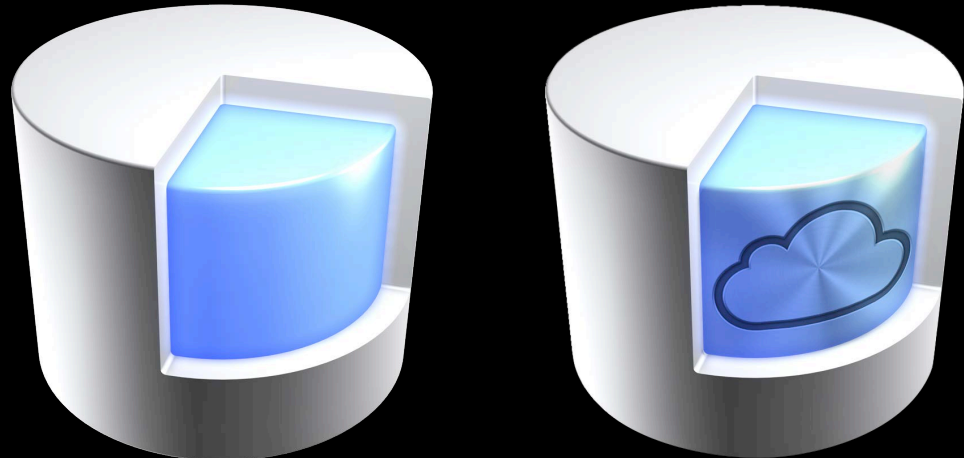
me4@icloud.com



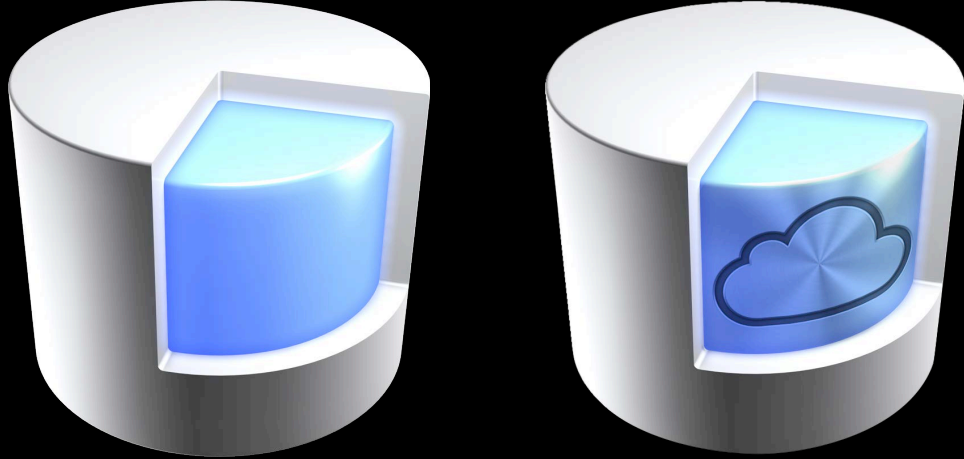
me5@icloud.com



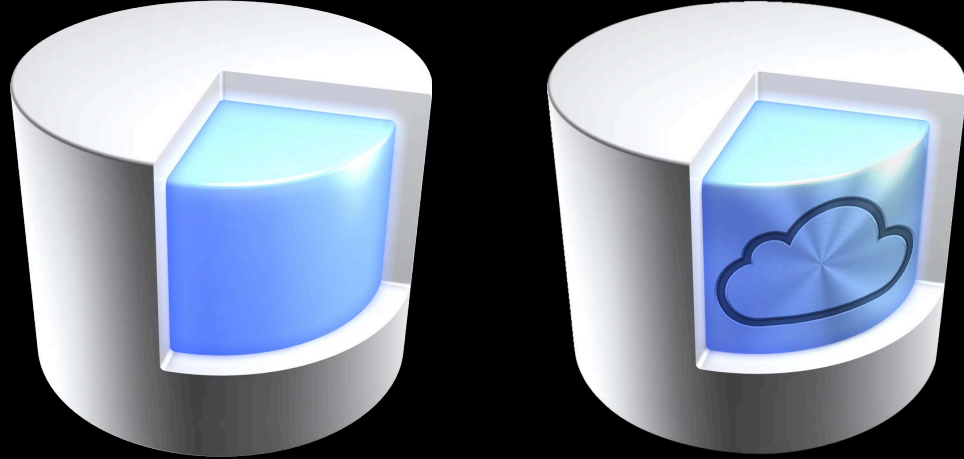
me6@icloud.com



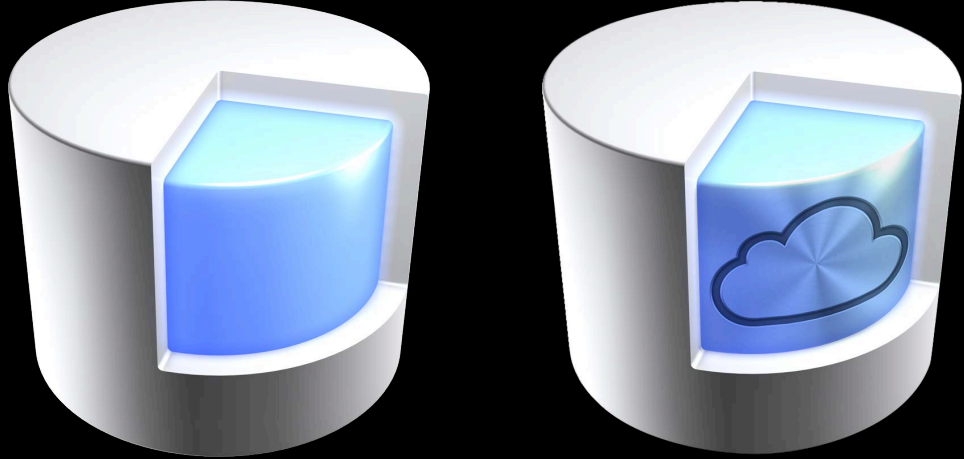
me7@icloud.com



me8@icloud.com



me9@icloud.com



# Fallback Store



# Fallback Store



# Fallback Store

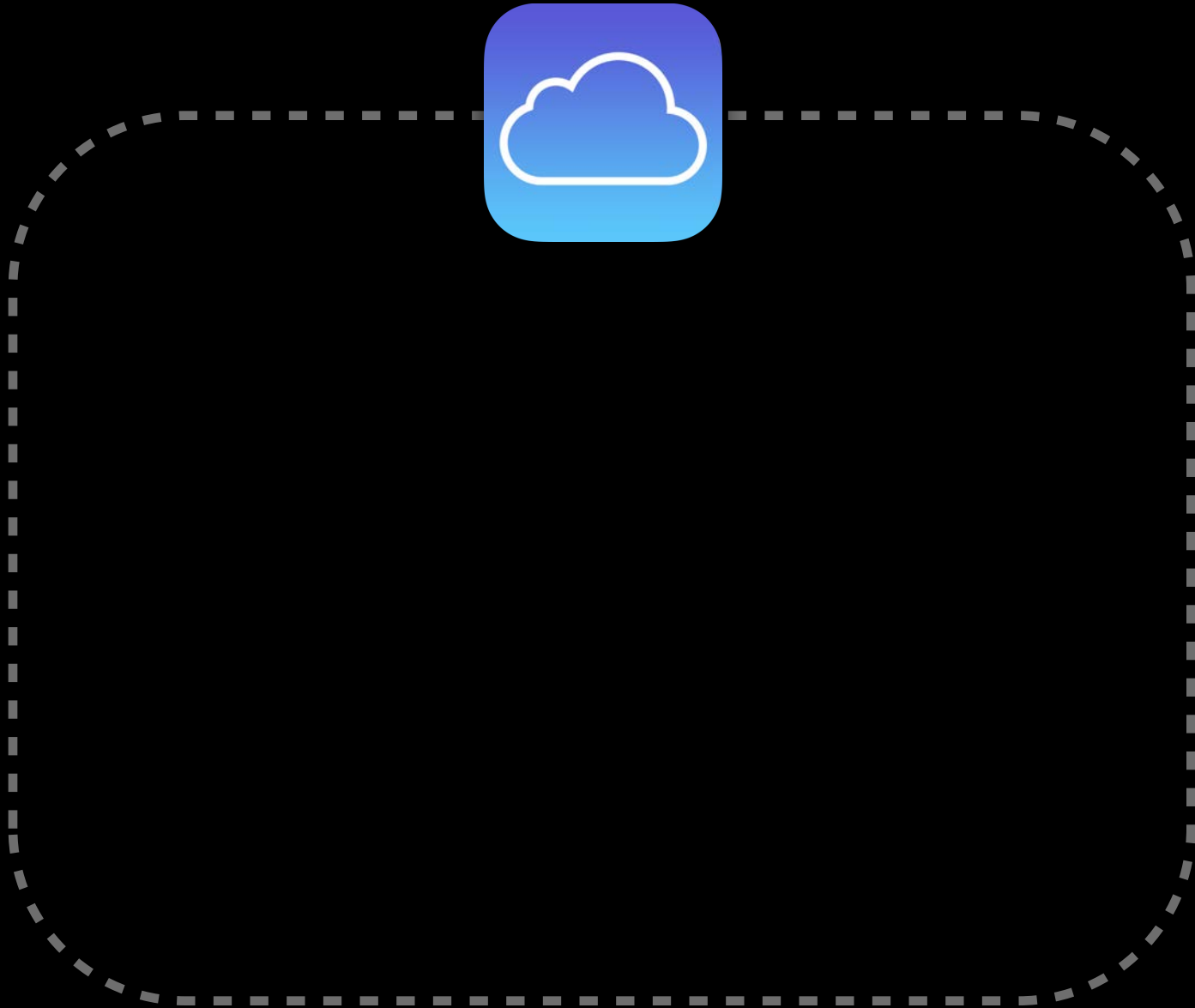
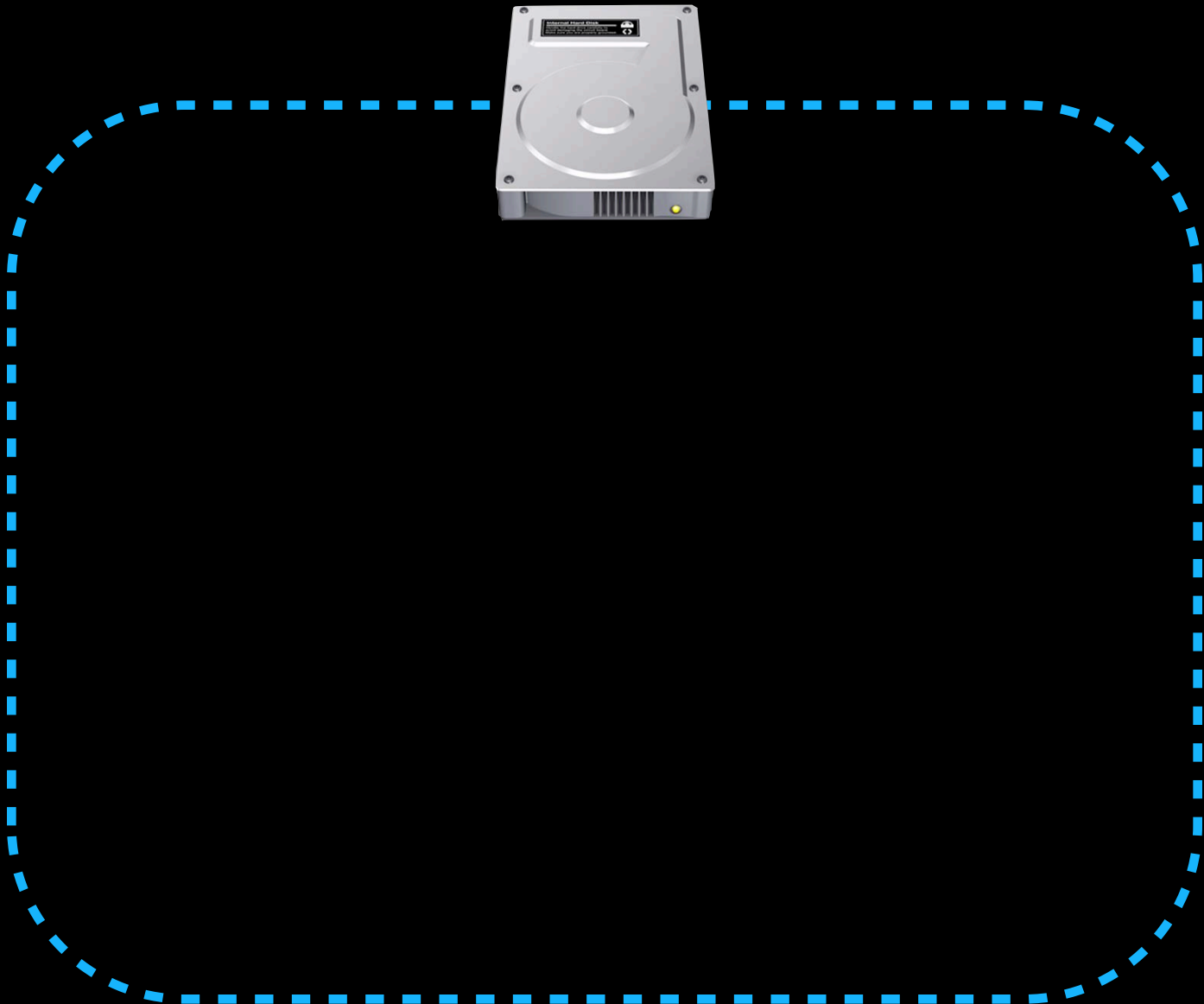
# Fallback Store

