Frameworks

#WWDC14

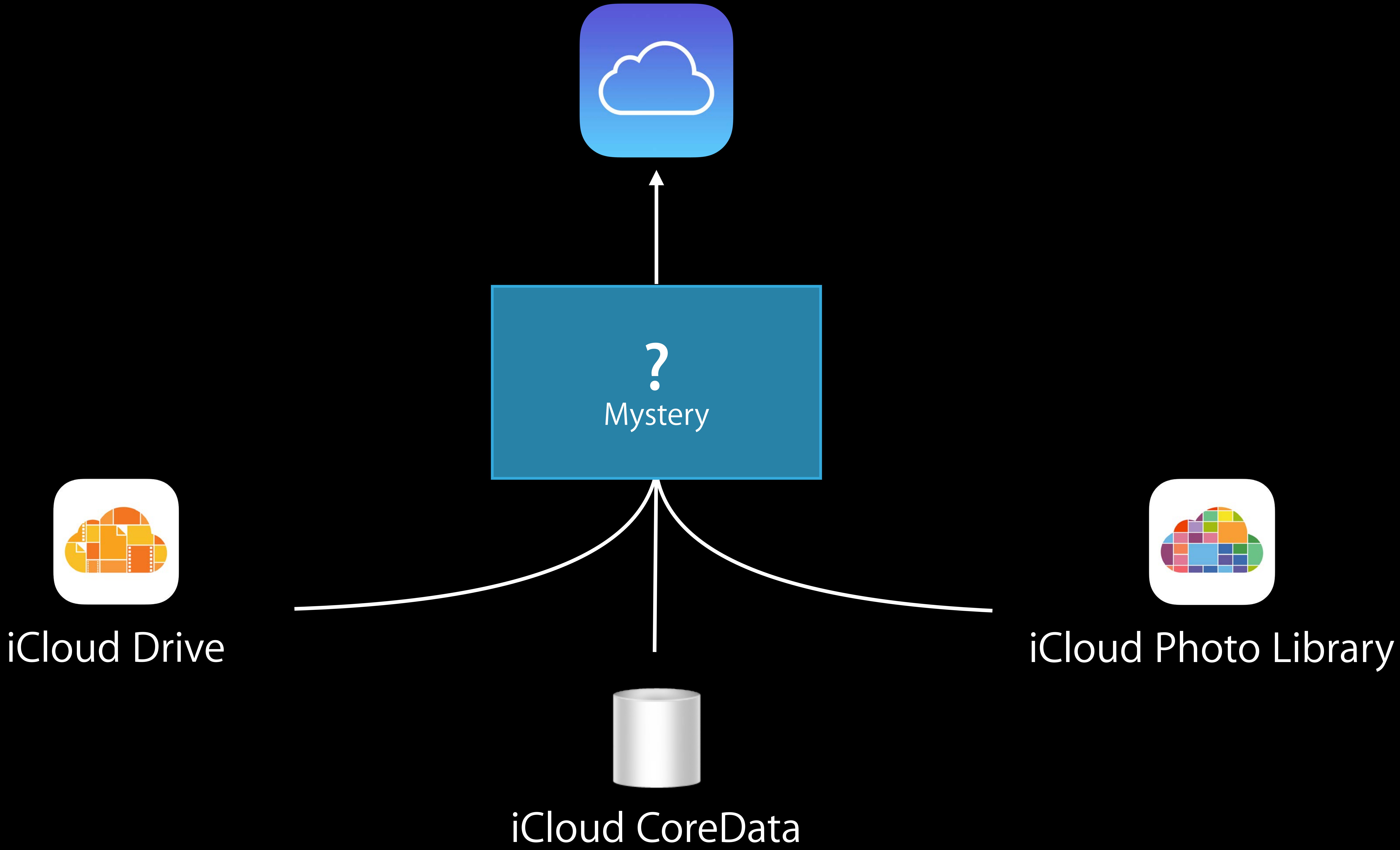
Introducing CloudKit

A how-to guide for iCloud for your Apps

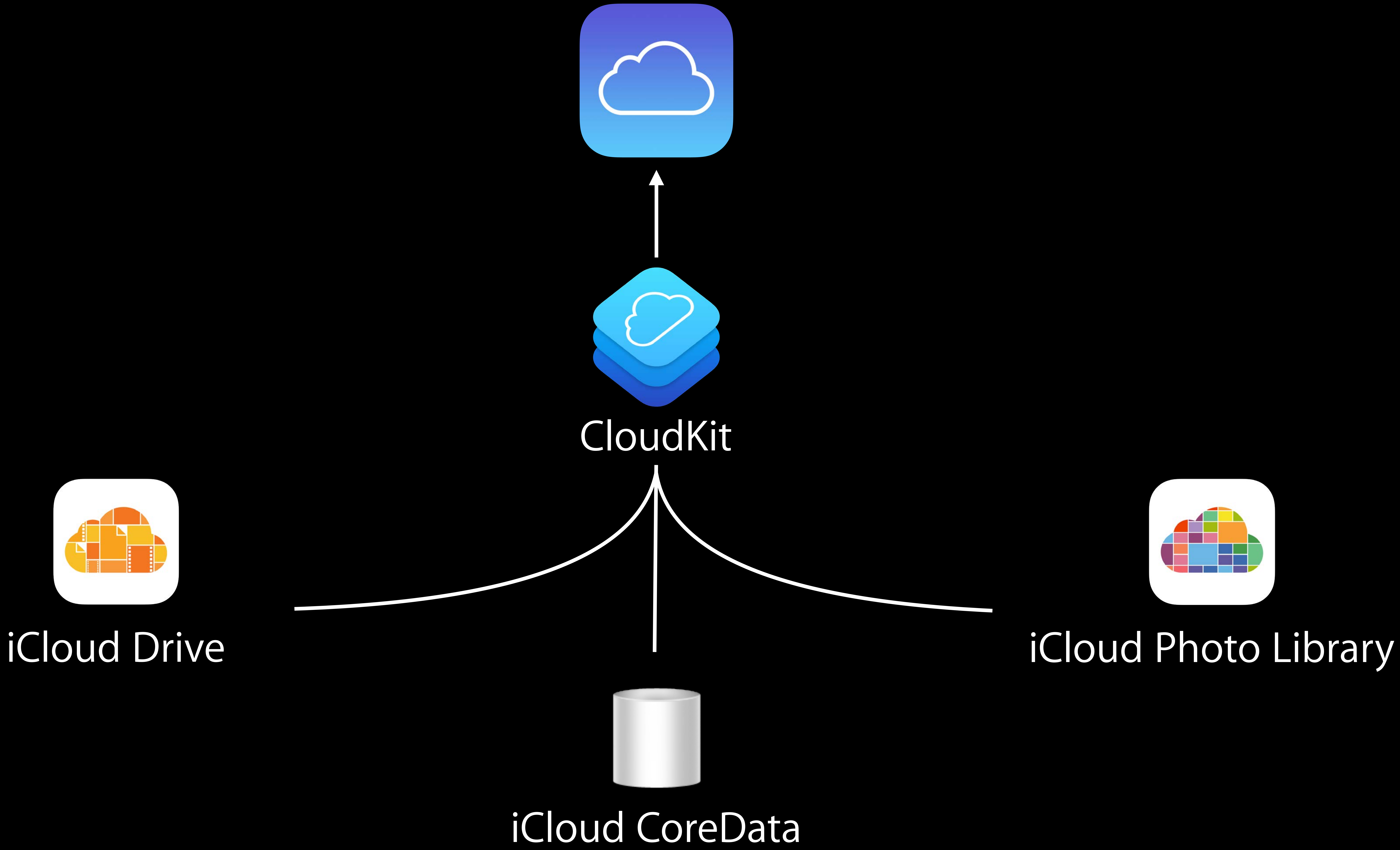
Session 208

Olivier Bonnet
CloudKit Client Software

Introduction



Introduction



Overview

Overview

What is CloudKit

Overview

What is CloudKit

Enabling CloudKit in your application

Overview

What is CloudKit

Enabling CloudKit in your application

Introduction to the API

Overview

What is CloudKit

Enabling CloudKit in your application

Introduction to the API

User Accounts

Overview

What is CloudKit

Enabling CloudKit in your application

Introduction to the API

User Accounts

When to use CloudKit

What is CloudKit?

NEW

What is CloudKit?

NEW

Access to iCloud servers

What is CloudKit?

NEW

Access to iCloud servers

Supported on OS X and iOS