- Entirely managed by Core Data

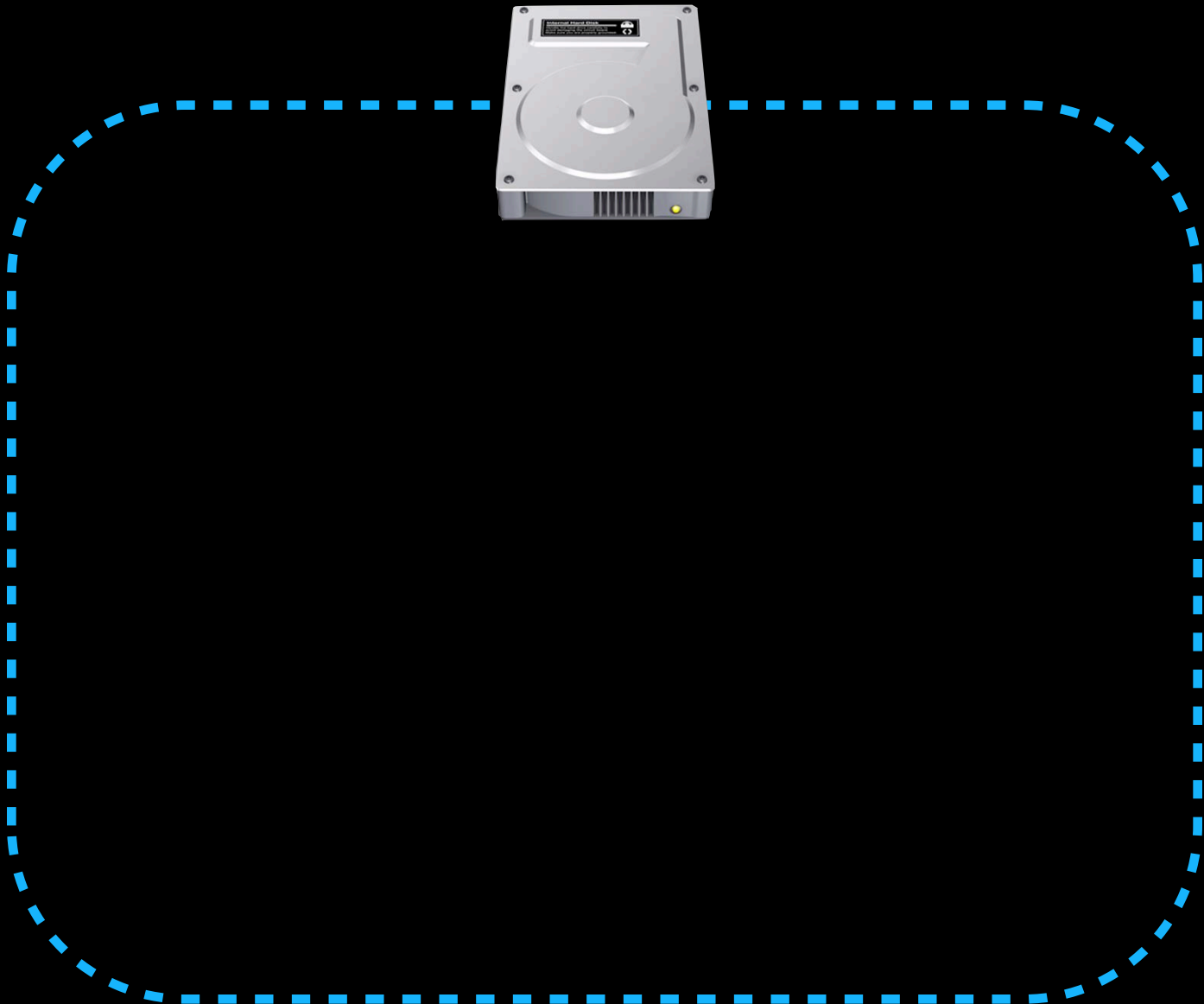
# Fallback Store

- Entirely managed by Core Data
- Only one store file per account

# Fallback Store

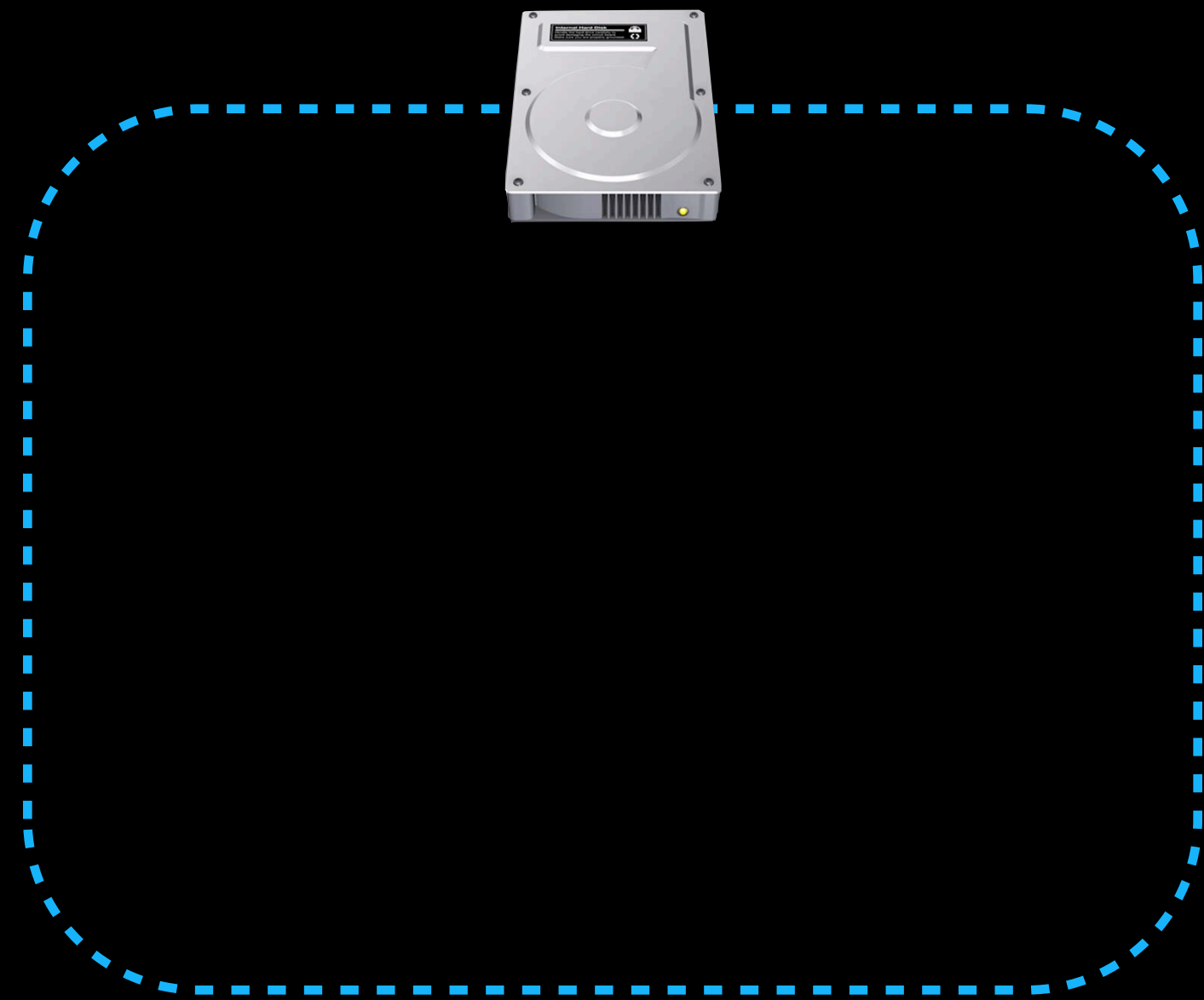


# Fallback Store

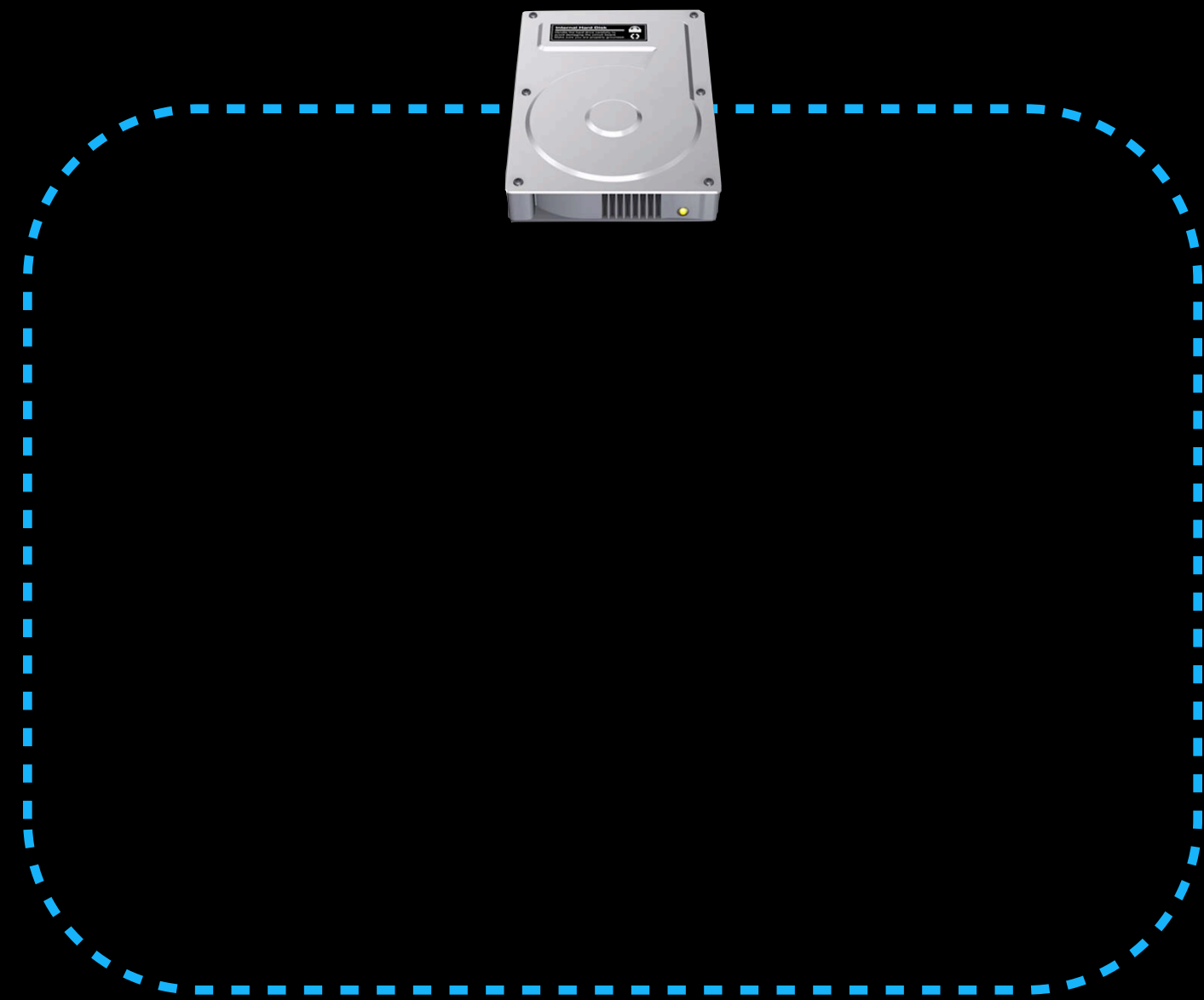




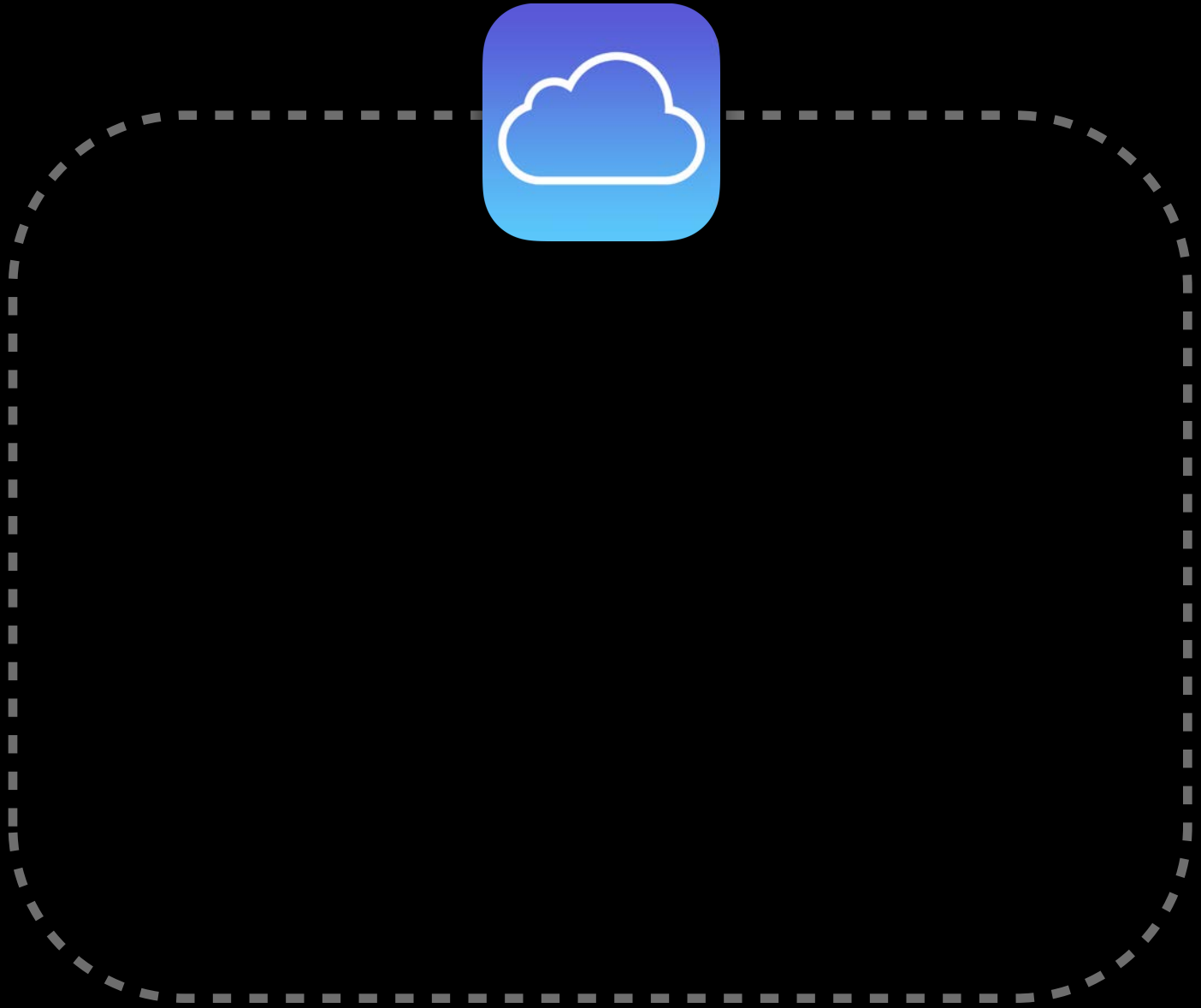
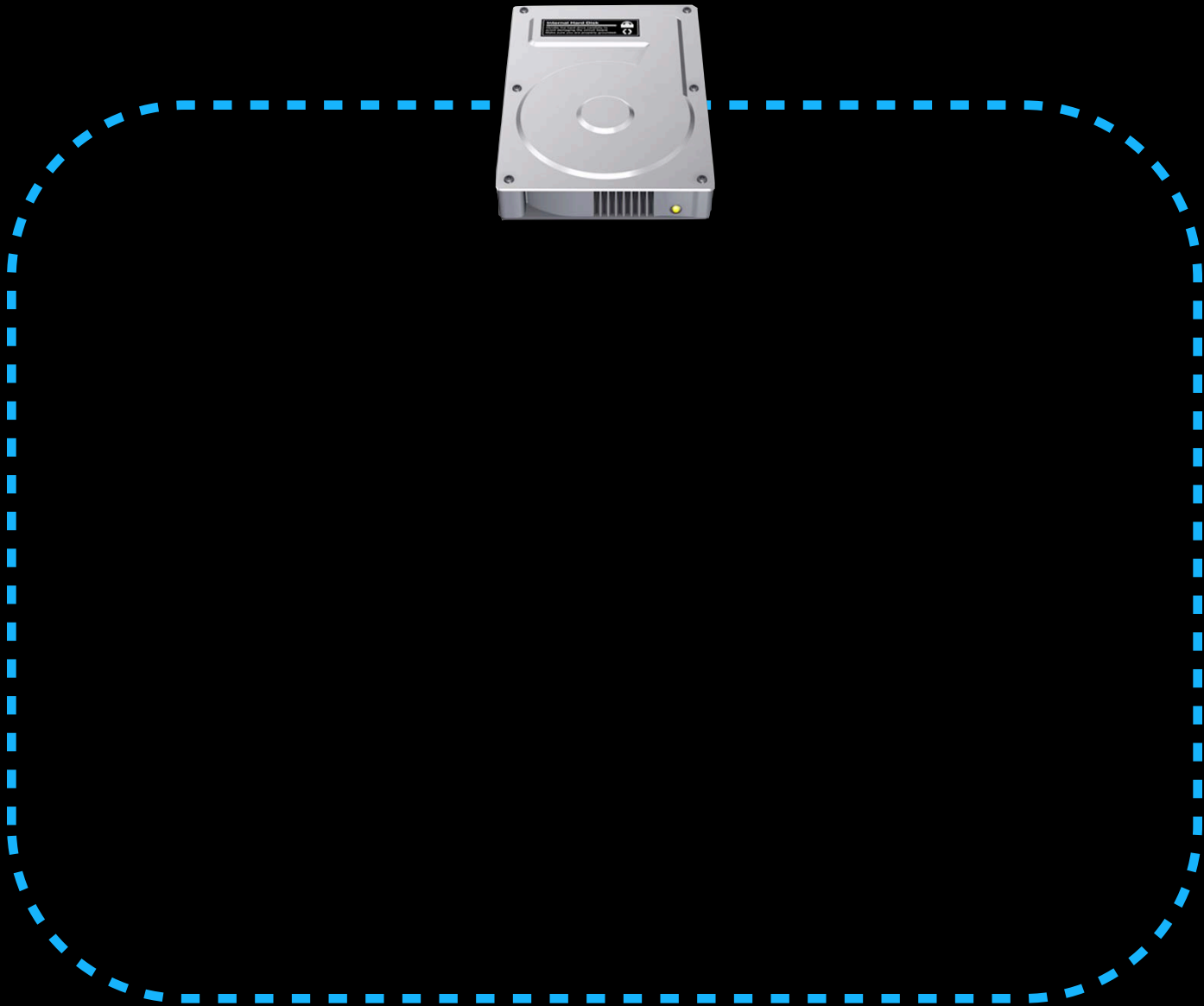
# Fallback Store



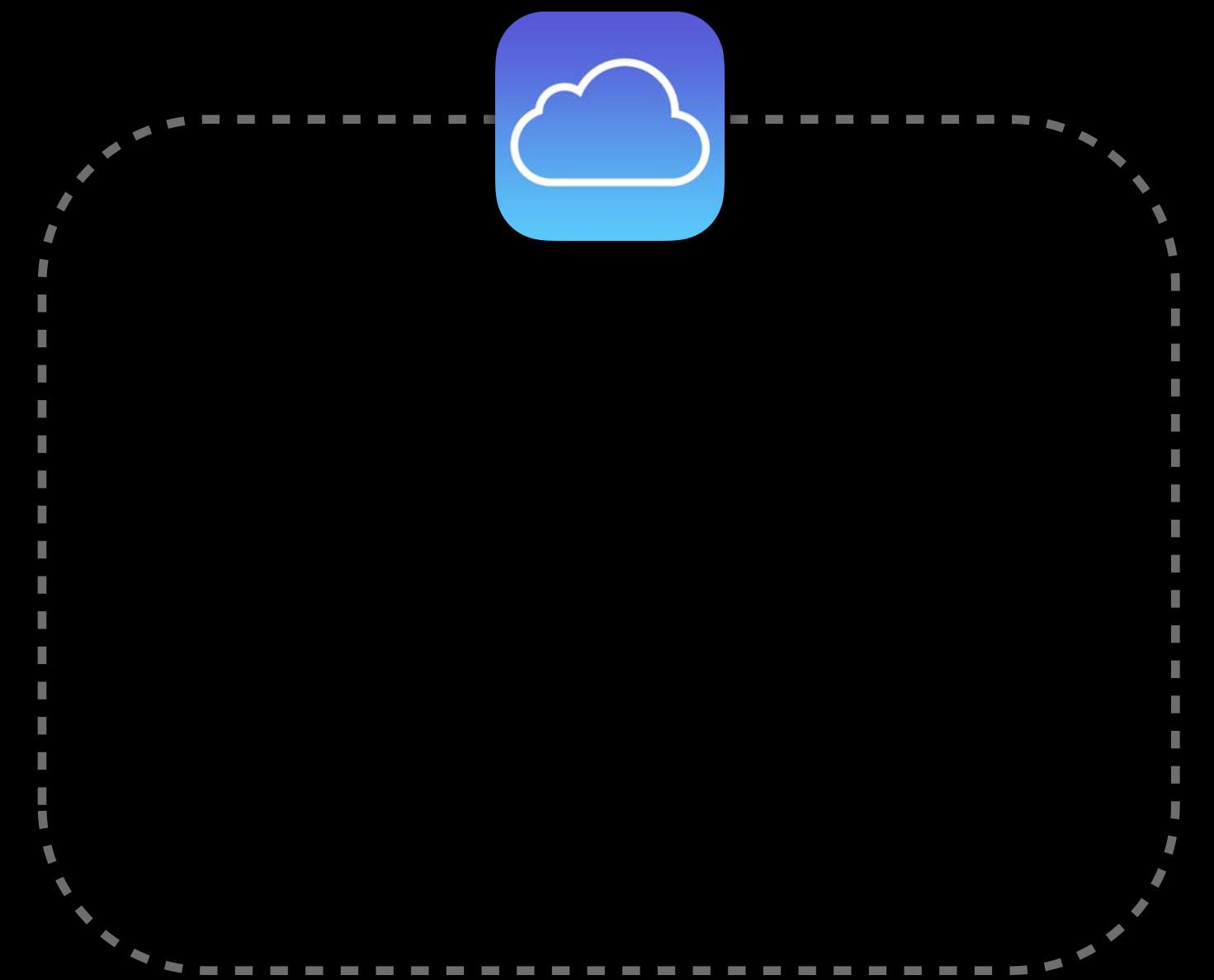
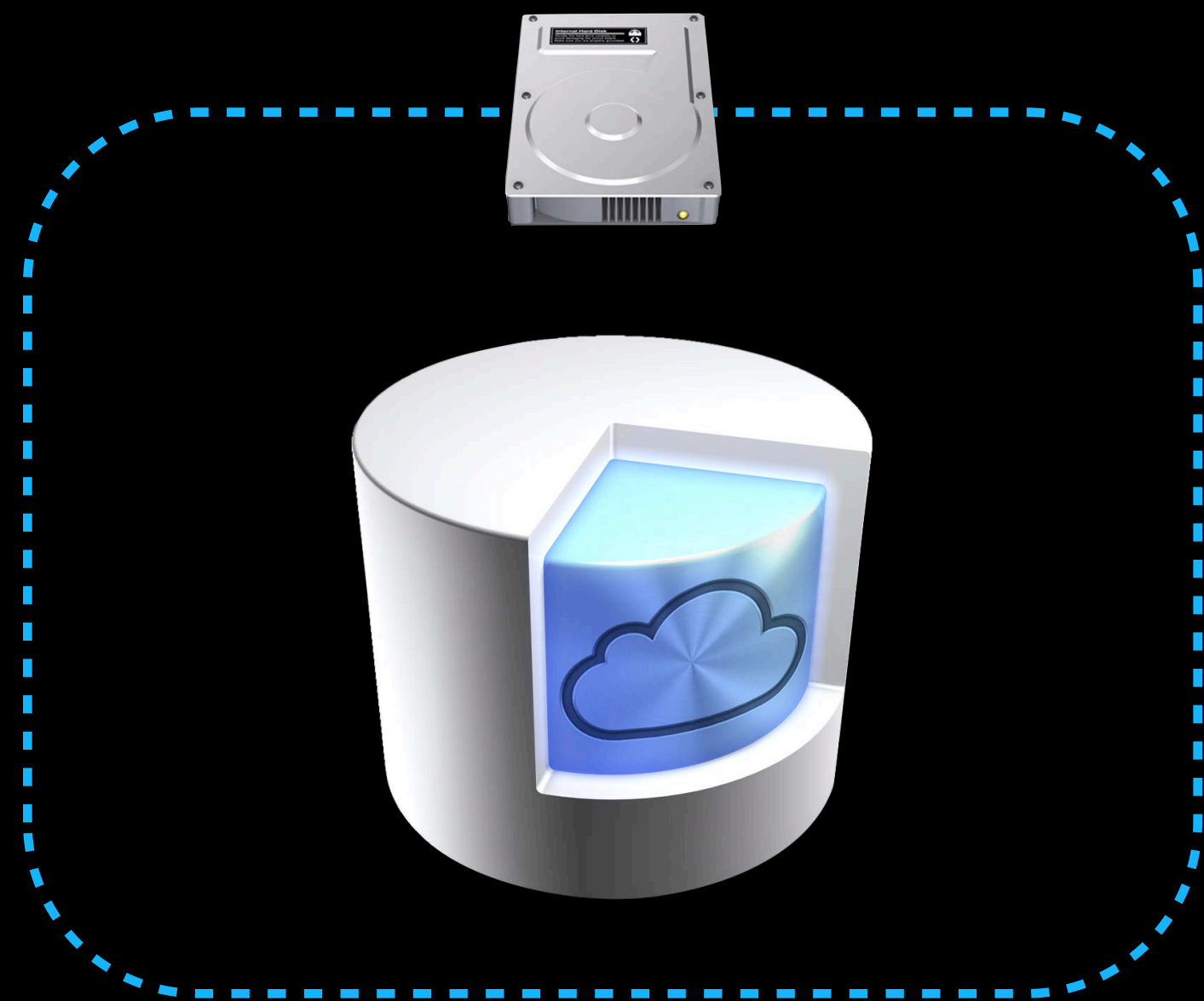
# Fallback Store



# Fallback Store



# Fallback Store



# Fallback Store

- Entirely managed by Core Data
- Only one store file per account

# Fallback Store

- Entirely managed by Core Data
- Only one store file per account
- Must be stored in local storage (e.g. App Sandbox)

# Fallback Store

- Entirely managed by Core Data
- Only one store file per account
- Must be stored in local storage (e.g. App Sandbox)
- Events logged to console

# Fallback Store

- Entirely managed by Core Data
- Only one store file per account
- Must be stored in local storage (e.g. App Sandbox)
- Events logged to console

```
Core Data: Ubiquity: peerID:StoreName - Using local storage: 1
```



# Fallback Store

- Entirely managed by Core Data
- Only one store file per account
- Must be stored in local storage (e.g. App Sandbox)
- Events logged to console

```
Core Data: Ubiquity: peerID:StoreName - Using local storage: 1
```

```
Core Data: Ubiquity: peerID:StoreName - Using local storage: 0
```

# Fallback Store

- Entirely managed by Core Data
- Only one store file per account
- Must be stored in local storage (e.g. App Sandbox)
- Events logged to console

```
Core Data: Ubiquity: peerID:StoreName - Using local storage: 1
```

```
Core Data: Ubiquity: peerID:StoreName - Using local storage: 0
```

# Fallback Store

- Entirely managed by Core Data
- Only one store file per account
- Must be stored in local storage (e.g. App Sandbox)
- Events logged to console

```
Core Data: Ubiquity: peerID:StoreName - Using local storage: 1
```

```
Core Data: Ubiquity: peerID:StoreName - Using local storage: 0
```

# Asynchronous Setup

# Asynchronous Setup

```
[psc addPersistentStoreWithType:NSSQLiteStoreType  
    configuration:nil  
    URL:url  
    options:options  
    error:&error];
```

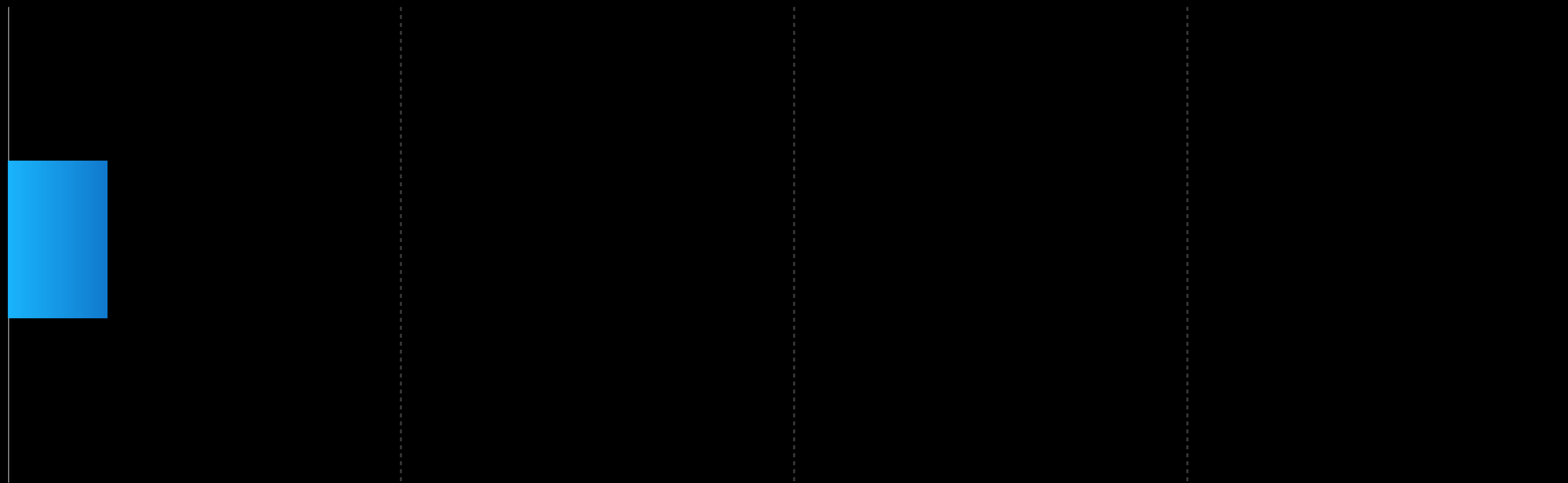
# Asynchronous Setup

```
[psc addPersistentStoreWithType:NSSQLiteStoreType  
    configuration:nil  
    URL:url  
    options:options  
    error:&error];
```



# Asynchronous Setup

# Asynchronous Setup



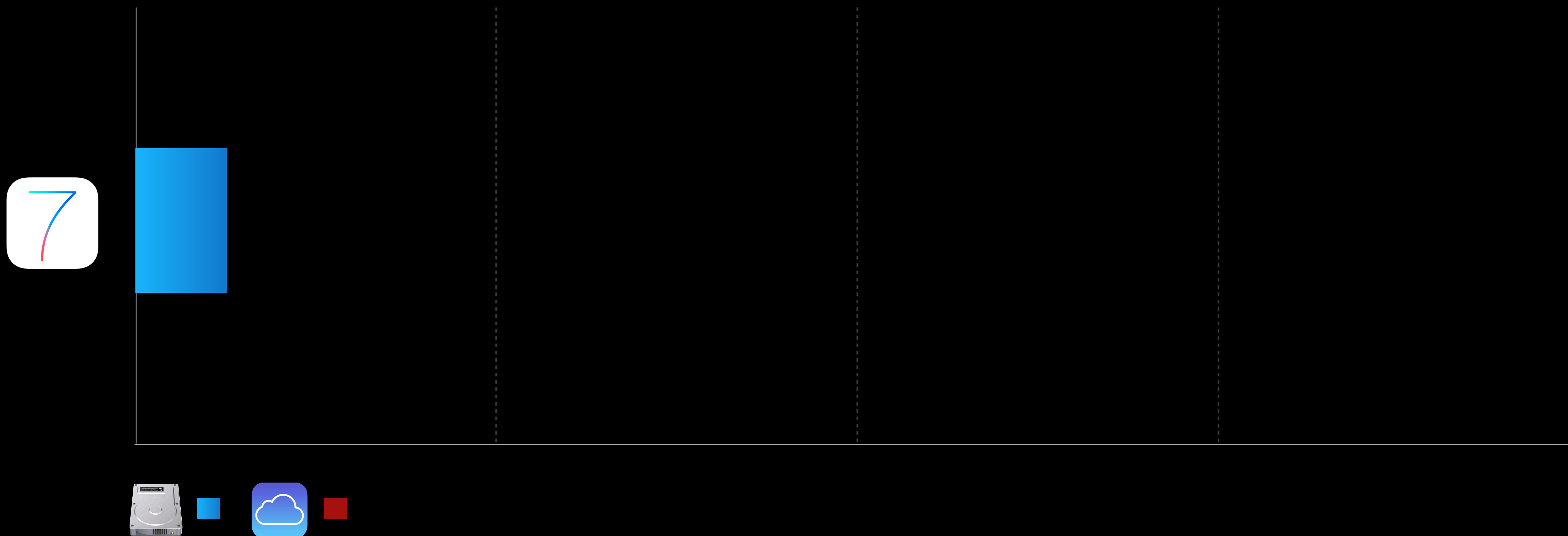


# Asynchronous Setup

6



# Asynchronous Setup



# Asynchronous Setup

```
[psc addPersistentStoreWithType:NSSQLiteStoreType  
    configuration:nil  
    URL:url  
    options:options  
    error:&error];
```



# Asynchronous Setup



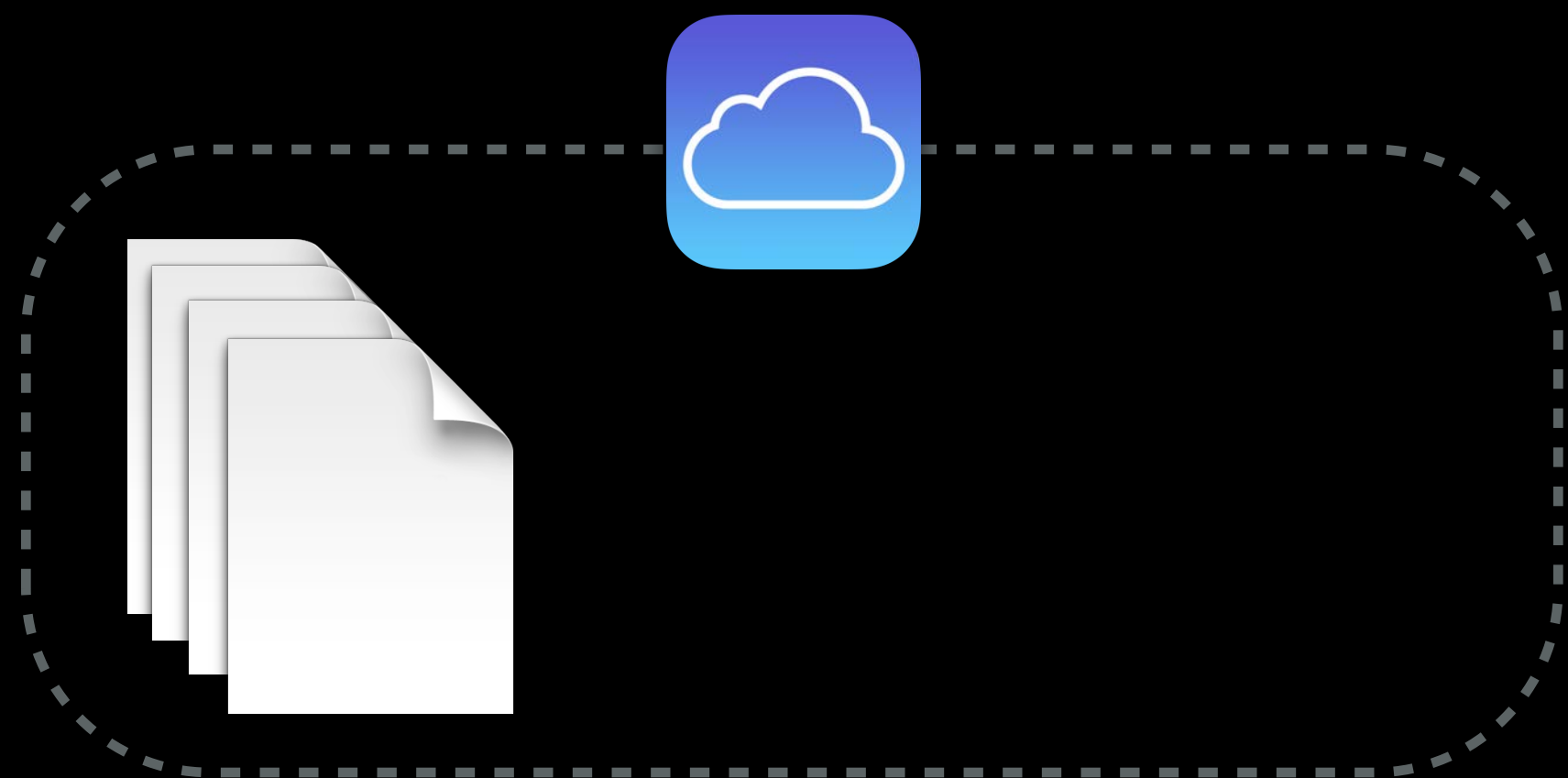
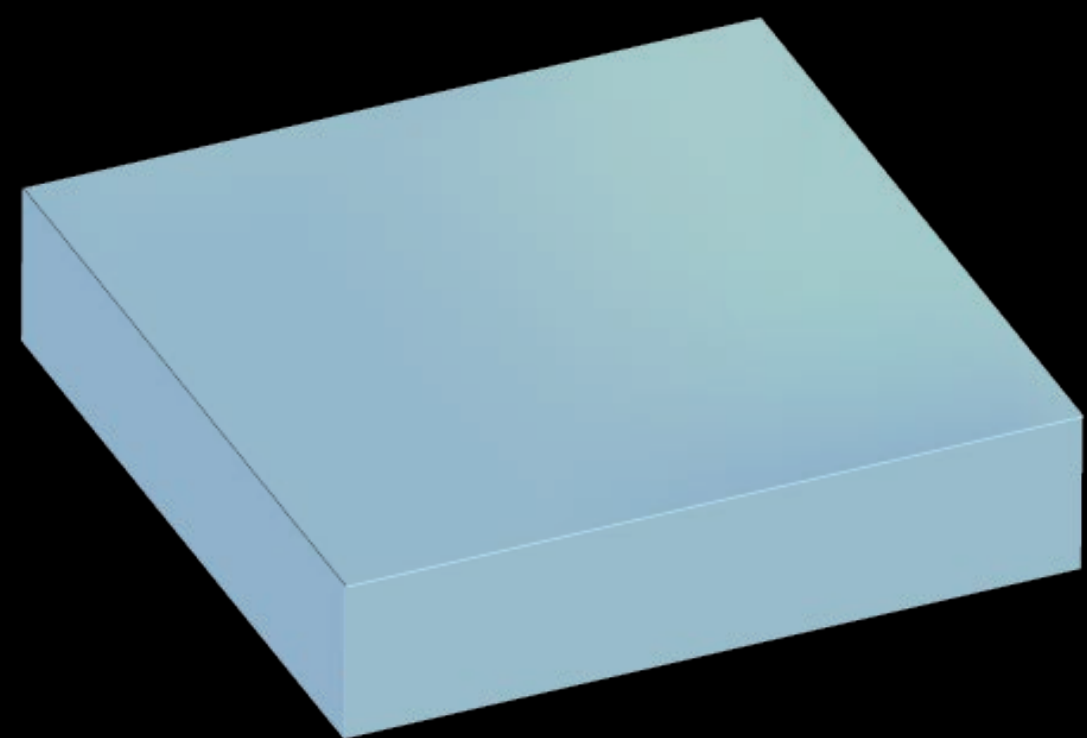
# Asynchronous Setup



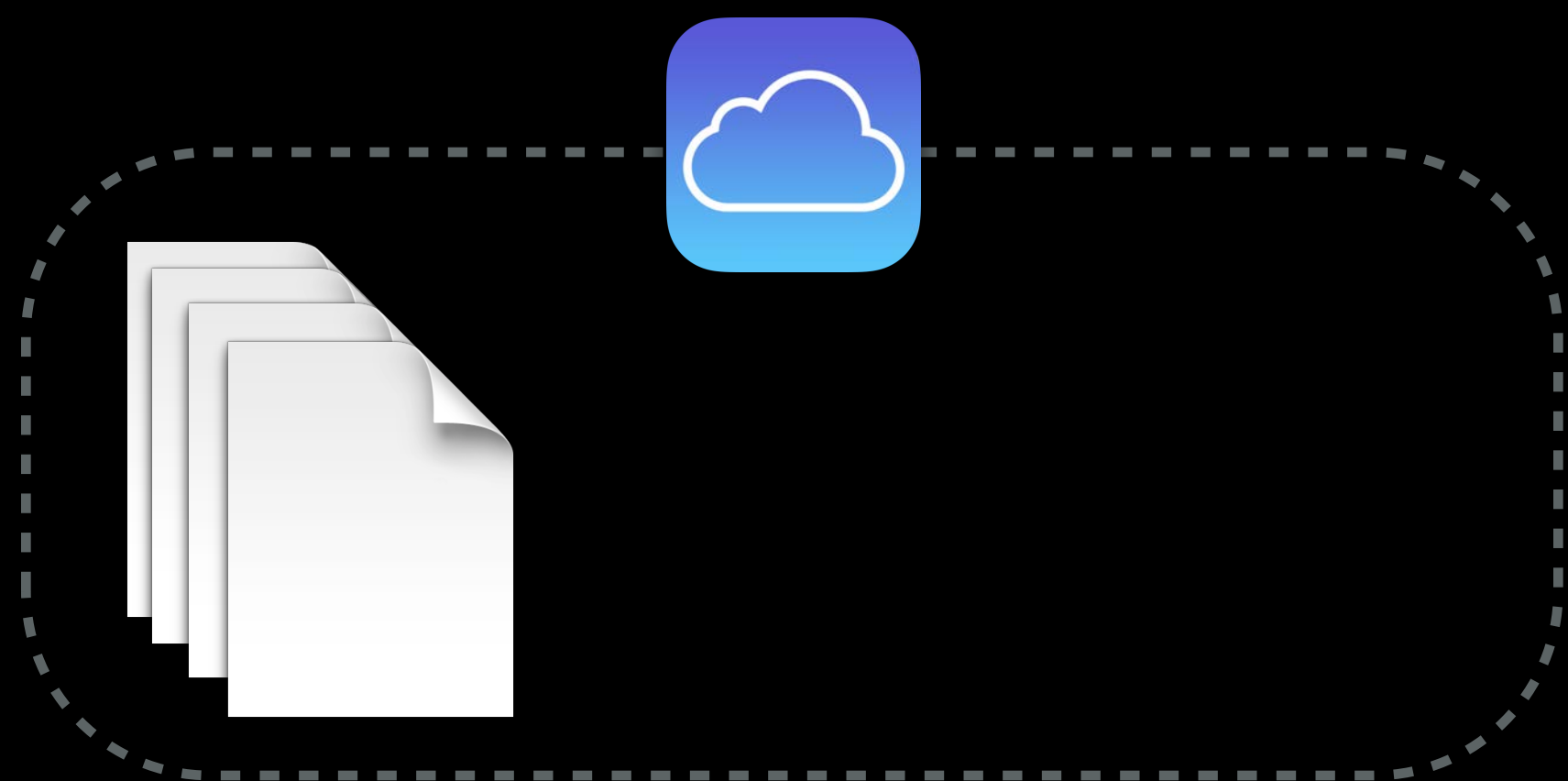
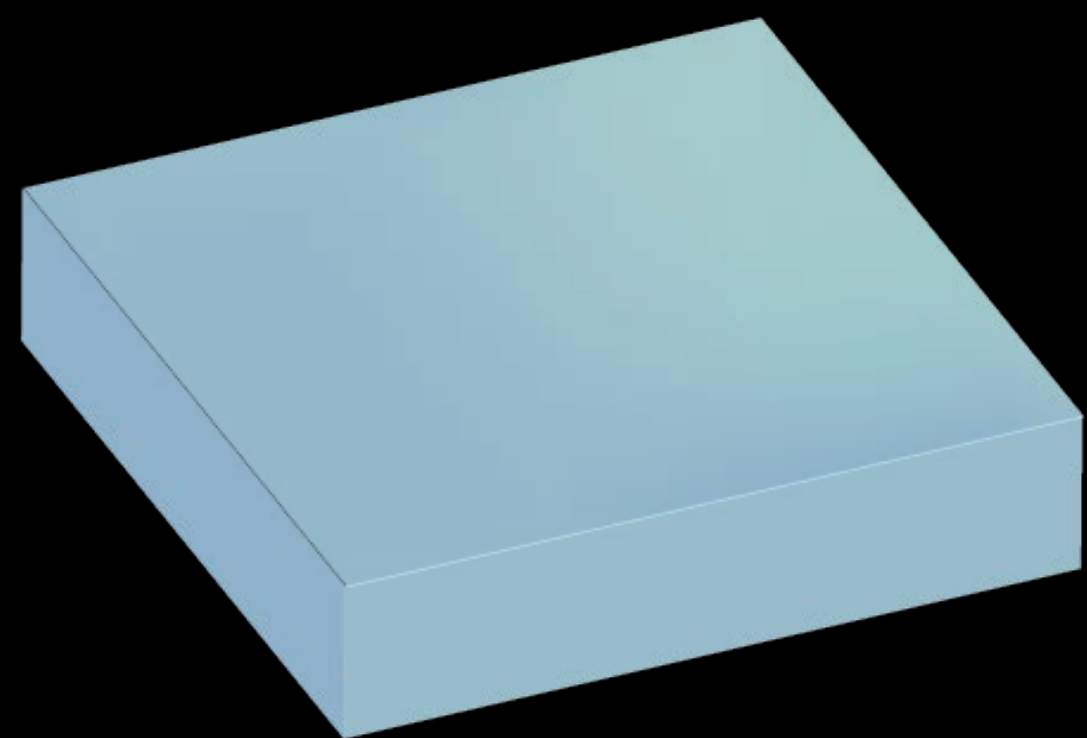
# Asynchronous Setup