What is CloudKit?

NEW

Access to iCloud servers

Supported on OS X and iOS

Uses iCloud accounts

What is CloudKit?

NEW

Access to iCloud servers

Supported on OS X and iOS

Uses iCloud accounts

Public and private databases

What is CloudKit?

NEW

Access to iCloud servers

Supported on OS X and iOS

Uses iCloud accounts

Public and private databases

Structured and bulk data

What is CloudKit?

NEW

Access to iCloud servers

Supported on OS X and iOS

Uses iCloud accounts

Public and private databases

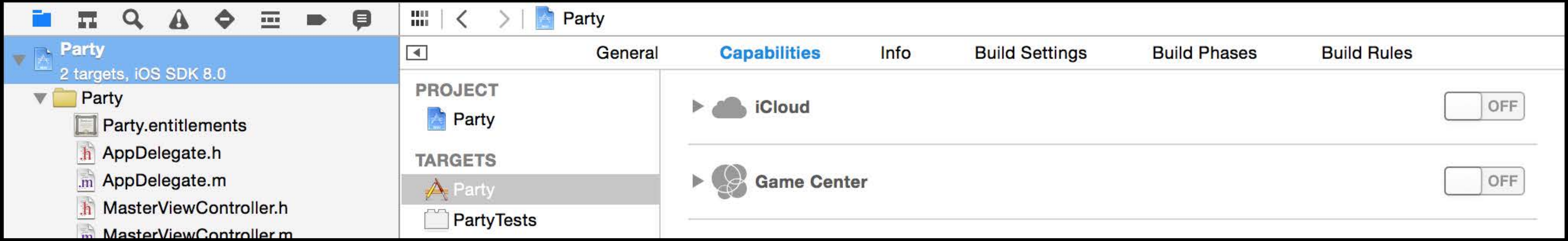
Structured and bulk data

Transport, not local persistence

Enabling CloudKit in Your Application

Enabling CloudKit in Your Application

Navigate to your application's Capabilities pane


Party
2 targets, iOS SDK 8.0
Party
Party.entitlements
AppDelegate.h
AppDelegate.m
MasterViewController.h

Party

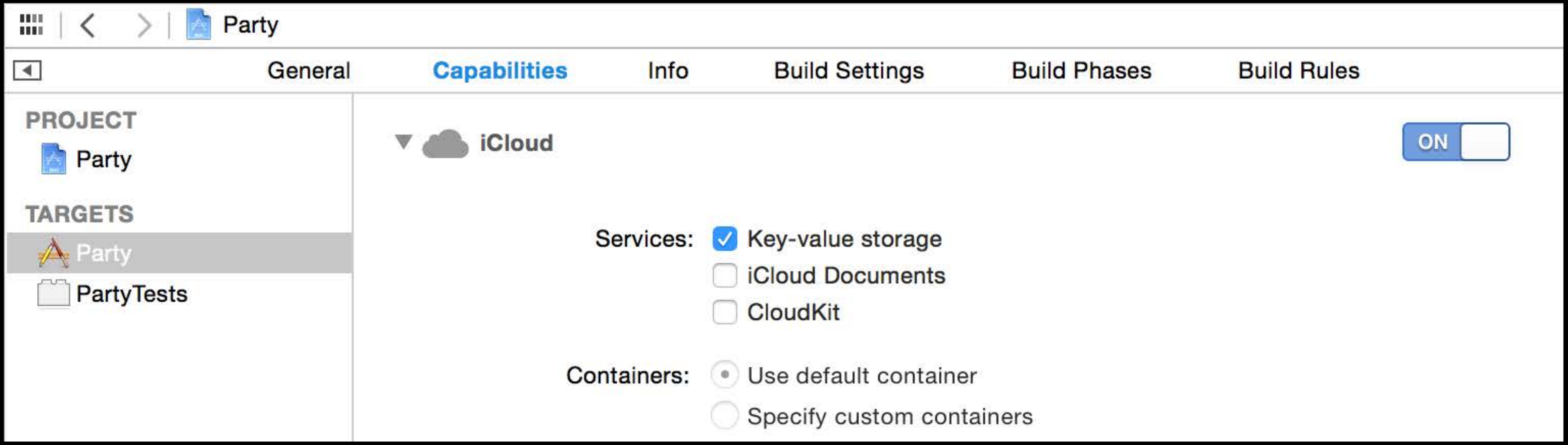
General | Capabilities | Info | Build Settings | Build Phases | Build Rules

PROJECT
Party
TARGETS
Party
PartyTests

iCloud OFF
Game Center OFF

Enabling CloudKit in Your Application

Enable iCloud

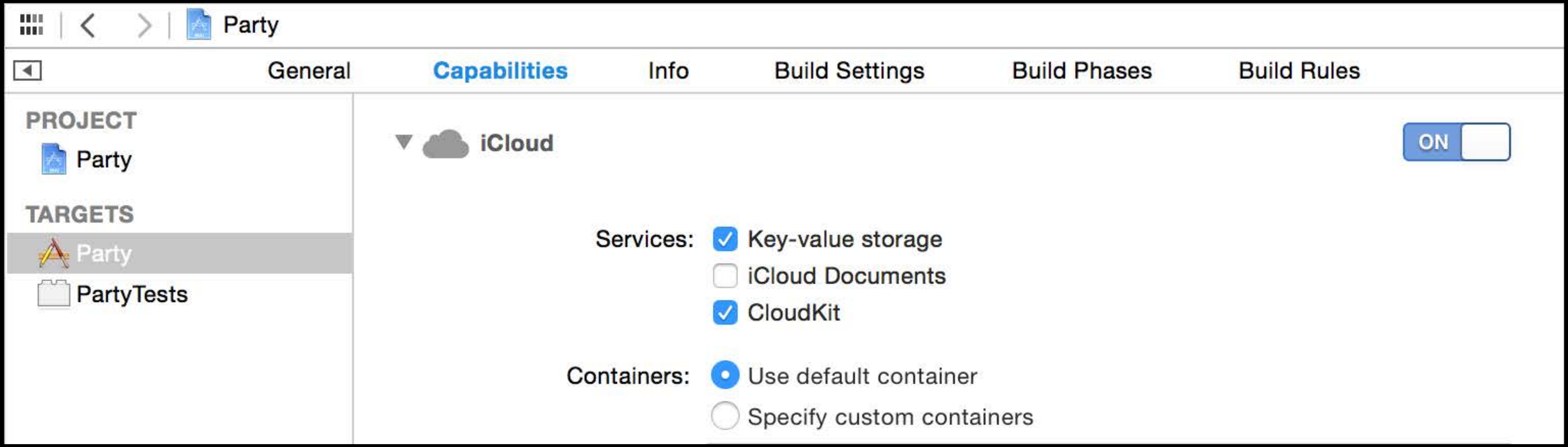


Services: [✓] Key-value storage
[ ] iCloud Documents
[ ] CloudKit

Containers: [✓] Use default container
[ ] Specify custom containers

Enabling CloudKit in Your Application

Enable CloudKit



Services: [✓ Key-value storage] iCloud Documents [✓ CloudKit]