# Asynchronous Setup

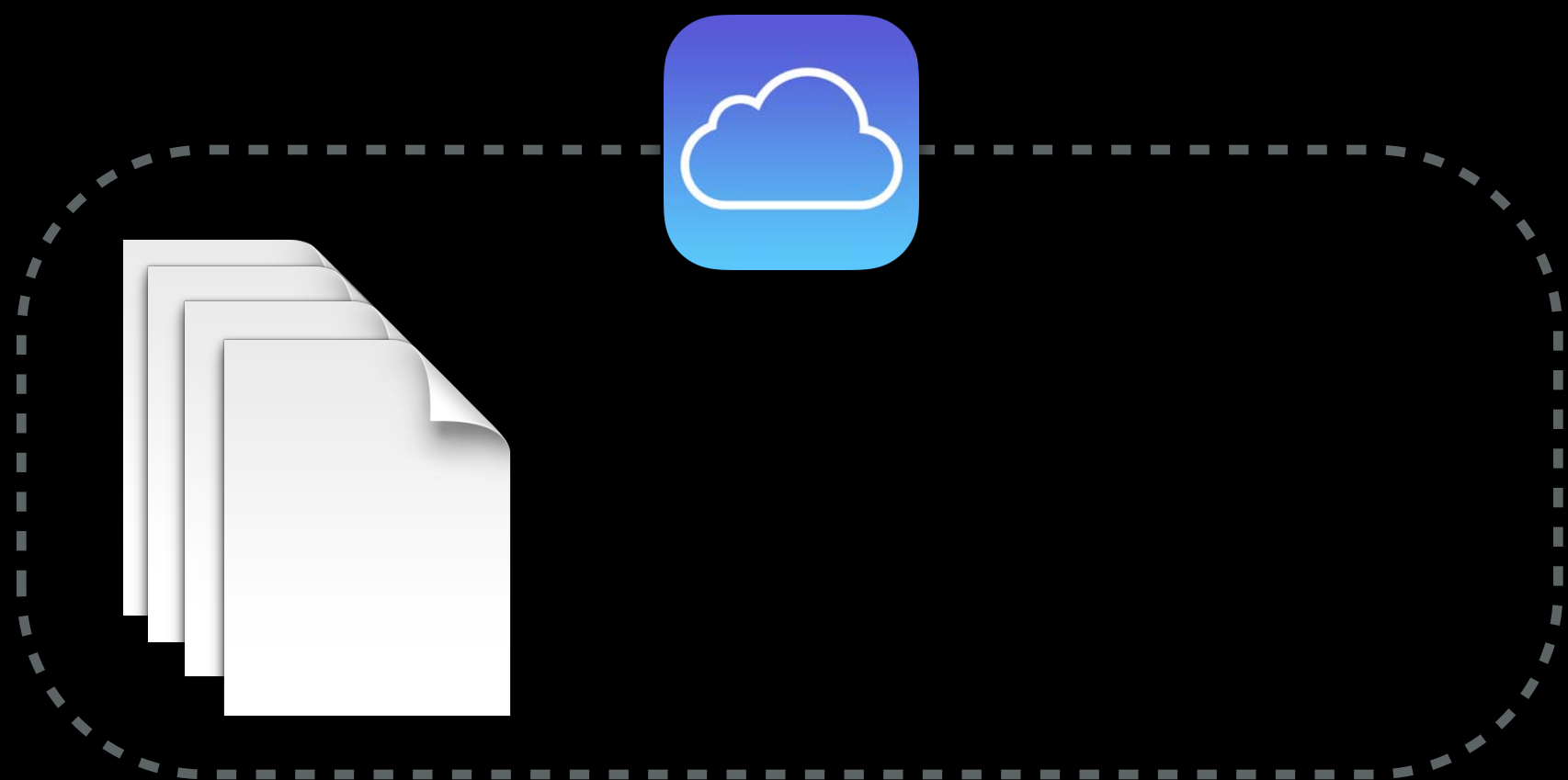
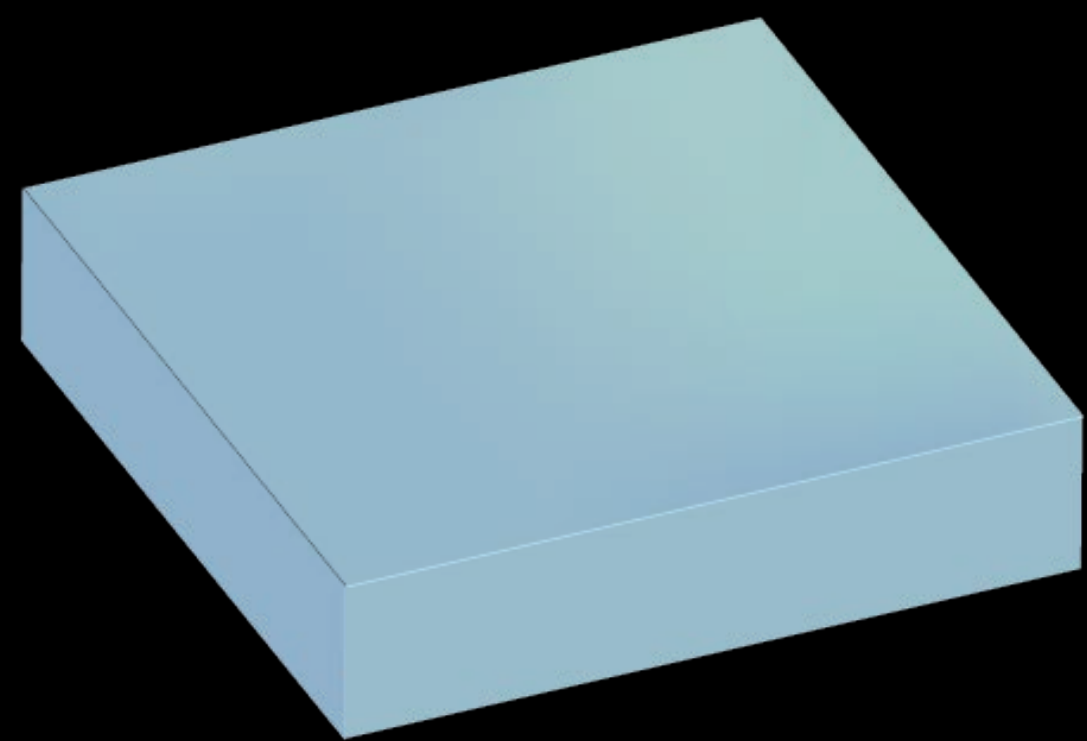


# Asynchronous Setup

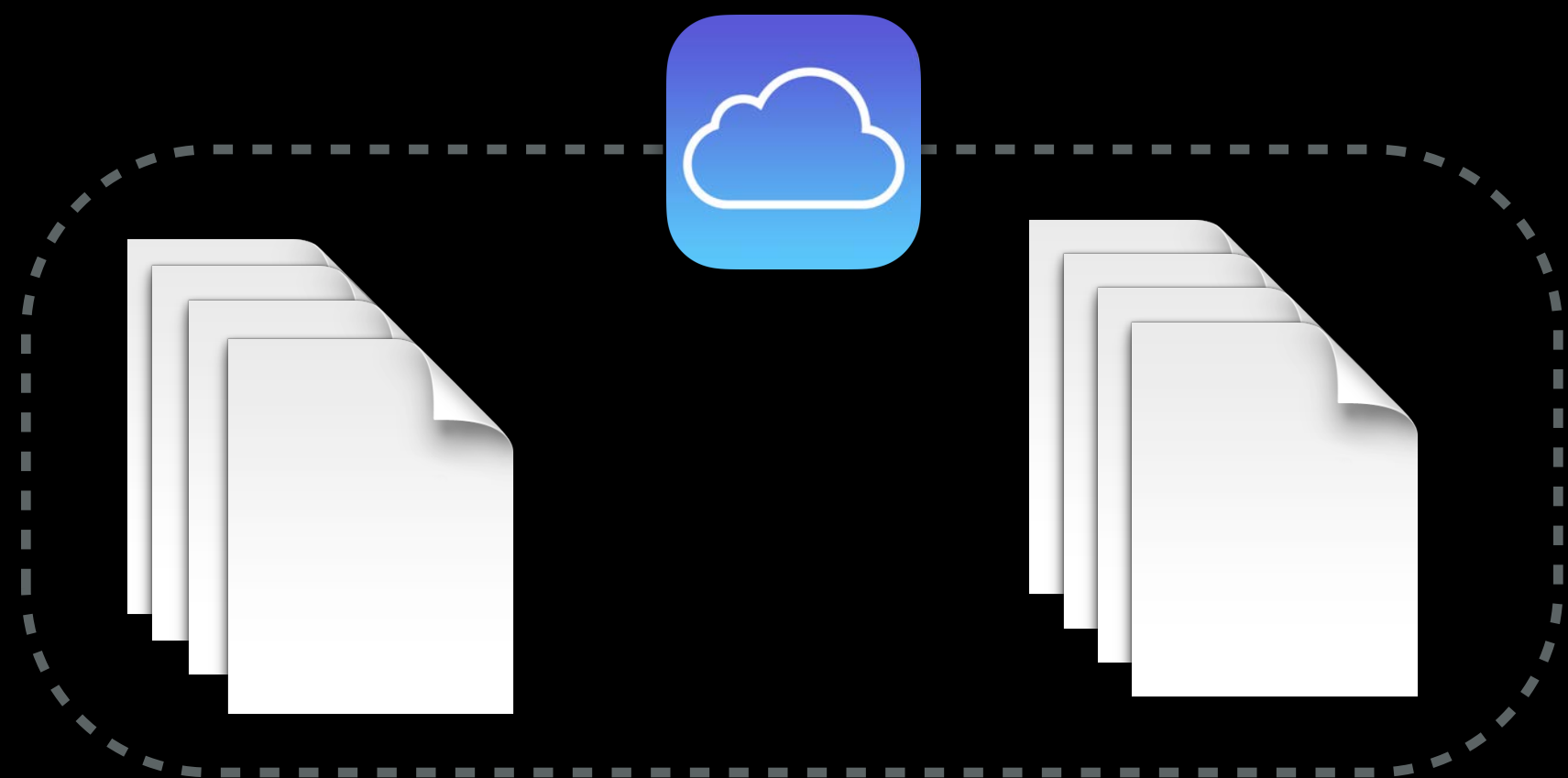
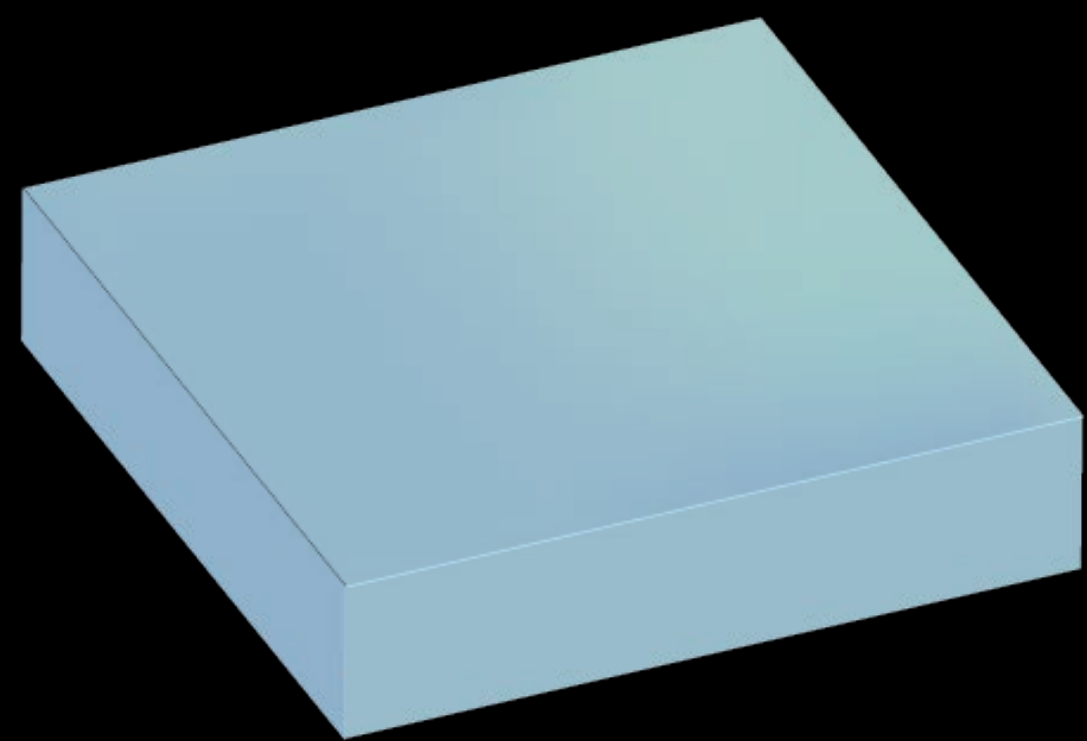




# Asynchronous Setup



# Asynchronous Setup



# Asynchronous Setup

`NSPersistentStoreCoordinatorStoresWillChangeNotification`



# Asynchronous Setup

NSPersistentStoreCoordinatorStoresWillChangeNotification

- [NSManagedObjectContext save:]
- [NSManagedObjectContext reset:]



# Asynchronous Setup

`NSPersistentStoreCoordinatorStoresWillChangeNotification`

- `-[NSManagedObjectContext save:]`
- `-[NSManagedObjectContext reset:]`

Store Removed by Core Data



# Asynchronous Setup

`NSPersistentStoreCoordinatorStoresWillChangeNotification`

- `[NSManagedObjectContext save:]`
- `[NSManagedObjectContext reset:]`

Store Removed by Core Data

`NSPersistentStoreCoordinatorStoresDidChangeNotification`



# Asynchronous Setup

`NSPersistentStoreCoordinatorStoresWillChangeNotification`

- `-[NSManagedObjectContext save:]`
- `-[NSManagedObjectContext reset:]`

Store Removed by Core Data

`NSPersistentStoreCoordinatorStoresDidChangeNotification`

- `-[NSManagedObjectContext save:]`



# Asynchronous Setup



# Asynchronous Setup

```
NSPersistentStoreDidImportUbiquitousContentChangesNotification
```

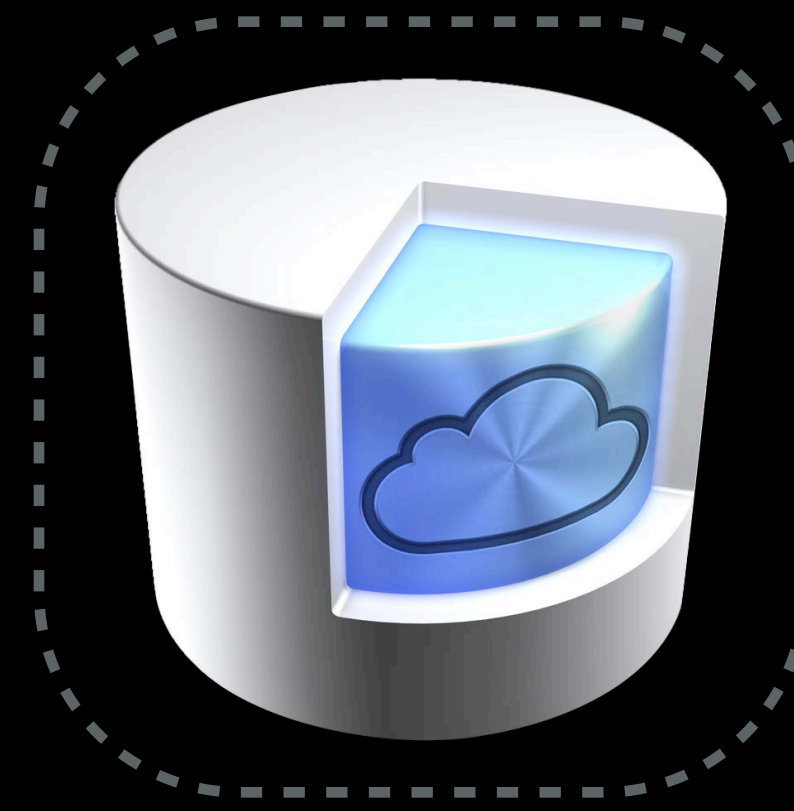
```
–[NSManagedObjectContext mergeChangesFromContextDidSaveNotification:]
```

# Account Changes

# Handling Account Changes



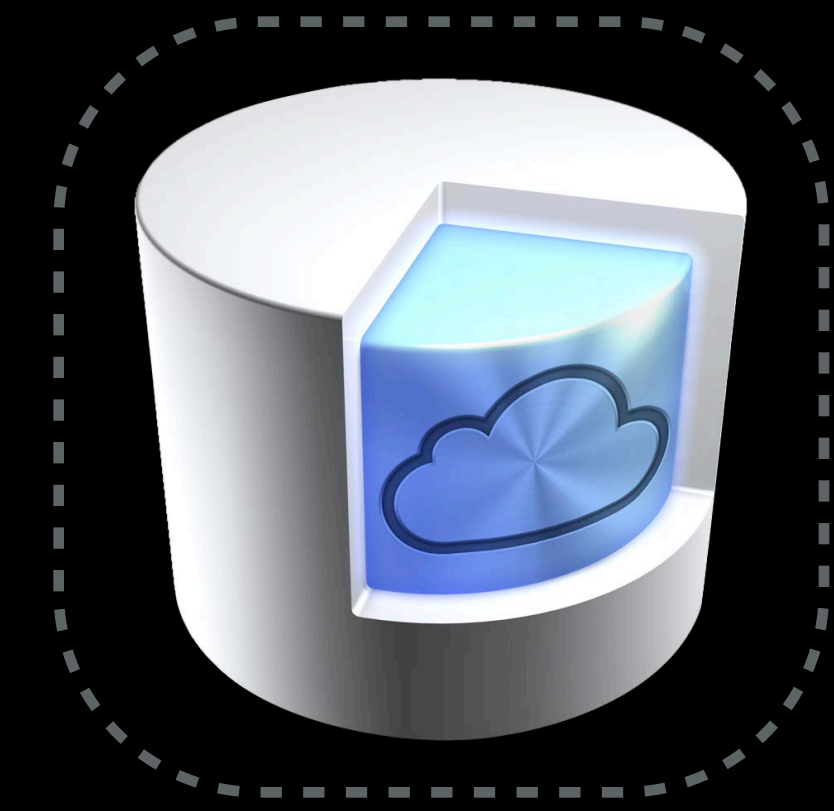
me@icloud.com



# Handling Account Changes



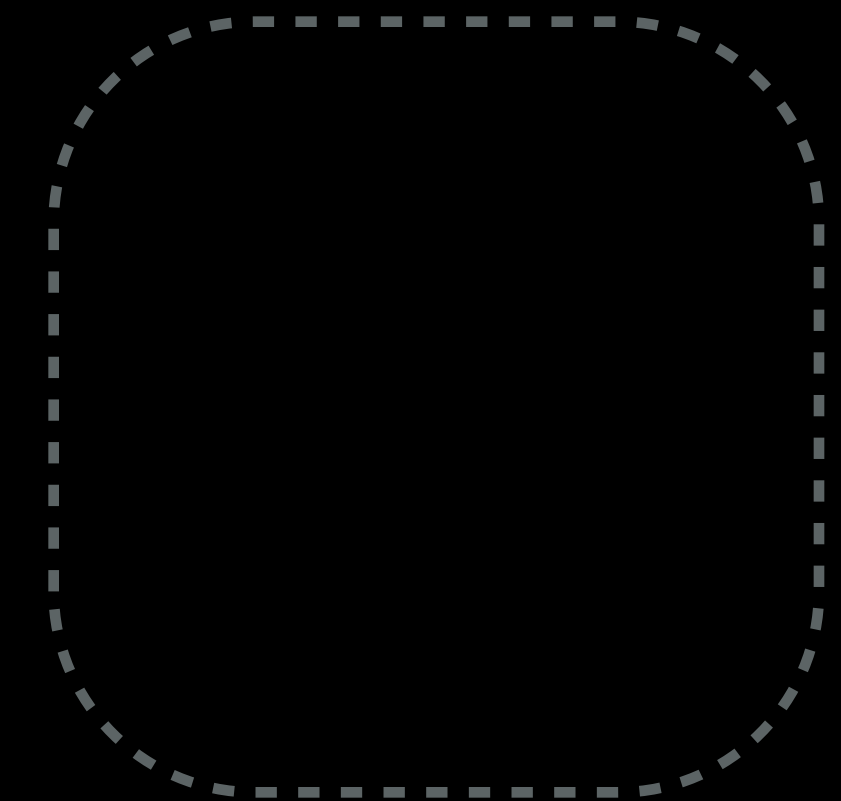
`NSUbiquityIdentityTokenDidChangeNotification`



# Handling Account Changes



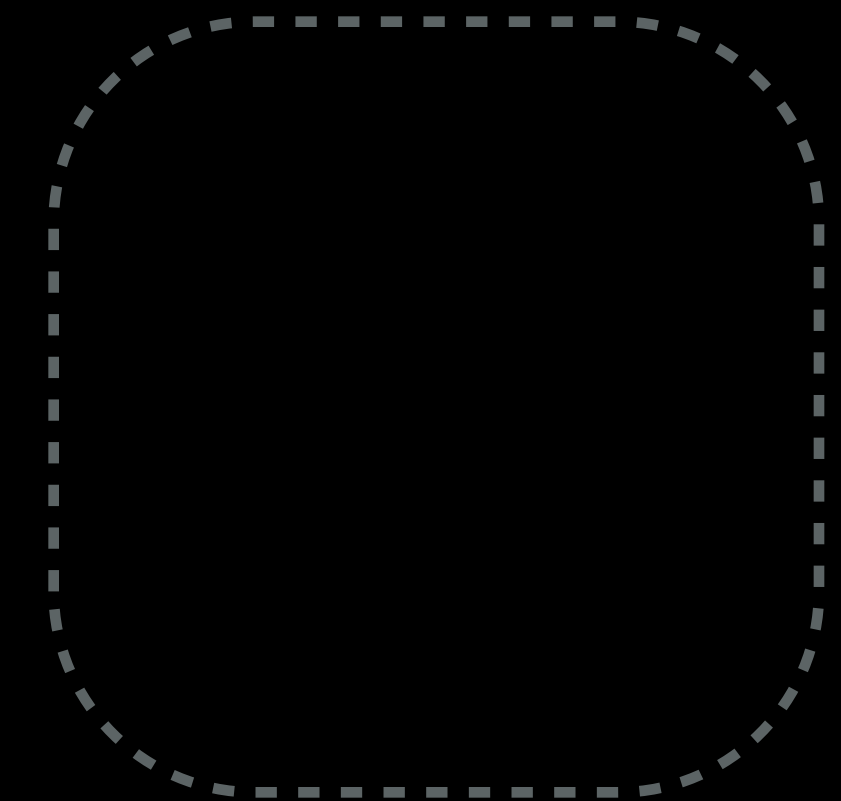
`NSUbiquityIdentityTokenDidChangeNotification`



# Handling Account Changes



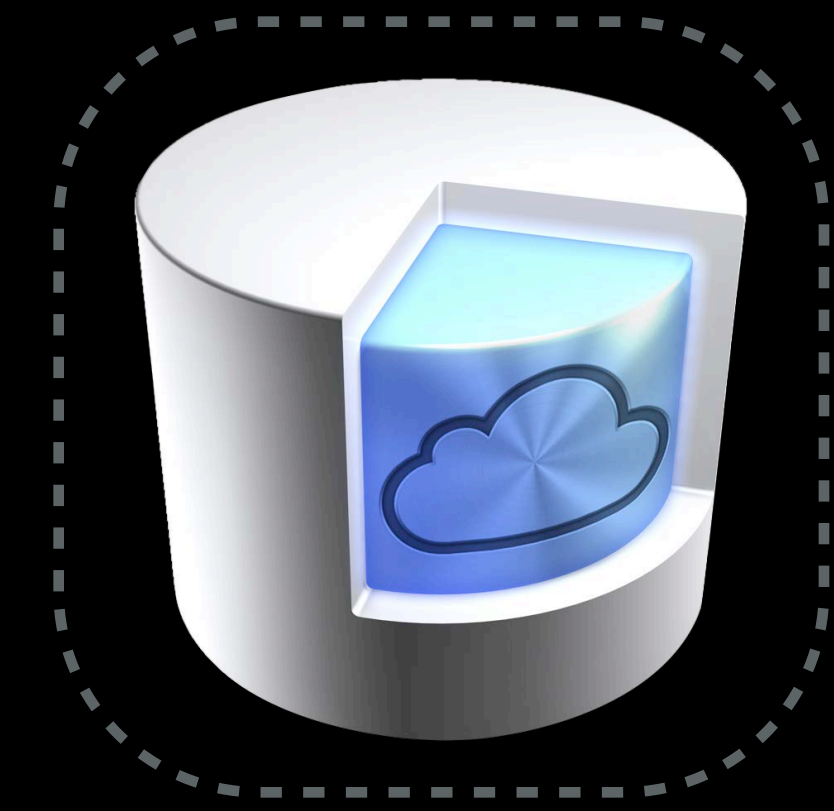
`NSUbiquityIdentityTokenDidChangeNotification`



# Handling Account Changes



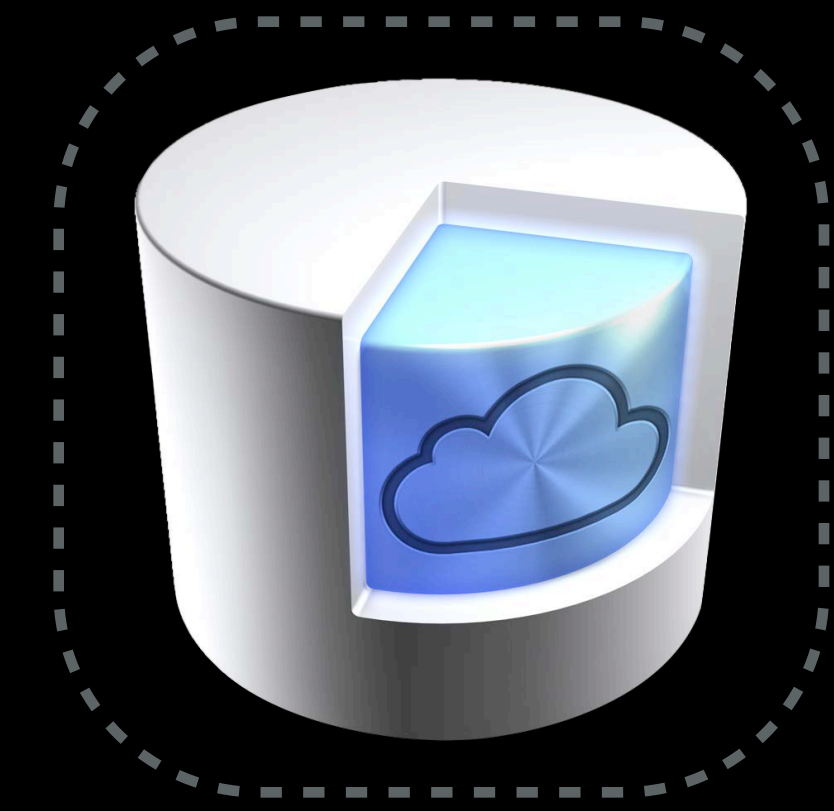
notme@icloud.com



# Handling Account Changes



`NSUbiquityIdentityTokenDidChangeNotification`





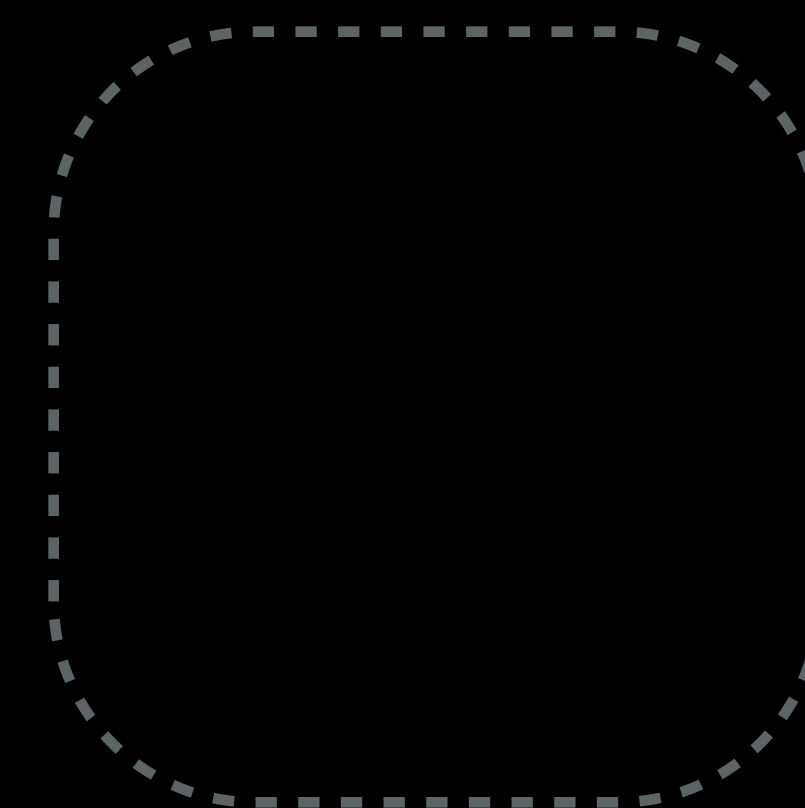
# Handling Account Changes



`NSUbiquityIdentityTokenDidChangeNotification`

`-[NSManagedObjectContext reset:]`

`-[NSPersistentStoreCoordinator removePersistentStore:]`

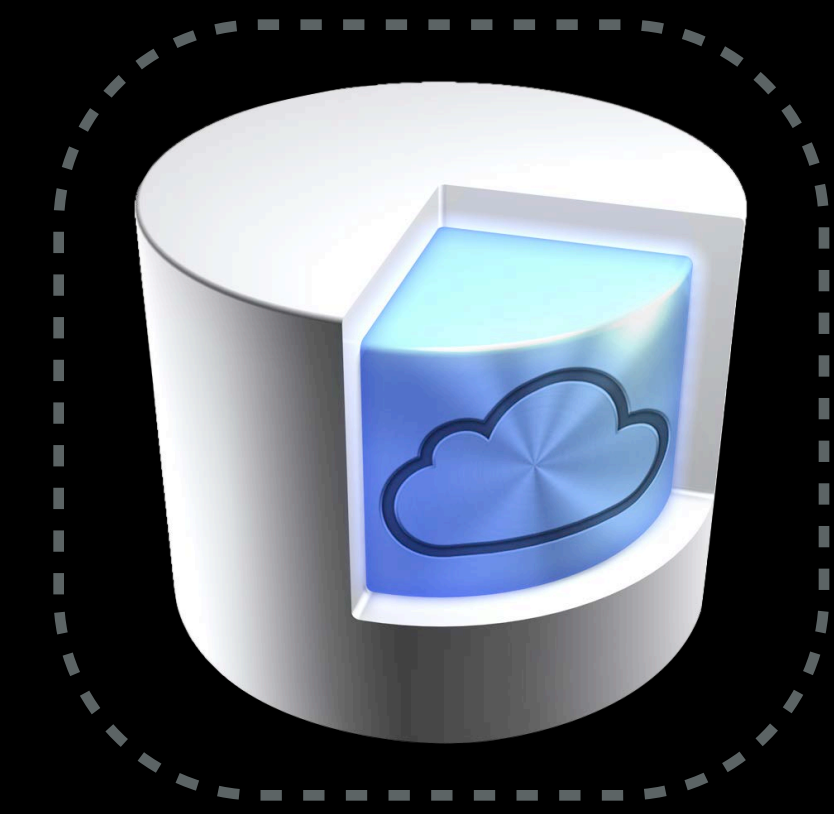


# Handling Account Changes



`NSUbiquityIdentityTokenDidChangeNotification`

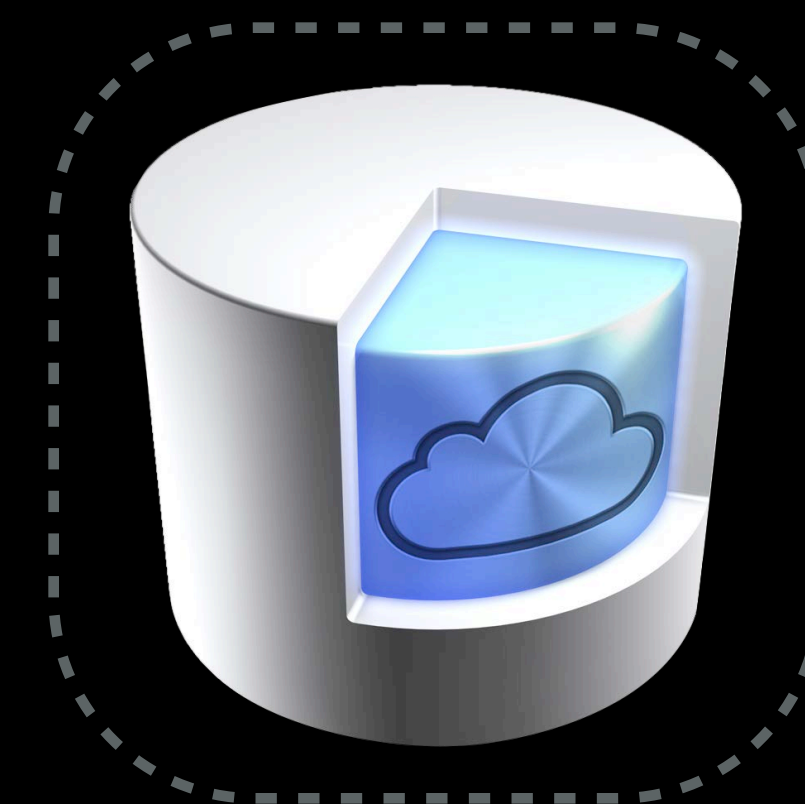
- `-[NSManagedObjectContext reset:]`
- `-[NSPersistentStoreCoordinator removePersistentStore:]`
- `-[NSPersistentStoreCoordinator addPersistentStore:]`