Containers: [✓ Use default container] Specify custom containers

Introducing CloudKit API

Paul Seligman
CloudKit Client Software

Fundamental CloudKit Objects

Fundamental CloudKit Objects

Fundamental CloudKit Objects

Containers

Fundamental CloudKit Objects

Containers

Databases

Fundamental CloudKit Objects

Containers

Databases

Records

Fundamental CloudKit Objects

Containers

Databases

Records

Record Zones

Fundamental CloudKit Objects

Containers

Databases

Records

Record Zones

Record Identifiers

Fundamental CloudKit Objects

Containers

Databases

Records

Record Zones

Record Identifiers

References

Fundamental CloudKit Objects

Containers

Databases

Records

Record Zones

Record Identifiers

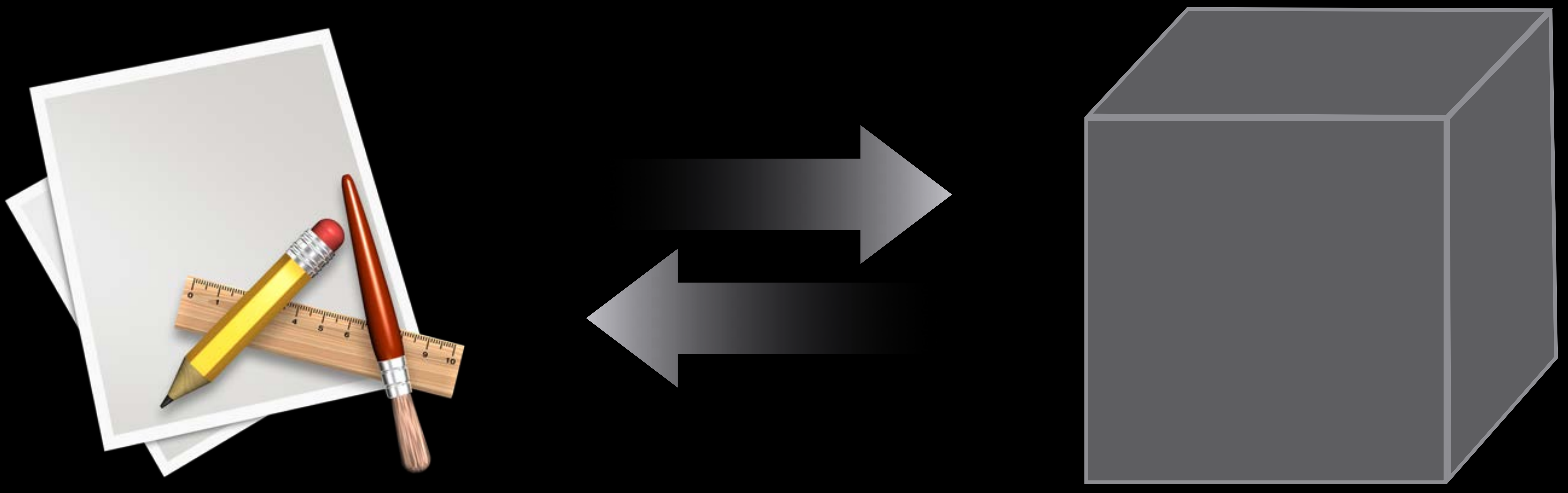
References

Assets

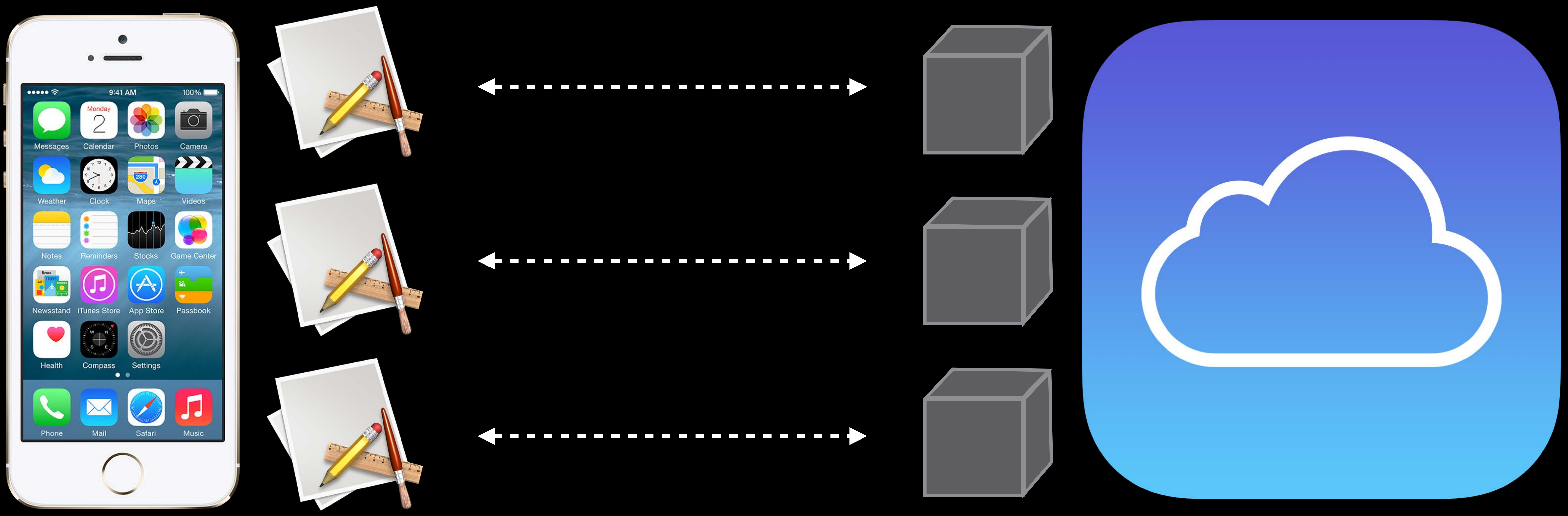
Fundamental CloudKit Objects

Containers
Databases
Records
Record Zones
Record Identifiers
References
Assets

Containers



Containers



Containers

Containers

CKContainer

Containers

CKContainer

One container per app

Containers

CKContainer

One container per app

Data segregation

Containers

CKContainer

One container per app

Data segregation

User encapsulation

Containers

CKContainer

One container per app

Data segregation

User encapsulation

Managed by the developer

Containers

CKContainer

One container per app

Data segregation

User encapsulation

Managed by the developer

- Managed via WWDR portal

Containers

CKContainer

One container per app

Data segregation

User encapsulation

Managed by the developer

- Managed via WWDR portal
- Unique across all developers

Containers

CKContainer

One container per app

Data segregation

User encapsulation

Managed by the developer

- Managed via WWDR portal
- Unique across all developers

Can be shared between apps

Fundamental CloudKit Objects

Containers

Databases

Records

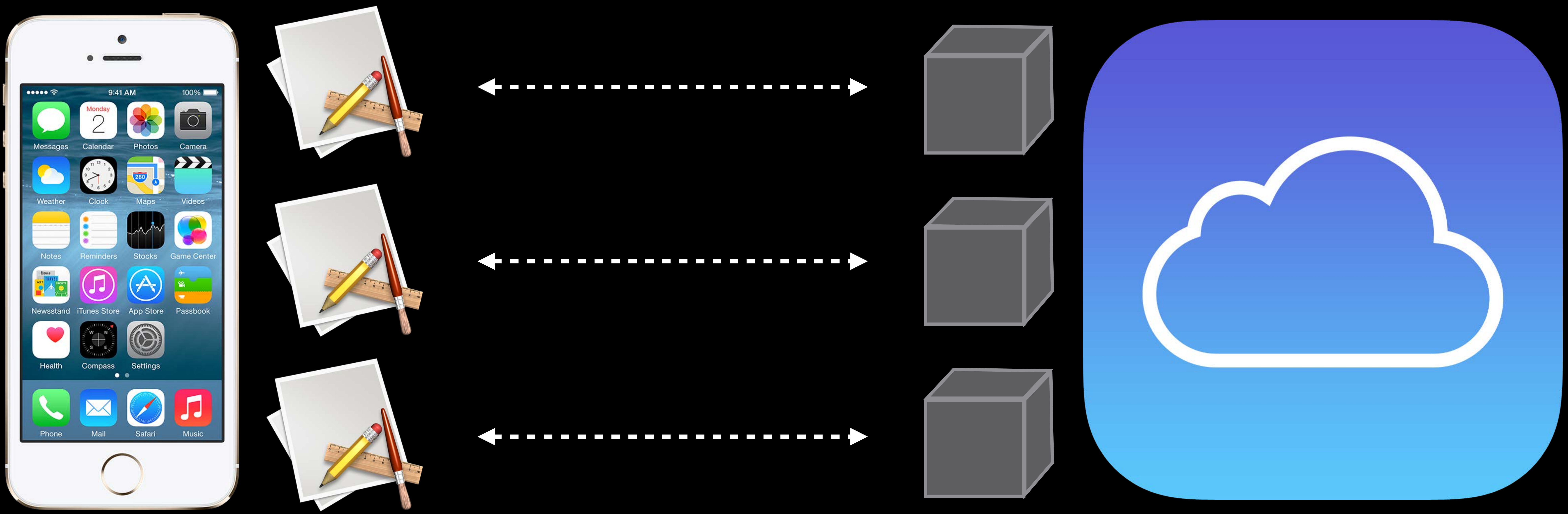
Record Zones

Record Identifiers

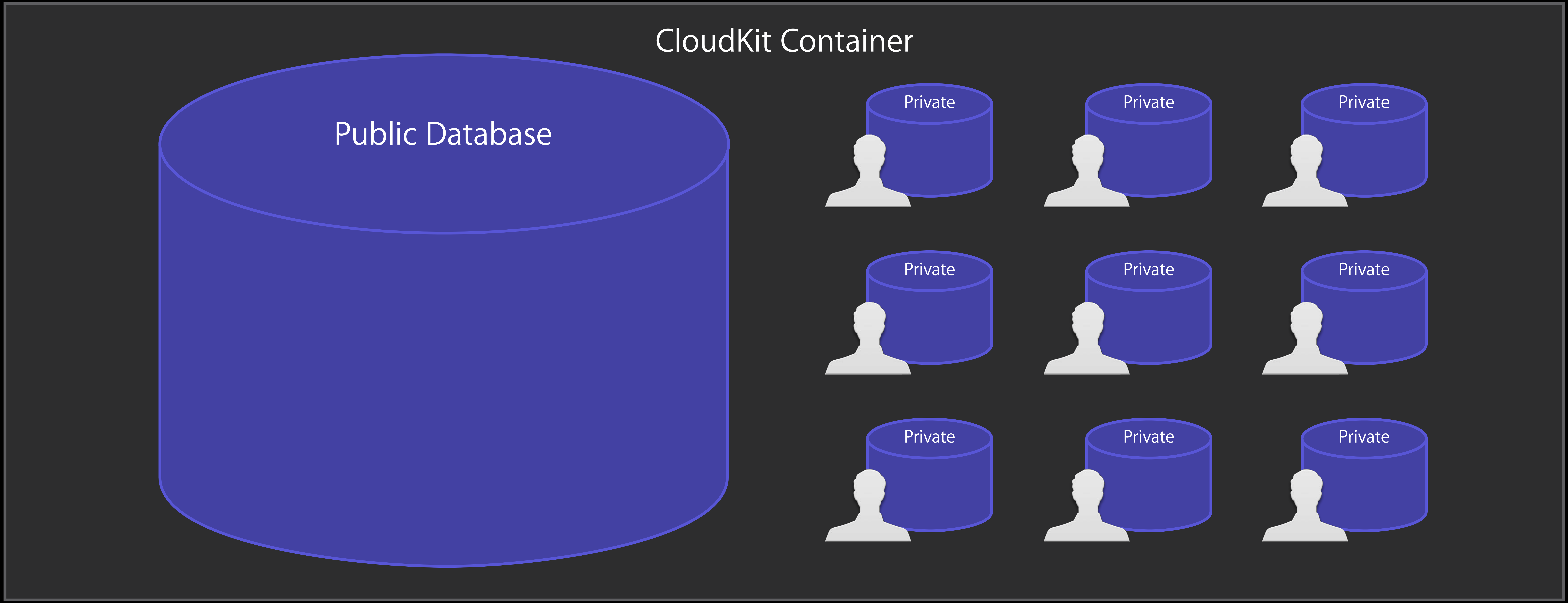
References

Assets

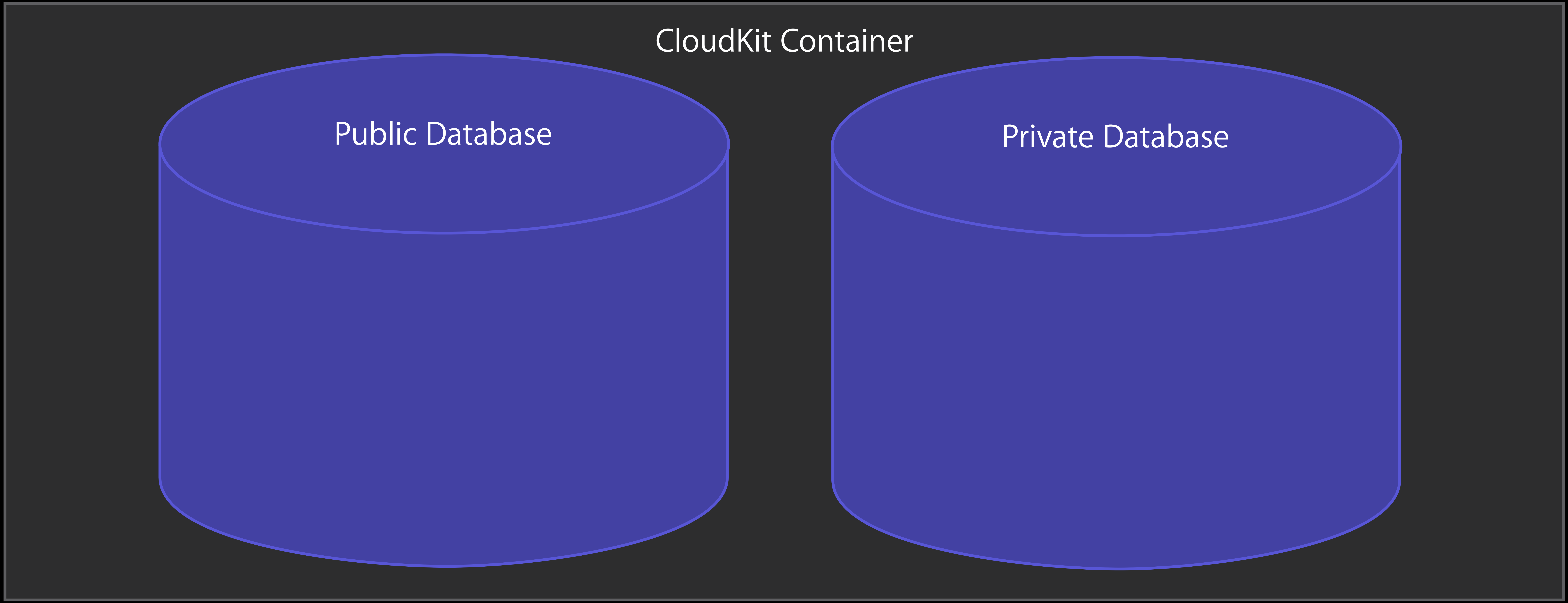
Databases



Databases


CloudKit Container
Public Database
Private
Private
Private
Private
Private
Private
Private
Private
Private