# Handling Account Changes



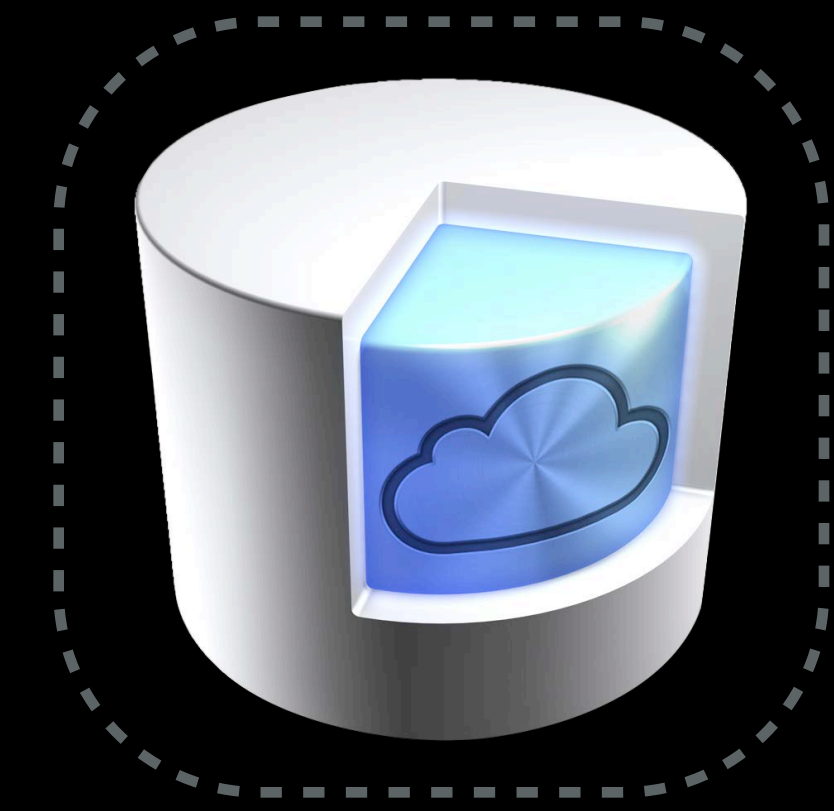
# Account Changes Now



# Account Changes Now



NSPersistentStoreCoordinatorStoresWillChangeNotification

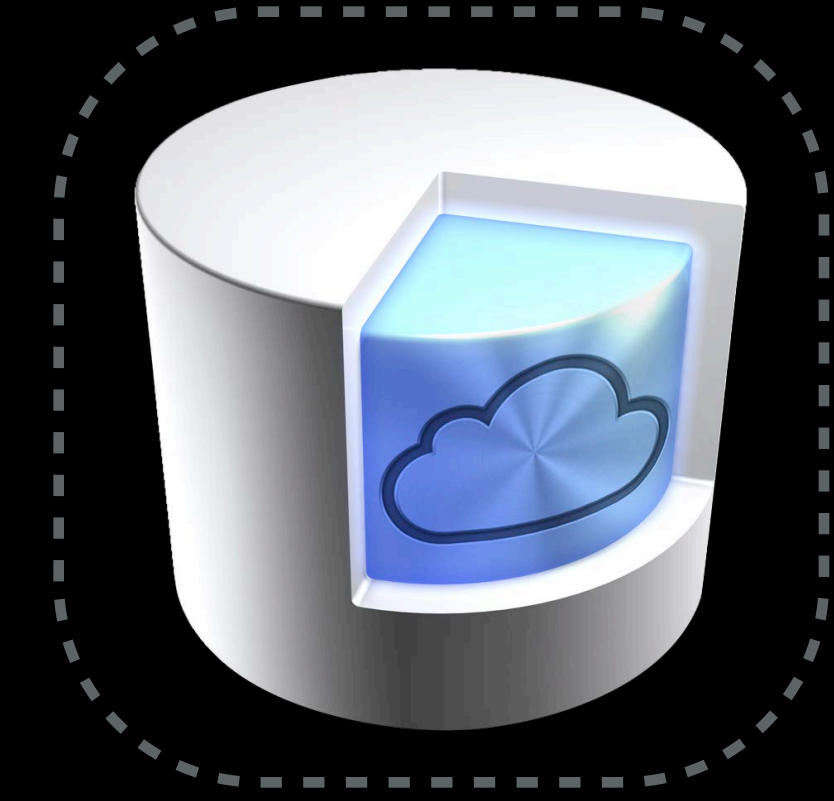


# Account Changes Now



`NSPersistentStoreCoordinatorStoresWillChangeNotification`

- `-[NSManagedObjectContext save:]`
- `-[NSManagedObjectContext reset:]`



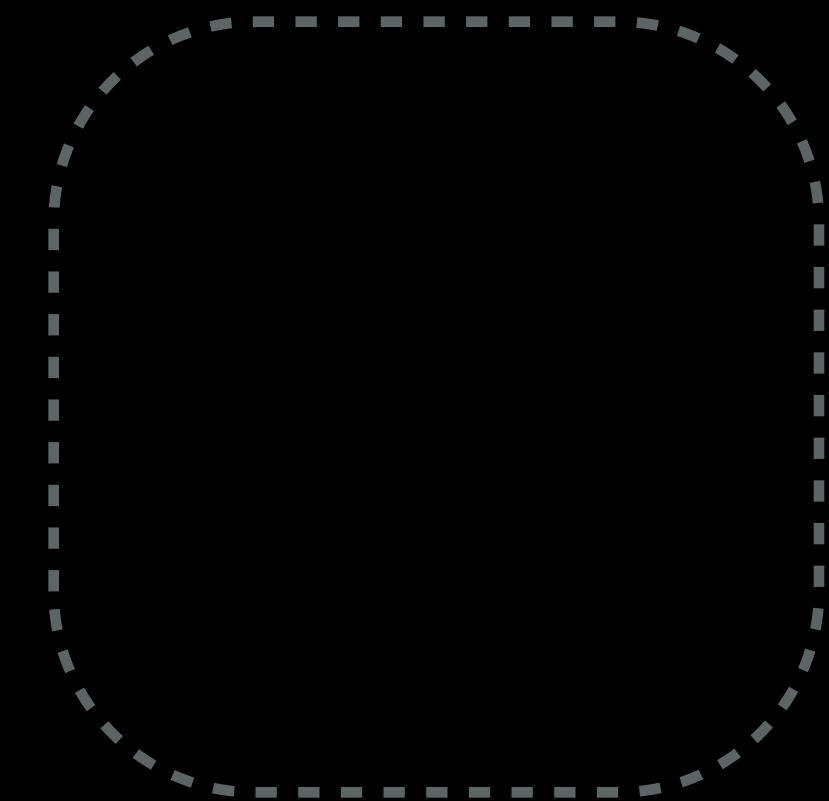
# Account Changes Now



`NSPersistentStoreCoordinatorStoresWillChangeNotification`

- `-[NSManagedObjectContext save:]`
- `-[NSManagedObjectContext reset:]`

Store Removed by Core Data



# Account Changes Now



`NSPersistentStoreCoordinatorStoresWillChangeNotification`

- `-[NSManagedObjectContext save:]`
- `-[NSManagedObjectContext reset:]`

Store Removed by Core Data

`NSPersistentStoreCoordinatorStoresDidChangeNotification`





# Account Changes Now



`NSPersistentStoreCoordinatorStoresWillChangeNotification`

- `-[NSManagedObjectContext save:]`
- `-[NSManagedObjectContext reset:]`

Store Removed by Core Data

`NSPersistentStoreCoordinatorStoresDidChangeNotification`

- `-[NSManagedObjectContext save:]`



# Account Changes Now



## NSPersistentStoreCoordinatorStoresWillChangeNotification

- [NSManagedObjectContext save:]
- [NSManagedObjectContext reset:]

Store Removed by Core Data

## NSPersistentStoreCoordinatorStoresDidChangeNotification

- [NSManagedObjectContext save:]



# Account Changes Now



NSPersistentStoreCoordinatorStoresWillChangeNotification

- [NSManagedObjectContext save:]
- [NSManagedObjectContext reset:]

Store Removed by Core Data

NSPersistentStoreCoordinatorStoresDidChangeNotification

- [NSManagedObjectContext save:]



# Account Changes Now



`NSPersistentStoreCoordinatorStoresWillChangeNotification`

- `-[NSManagedObjectContext save:]`
- `-[NSManagedObjectContext reset:]`

Store Removed by Core Data

`NSPersistentStoreCoordinatorStoresDidChangeNotification`

- `-[NSManagedObjectContext save:]`



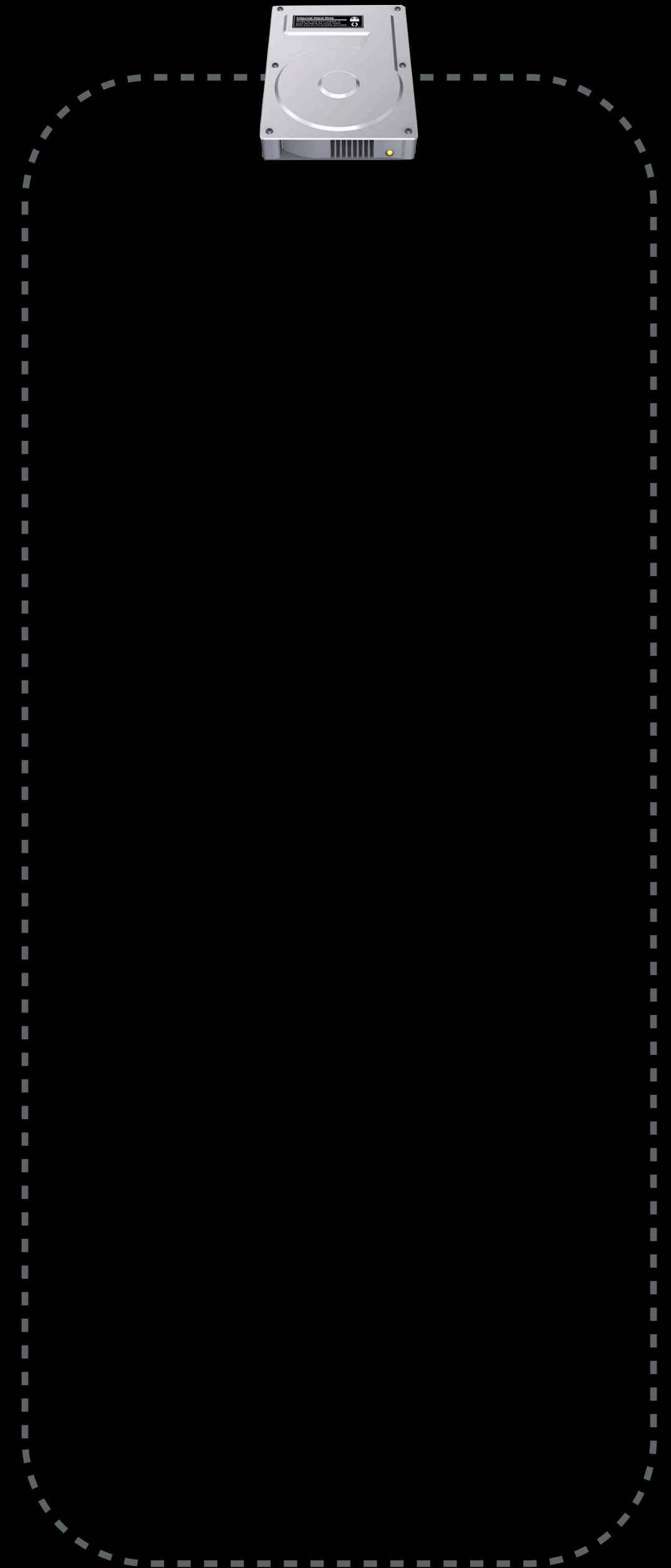
# Accounts and Stores

# Accounts and Stores

- Provide Store URL inside App Sandbox

# Accounts and Stores

- Provide Store URL inside App Sandbox
- Core Data creates an opaque container



# Accounts and Stores

- Provide Store URL inside App Sandbox
- Core Data creates an opaque container

me@icloud.com

notme@icloud.com

Local





# Accounts and Stores

# Accounts and Stores

file:///sandbox/store.sqlite

# Accounts and Stores

file://sandbox/store.sqlite

file://sandbox/CoreDataUbiquitySupport/[Core Data Directories]/store.sqlite

# Accounts and Stores

- Provide Store URL inside App Sandbox
- Core Data creates an opaque container

me@icloud.com

notme@icloud.com

Local



# Accounts and Stores

- Provide Store URL inside App Sandbox
- Core Data creates an opaque container
- Store URL tied to a single iCloud account

me@icloud.com

notme@icloud.com

Local



# Accounts and Stores

- Provide Store URL inside App Sandbox
- Core Data creates an opaque container
- Store URL tied to a single iCloud account
- Automatically removed by Core Data



**New API**

# iCloud Options



# iCloud Options

```
NSFileManager *fm = [NSFileManager defaultManager];
NSURL *url = [fm URLForUbiquitousContainerIdentifier:nil];

NSDictionary *options = @{
    NSPersistentStoreUbiquitousContentNameKey: @"Store",
    NSPersistentStoreUbiquitousContentURLKey: url
};
```

# iCloud Options

```
NSFileManager *fm = [NSFileManager defaultManager];  
NSURL *url = [fm URLForUbiquitousContainerIdentifier:nil];
```

```
NSDictionary *options = @{  
    NSPersistentStoreUbiquitousContentNameKey: @"Store",  
    NSPersistentStoreUbiquitousContentURLKey: url  
};
```

```
NSDictionary *options = @{  
    NSPersistentStoreUbiquitousContentNameKey: @"Store"  
};
```

# iCloud Options

```
NSFileManager *fm = [NSFileManager defaultManager];  
NSURL *url = [fm URLForUbiquitousContainerIdentifier:nil];
```

```
NSDictionary *options = @{  
    NSPersistentStoreUbiquitousContentNameKey: @"Store",  
    NSPersistentStoreUbiquitousContentURLKey: url  
};
```

```
NSDictionary *options = @{  
    NSPersistentStoreUbiquitousContentNameKey: @"Store"  
};
```

```
NSDictionary *options = @{  
    NSPersistentStoreUbiquitousContentNameKey: @"Store",  
    NSPersistentStoreUbiquitousContentURLKey: @"Subdir"  
};
```

# iCloud Options

# iCloud Options

- `NSPersistentStoreUbiquitousContainerIdentifierKey`
  - Passed directly to `-[NSFileManager URLForUbiquitousContainerIdentifier:]`
  - For apps that have multiple iCloud entitlements, nil will elect the first one by default

# iCloud Options

- `NSPersistentStoreUbiquitousContainerIdentifierKey`
  - Passed directly to `-[NSFileManager URLForUbiquitousContainerIdentifier:]`
  - For apps that have multiple iCloud entitlements, nil will elect the first one by default

```
NSDictionary *options = @{
    NSPersistentStoreUbiquitousContentNameKey: @"Store",
    NSPersistentStoreUbiquitousContentURLKey: @"Subdir",
    NSPersistentStoreUbiquitousContainerIdentifierKey: @"com.myapp.bundleID"
};
```

# Managing iCloud Content

# Managing iCloud Content

- `NSPersistentStoreRebuildFromUbiquitousContentOption`
  - Deletes the store file and rebuilds it from the iCloud content
  - `-addPersistentStore` will return an empty store



# Managing iCloud Content

- `NSPersistentStoreRebuildFromUbiquitousContentOption`
  - Deletes the store file and rebuilds it from the iCloud content
  - `-addPersistentStore` will return an empty store

```
NSDictionary *options = @{
    NSPersistentStoreUbiquitousContentNameKey: @"Store",
    NSPersistentStoreUbiquitousContentURLKey: @"Subdir",
    NSPersistentStoreRebuildFromUbiquitousContentOption: @YES
};
```

# Managing iCloud Content

# Managing iCloud Content

- `NSPersistentStoreRemoveUbiquitousMetadataOption`
  - Removes all iCloud associated metadata

# Managing iCloud Content

- `NSPersistentStoreRemoveUbiquitousMetadataOption`
  - Removes all iCloud associated metadata

```
NSDictionary *options = @{  
    NSPersistentStoreRemoveUbiquitousMetadataOption: @YES  
};
```

```
[psc migratePersistentStore:iCloudStore  
    toURL:url  
    options:options  
    withType:NSSQLiteStoreType  
    error:&error];
```

# Managing iCloud Content

# Managing iCloud Content

```
+ (BOOL)removeUbiquitousContentAndPersistentStoreAtURL:(NSURL *)storeURL  
                                options:(NSDictionary *)options  
                                error:(NSError**)error;
```

- Deletes ubiquitous content and persistent store file

# Managing iCloud Content

```
+ (BOOL)removeUbiquitousContentAndPersistentStoreAtURL:(NSURL *)storeURL
                                options:(NSDictionary *)options
                                error:(NSError**)error;
```

- Deletes ubiquitous content and persistent store file

```
NSDictionary *options = @{
    NSPersistentStoreUbiquitousContentNameKey: @"Store",
    NSPersistentStoreUbiquitousContentURLKey: @"Subdir"
};
```

```
BOOL success = [NSPersistentStoreCoordinator
    removeUbiquitousContentAndPersistentStoreAtURL:storeURL
                                options:options
                                error:&error];
```

# Managing iCloud Content



# Managing iCloud Content

```
+ (BOOL)removeUbiquitousContentAndPersistentStoreAtURL:(NSURL *)storeURL  
options:(NSDictionary *)options  
error:(NSError**)error;
```



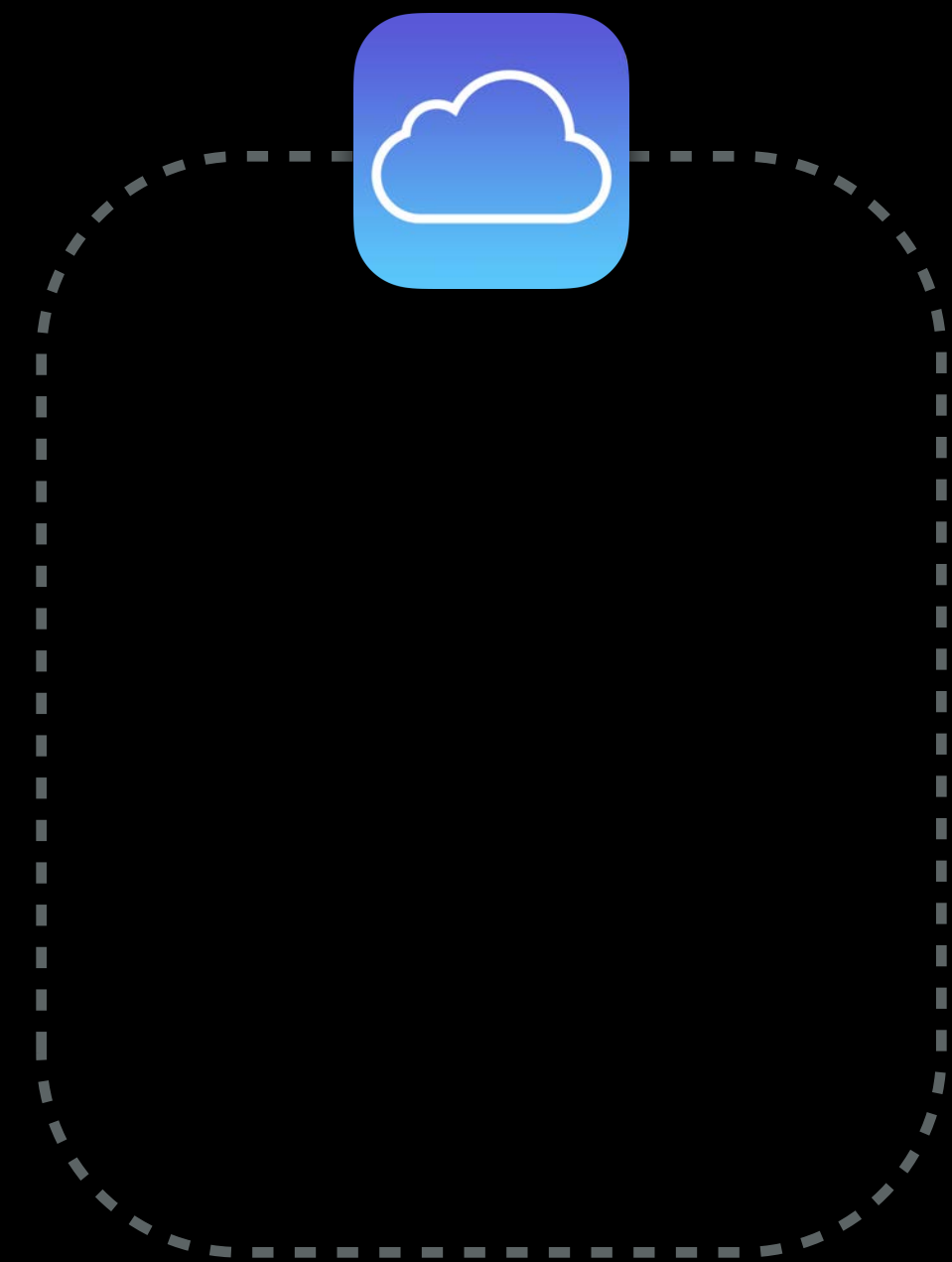
# Managing iCloud Content

```
+ (BOOL)removeUbiquitousContentAndPersistentStoreAtURL:(NSURL *)storeURL  
options:(NSDictionary *)options  
error:(NSError**)error;
```



# Managing iCloud Content

```
+ (BOOL)removeUbiquitousContentAndPersistentStoreAtURL:(NSURL *)storeURL  
options:(NSDictionary *)options  
error:(NSError**)error;
```



# NSPersistentStoreCoordinator Notifications

- NSPersistentStoreCoordinatorStoresWillChangeNotification
  - Only posted for iCloud Stores

# NSPersistentStoreCoordinator Notifications

- NSPersistentStoreCoordinatorStoresWillChangeNotification
  - Only posted for iCloud Stores

```
NSDictionary *userInfo = @{
    NSAddedPersistentStoresKey: @[store],
    NSRemovedPersistentStoresKey: @[store]
};
```

# NSPersistentStoreCoordinator Notifications

# NSPersistentStoreCoordinator Notifications

```
NSNotificationCenter *dc = [NSNotificationCenter defaultCenter];  
[dc addObserver:self  
    selector:@selector(storesWillChange:)  
    name:NSPersistentStoreCoordinatorStoresWillChangeNotification  
    object:pscOrNil];
```

# NSPersistentStoreCoordinator Notifications

```
NSNotificationCenter *dc = [NSNotificationCenter defaultCenter];  
[dc addObserver:self  
    selector:@selector(storesWillChange:)  
        name:NSPersistentStoreCoordinatorStoresWillChangeNotification  
        object:pscOrNil];  
[dc addObserver:self  
    selector:@selector(storesDidChange:)  
        name:NSPersistentStoreCoordinatorStoresDidChangeNotification  
        object:pscOrNil];
```



# NSPersistentStoreCoordinator Notifications

# NSPersistentStoreCoordinator Notifications

```
NSManagedObjectContext *moc = nil;
- (void)storesWillChange:(NSNotification *)n {
    if ([moc hasChanges]) {
        [moc save:&error];
    }

    [moc reset];
    //reset user interface
}
```

# NSPersistentStoreCoordinator Notifications

```
NSManagedObjectContext *moc = nil;
- (void)storesWillChange:(NSNotification *)n {
    if ([moc hasChanges]) {
        [moc save:&error];
    }

    [moc reset];
    //reset user interface
}

- (void)storesDidChange:(NSNotification *)n {
    //refresh user interface
}
```

# Living on iCloud

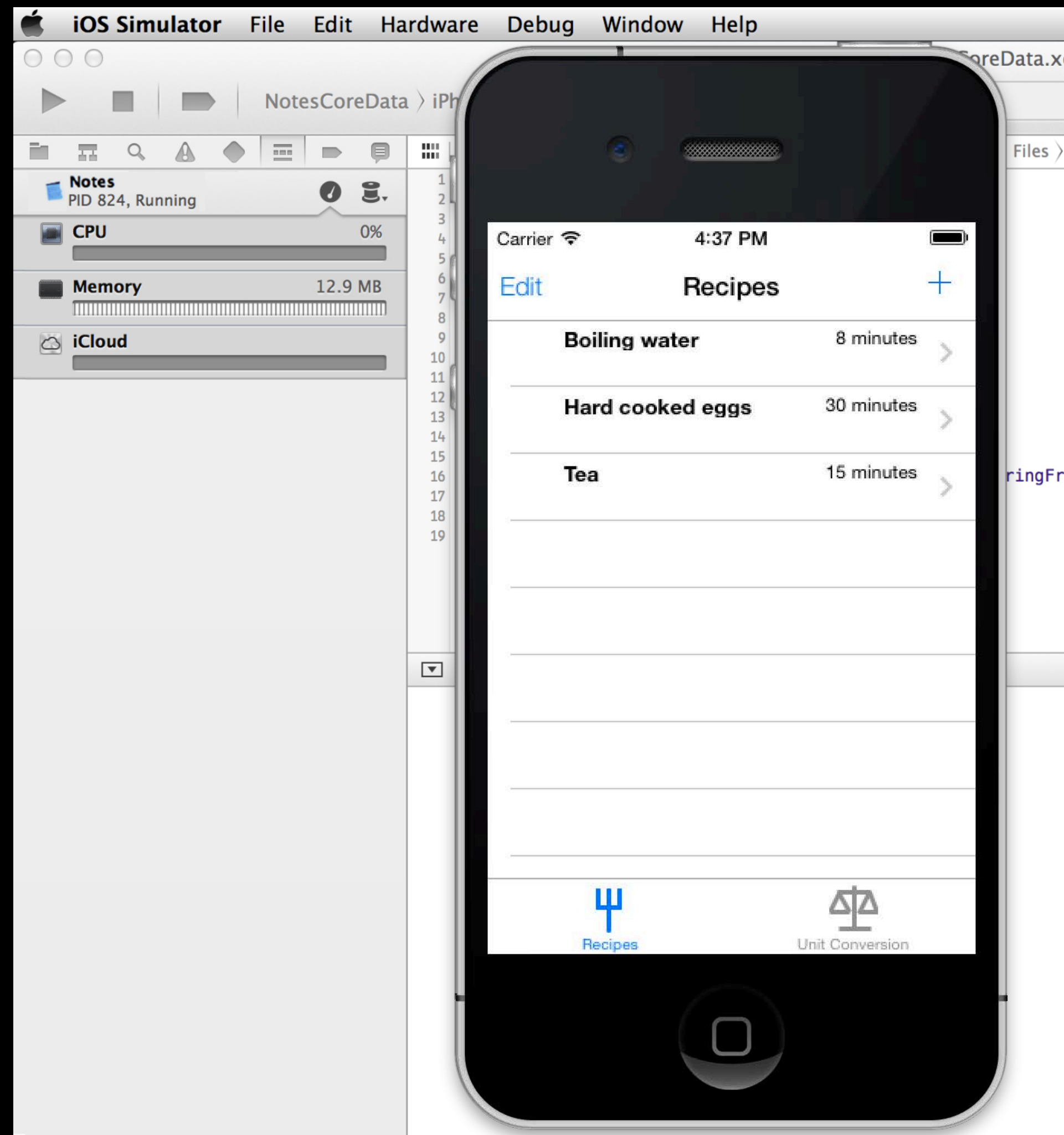
# Xcode Debugging Pane

The screenshot displays the Xcode Debugging Pane for a project named 'WWDCDemo.xcodeproj'. The pane is divided into several sections:

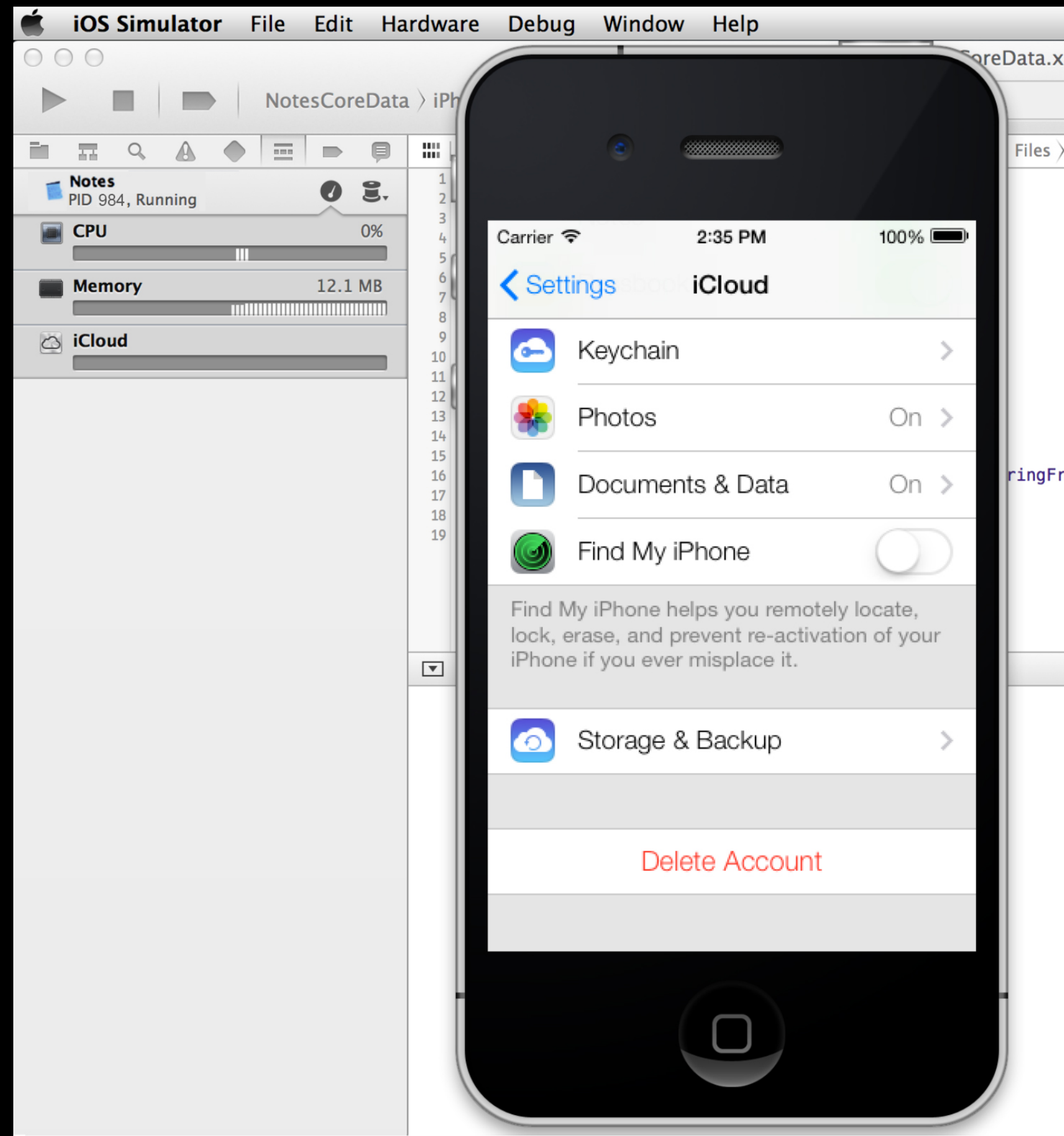
- Notes:** PID 3400, Running.
- CPU:** 0% usage.
- Memory:** Zero KB.
- Energy Use:** Zero.
- iCloud:** The main section, showing:
  - iCloud Usage:** com.apple.frameworks.wwdcdemo, 25 GB (2.7%), Status: Idle, Updated: 10 secs ago.
  - Storage:** 691.3 MB of 25 GB used.
  - Transfer Activity:** A bar chart showing upload and download activity over time. The y-axis is labeled '816 bytes'.
  - Documents:** A table listing documents and their status.

Name	Date Modified	Size	Kind	Status
▼ CoreData	Jun 7, 2013, 5:04 PM	Zero KB		Current
▼ CoreDataDemo	Jun 7, 2013, 5:04 PM	Zero KB		Current
▼ .baseline	Jun 7, 2013, 5:04 PM	Zero KB		Current
▼ CoreDataDemo	Jun 7, 2013, 5:04 PM	Zero KB		Current
▼ mDifS4UdwwYdvl7Go5truYxM2Ni5bmEWomzp7Bab...	Jun 7, 2013, 5:04 PM	Zero KB		Current
baseline.zip	Jun 7, 2013, 5:04 PM	10 KB	zip	Current, Stored in Cloud
▼ staging.nosync	Jun 7, 2013, 5:04 PM	Zero KB	nosync	Excluded
CoreDataDemo.sqlite	Jun 7, 2013, 5:04 PM	94 KB	sqlite	Excluded

# iOS Simulator Support



# iOS Simulator Support



# iOS Simulator Support



# iOS Simulator Support

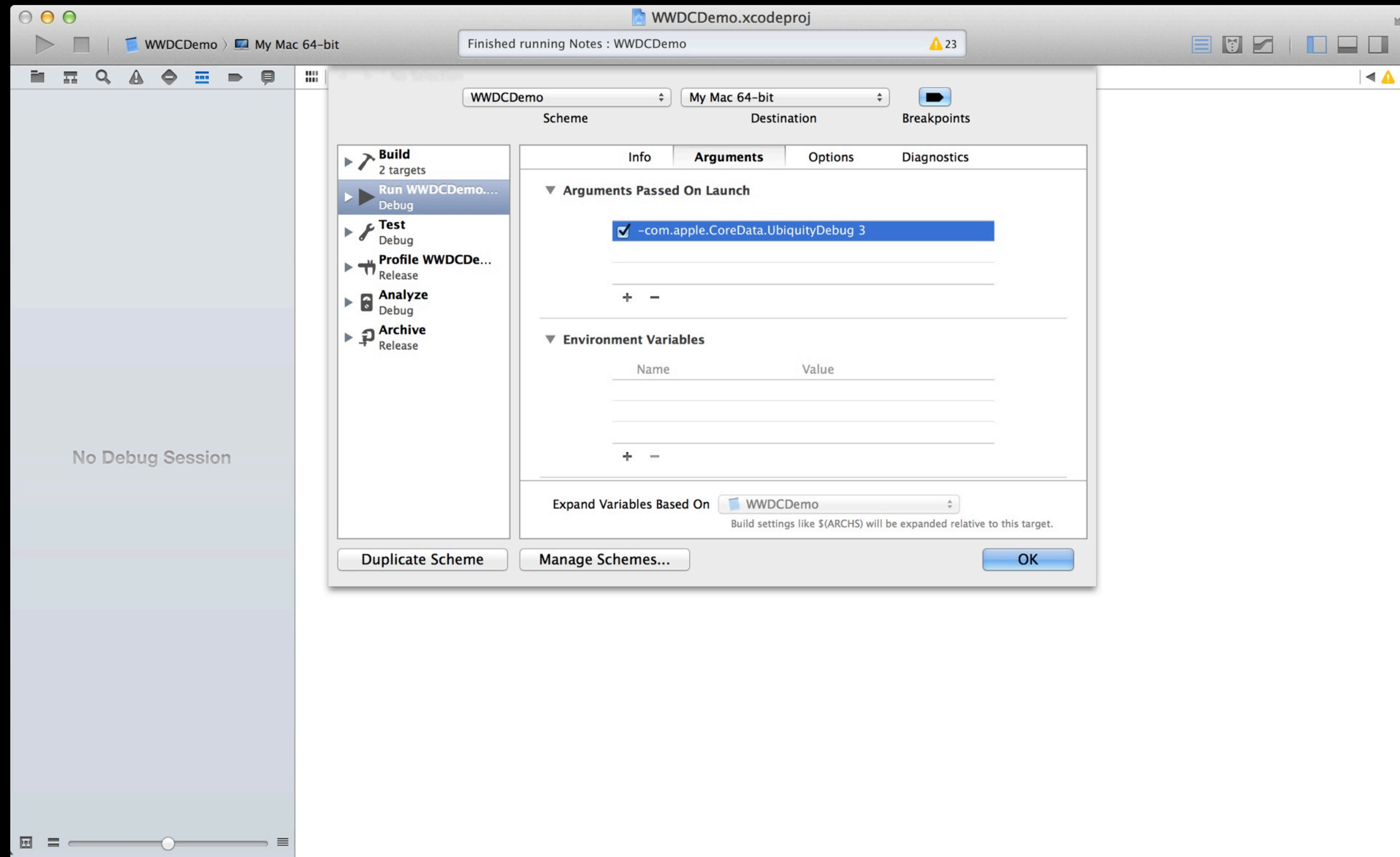
- iCloud Document Storage
- iCloud Key Value Store
- Push notifications

# iCloud Logging

# iCloud Logging

- Core Data Logging
  - NSUserDefaults
  - `-com.apple.coredata.ubiquity.logLevel 3`
  - `-com.apple.CoreData.SQLDebug 1`

# Core Data Logging



# iCloud Logging

# iCloud Logging

- OS X

# iCloud Logging

- OS X
  - ubcontrol tool

# iCloud Logging

- OS X
  - ubcontrol tool
  - \$ ubcontrol -k 7



# iCloud Logging

- OS X
  - ubcontrol tool
  - \$ ubcontrol -k 7
- iOS

# iCloud Logging

- OS X
  - ubcontrol tool
  - `$ ubcontrol -k 7`
- iOS
  - iCloud Debug Provisioning profile

# iCloud Logging Profile

The screenshot shows a web browser window with the URL `https://developer.apple.com/downloads/index.action`. The page title is "Downloads for Apple Developers". A search bar on the left contains the text "iCloud". The main content area displays a table of download items. The selected item is "iCloud Storage Debug Logging Profile", released on Oct 26, 2012. Below the table, there is a detailed description of the profile and two download links: "iCloud Debug Logging .mobileconfig(8.12 KB)" and "iCloud Debug Logging .zip(4.36 KB)". An important note is highlighted in a dashed box, stating that pre-release software is Apple Confidential Information.

Apple Developer | Technologies | Resources | Programs | Support | Member Center | Search Developer

## Downloads for Apple Developers

Hi, Developer | [My Profile](#) | [Sign out](#)

Search: iCloud

1 - 3 of 3 | Page 1 of 1

Description	Release Date
<a href="#">Bug Reporter Logging Profiles (iOS)</a>	Jun 2, 2013
<b><a href="#">iCloud Storage Debug Logging Profile</a></b>	Oct 26, 2012

The iCloud Storage Debug Logging profile enables the generation of debug logs that are needed to diagnose any problems using iCloud storage. Install this profile on your iOS 6 or OS X 10.8 or later devices and reproduce any iCloud storage bugs. Connect your iOS device to iTunes to pull the logs off of your device and attach them to your bug report.

**Important:** Pre-release software, including information about pre-release software, is Apple Confidential Information and is subject to the terms of your iOS Developer Program License Agreement, Mac Developer Program License Agreement, and/or Registered Apple Developer Agreement, as applicable. Unauthorized distribution or disclosure of Apple Confidential Information is prohibited.

- [iCloud Debug Logging .mobileconfig\(8.12 KB\)](#)
- [iCloud Debug Logging .zip\(4.36 KB\)](#)

# Installing Profile On iOS

# Installing Profile On iOS

- Download and unzip the profile from [developer.apple.com/downloads](https://developer.apple.com/downloads)
- Email file to yourself
- On iOS
  - Tap attachment in email
  - Follow instructions to install
  - Reboot your iOS device

# Retrieving Logs From iOS

# Retrieving Logs From iOS

- Sync your iOS device directly with iTunes
- Gather logs from
  - `~/Library/Logs/CrashReporter/MobileDevice/device-name/DiagnosticLogs`

*Demo*

**Melissa Turner**

Core Data Software Engineer



# Core Data SQLite Store

**Ben Trumbull**

Core Data Engineering Manager

# SQLite Store Database Journaling

- Core Data switched default journaling in iOS 7 and 10.9
  - Original SQLite rollback journaling
  - SQLite Write Ahead Logging (WAL)
  - WAL default for applications rebuilt on newer OS

# What's the Difference?

- Rollback journaling
  - Copies original pages to -journal file
  - Updates main db file in place
  - Deletes -journal on commit

# What's the Difference?

- Write ahead logging
  - Leaves main db file untouched
  - Appends transactions to -wal file
  - Uses -shm file to track which pages are in which file
  - Performs a checkpoint operation to merge the -wal into main file

# Rollback Journaling Behavior



SQLite Store

main db file

# Rollback Journaling Behavior



SQLite Store

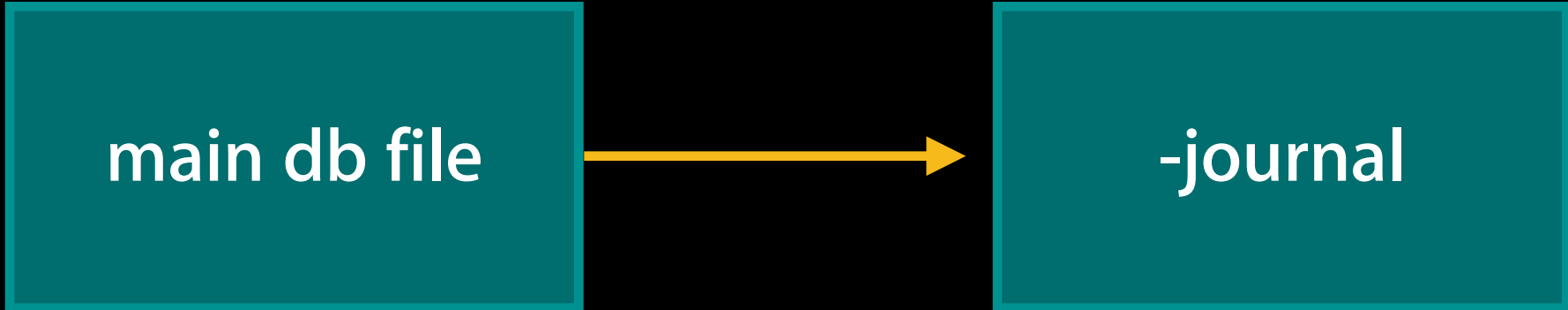
main db file

Begin Transaction

# Rollback Journaling Behavior



SQLite Store

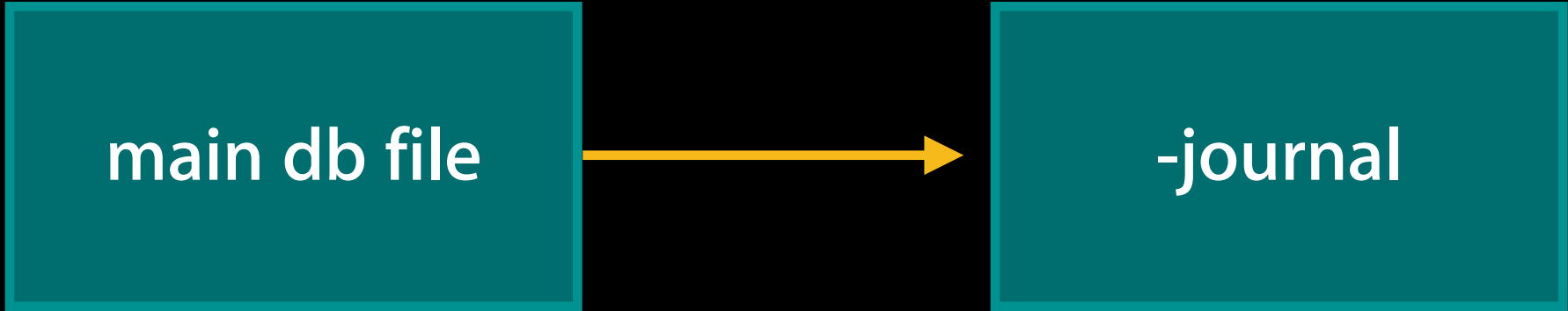


Begin Transaction

# Rollback Journaling Behavior



SQLite Store

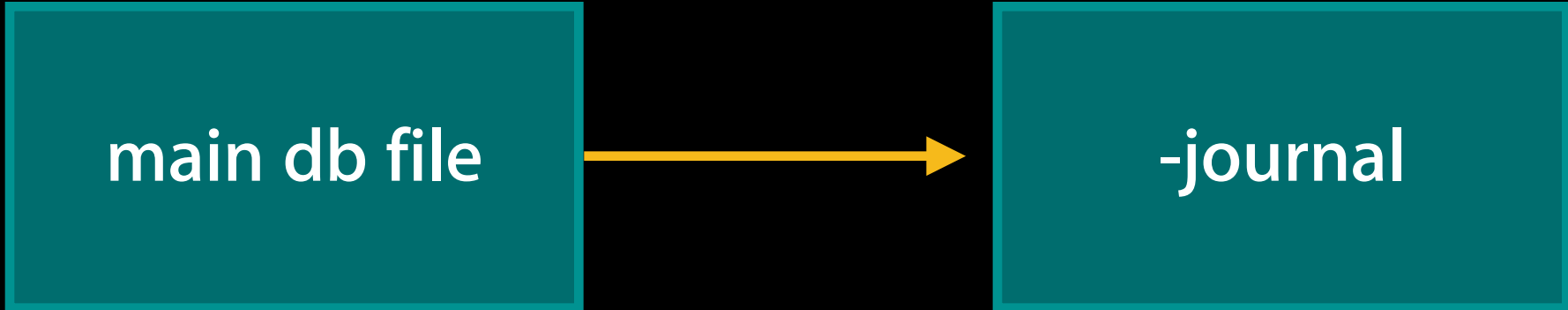




# Rollback Journaling Behavior



SQLite Store



Commit Transaction

# Rollback Journaling Behavior



SQLite Store

main db file

Commit Transaction

# Rollback Journaling Behavior



SQLite Store

main db file

# WAL Journaling Behavior



SQLite Store

main db file

-shm

-wal

# WAL Journaling Behavior



SQLite Store

main db file

-shm

-wal

Begin Transaction

# WAL Journaling Behavior



SQLite Store



main db file

-shm

-wal

Begin Transaction

# WAL Journaling Behavior



SQLite Store



main db file

-shm

-wal

# WAL Journaling Behavior



SQLite Store



main db file

-shm

-wal

Commit Transaction



# WAL Journaling Behavior



SQLite Store

main db file

-shm

-wal

Commit Transaction

# WAL Journaling Behavior



SQLite Store

main db file

-shm

-wal

# WAL Journaling Behavior



SQLite Store

main db file

-shm

-wal

Checkpoint Operation

# WAL Journaling Behavior



SQLite Store

main db file

-shm

-wal



Checkpoint Operation

# WAL Journaling Behavior



SQLite Store

main db file

-shm

-wal

# WAL Journaling Behavior



SQLite Store

main db file

-shm

# WAL Journaling Behavior



SQLite Store

main db file

-shm

-wal

# WAL Journaling Benefits

- Improved reliability across kernel panics and power loss
- Improved performance
  - Minimizes file system sync overhead, especially on iOS
- Improved concurrency
  - Multiple readers **and** one writer
- System wide adoption



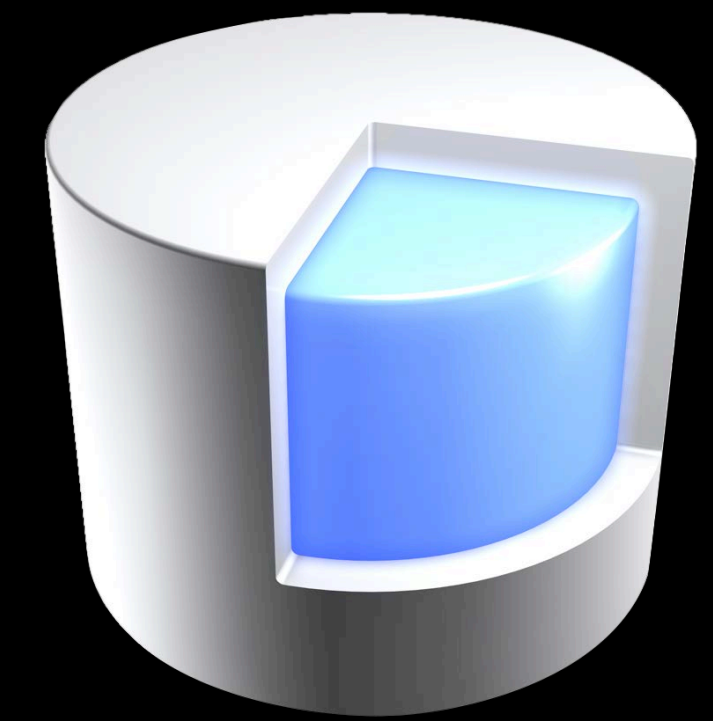
# Concurrency

- WAL databases support a writer concurrently with readers
- NSPersistentStoreCoordinator serializes all its requests
- Maximizing concurrency requires two Core Data stacks

# Leveraging WAL Concurrency



Main UI stack

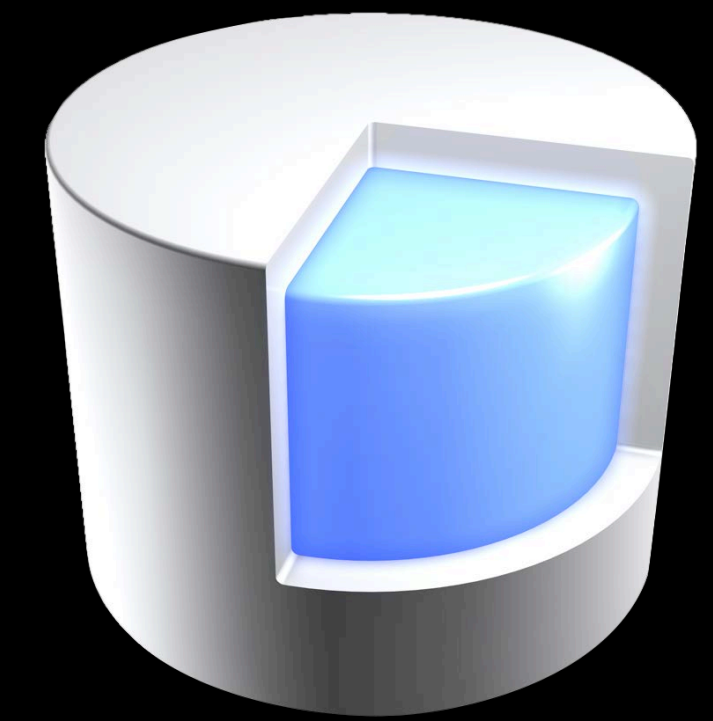


Background import  
stack

# Leveraging WAL Concurrency

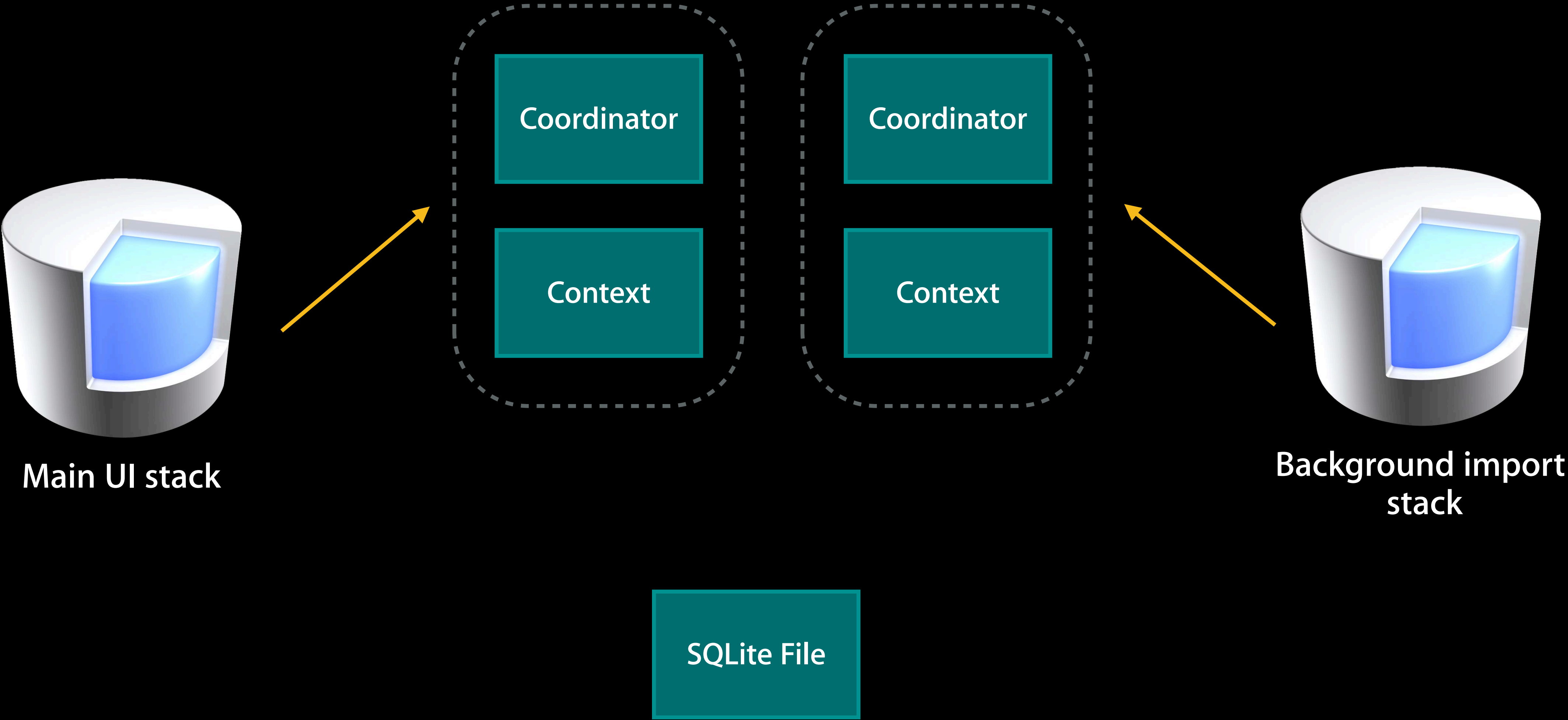


Main UI stack

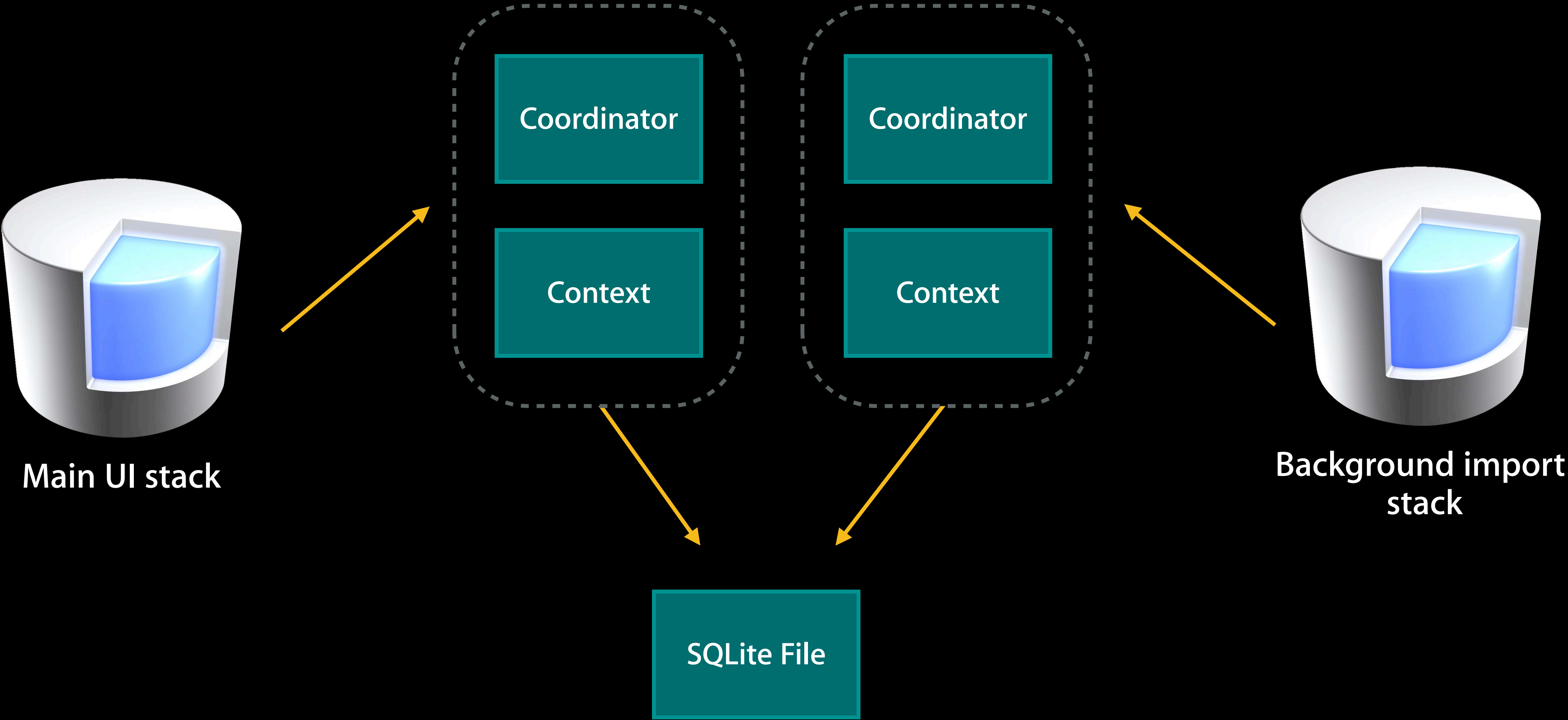


Background import  
stack

# Leveraging WAL Concurrency



# Leveraging WAL Concurrency



# Using Multiple NSPersistentStoreCoordinators

- Cannot pass objects between NSPersistentStoreCoordinators
  - Including objectIDs
  - Use -URIRepresentation instead
- Excellent for background updates
  - Importing data
  - Batch changes
- Allows UI to perform reads without blocking
  - Especially useful for responsiveness on launch

# Sharp Edges with WAL

- Not recommended for files you moved around
- Not recommended for read only files
- Not recommended as document format without a package wrapper
- First introduced in 10.7

# Setting Old Behavior

- WAL default for applications rebuilt for iOS 7 and 10.9
- Classic behavior is “pragma journal\_mode=DELETE”
- Set in options dictionary to `-addPersistentStoreWithType:`



# Setting Old Behavior

- WAL default for applications rebuilt for iOS 7 and 10.9
- Classic behavior is “pragma journal\_mode=DELETE”
- Set in options dictionary to -addPersistentStoreWithType:

```
NSDictionary* storeOptions = @{ NSSQLitePragmaOptions : @{ @"journal_mode" :  
@"DELETE" } };
```

# General SQLite Caveats

- Never use file system routines directly on open db files
  - Bypassing SQLite APIs is hazardous
- Network file systems
  - No cache coherency between machines
- Avoid changing `pragma synchronous` behavior
  - Apple customizes its behavior

# Other Enhancements

- Improvements with lock contention
- Power management changes
  - Integration with I/O throttling
- Guarded file descriptors

# More Information

**David DeLong**

Technology Evangelist  
[delong@apple.com](mailto:delong@apple.com)

**Cocoa Feedback**

[cocoa-feedback@apple.com](mailto:cocoa-feedback@apple.com)

**Core Data Documentation**

Programming Guides, Examples, Tutorials  
<http://developer.apple.com/>

**Apple Developer Forums**

<http://devforums.apple.com>

# Related Sessions

Core Data Performance Optimization and Debugging

Nob Hill  
Wednesday 2:00PM

Hidden Gems in Cocoa and Cocoa Touch

Nob Hill  
Friday 10:15AM

# Labs

Core Data Lab	Services Lab B Wednesday 3:15PM	
iCloud Lab	Tools Lab C Thursday 9:00AM	
Core Data Lab	Frameworks Lab A Thursday 2:00PM	
Core Data Lab	Services Lab A Friday 9:00AM	

 WWDC2013