Databases



Databases

Databases

CKDatabase

Databases

CKDatabase

Every app has access to two databases

Databases

CKDatabase

Every app has access to two databases

- Public Database

Databases

CKDatabase

Every app has access to two databases

- Public Database
- Private Database

Databases

CKDatabase

Every app has access to two databases

- Public Database
- Private Database

```
 CKDatabase *publicDatabase = [[CKContainer defaultContainer] publicCloudDatabase];
CKDatabase *privateDatabase = [[CKContainer defaultContainer] privateCloudDatabase];
```

Databases

	Public Database	Private Database

Databases

	Public Database	Private Database
Data Type	Shared Data	Current User’s Data

Databases

	Public Database	Private Database
Data Type	Shared Data	Current User’s Data
Account	Required for Writing	Required

Databases

	Public Database	Private Database
Data Type	Shared Data	Current User’s Data
Account	Required for Writing	Required
Quota	Developer	User

Databases

	Public Database	Private Database
Data Type	Shared Data	Current User’s Data
Account	Required for Writing	Required
Quota	Developer	User
Default Permissions	World Readable	User Readable

Databases

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

Fundamental CloudKit Objects

Containers

Databases

Records

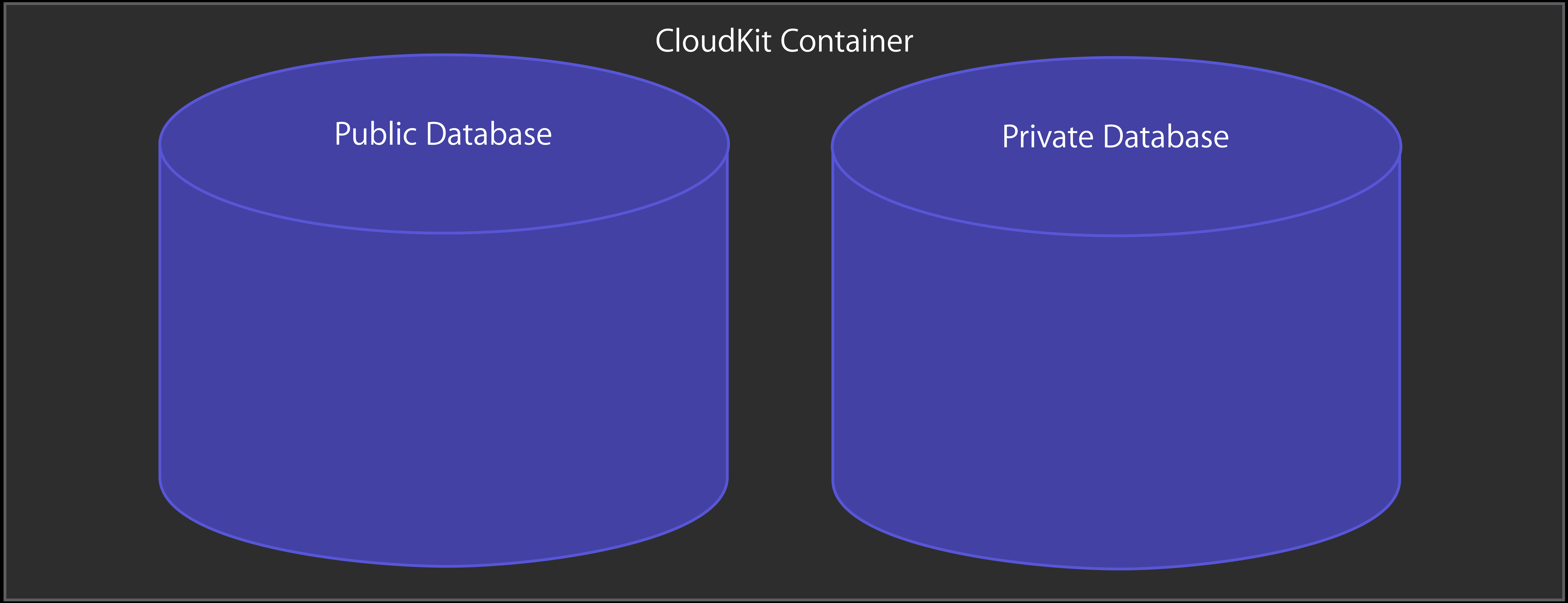
Record Zones

Record Identifiers

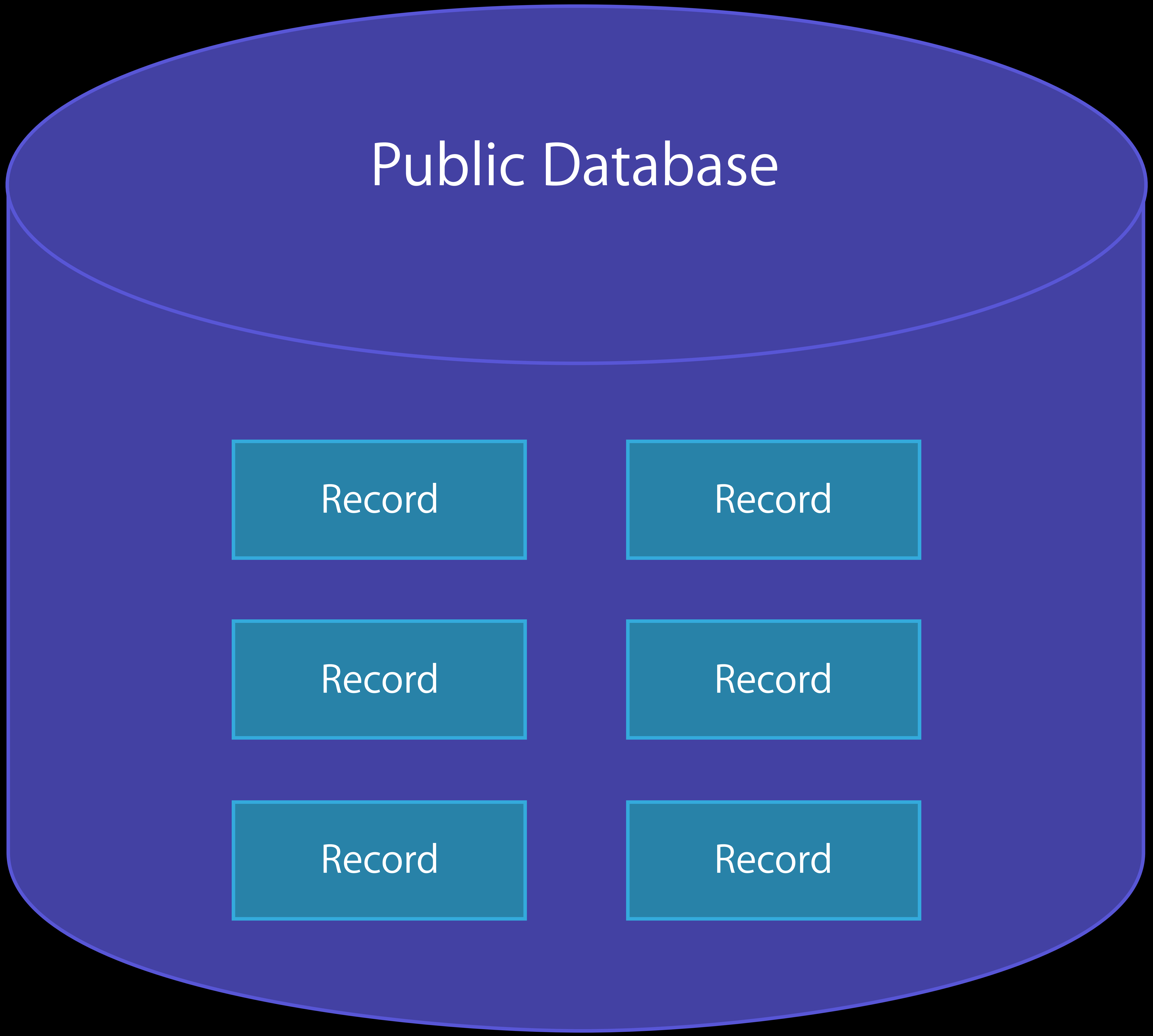
References

Assets

Records



Records


Public Database
Record
Record
Record
Record
Record
Record

Records

Records

CKRecord

Records

CKRecord

Structured Data

Records

CKRecord

Structured Data

Wraps key/value pairs

Records

CKRecord

Structured Data

Wraps key/value pairs

Record Type

Records

CKRecord

Structured Data

Wraps key/value pairs

Record Type

Just-in-time schema

Records

CKRecord

Structured Data

Wraps key/value pairs

Record Type

Just-in-time schema

Metadata

Records

Record Values

Records

Record Values

- NSString
- NSNumber
- NSData
- NSDate

Records

Record Values

- NSString
- NSNumber
- NSData
- NSDate
- CLLocation

Records

Record Values

- NSString
- NSNumber
- NSData
- NSDate
- CLLocation
- CKReference
- CKAsset

Records

Record Values

- NSString
- NSNumber
- NSData
- NSDate
- CLLocation
- CKReference
- CKAsset
- Arrays of the above

Records

Records

```
@interface CKRecord : NSObject <NSSecureCoding, NSCopying>
```

Records

```
@interface CKRecord : NSObject <NSSecureCoding, NSCopying>

- (instancetype)initWithRecordType:(NSString *)recordType;
```

Records

```
@interface CKRecord : NSObject <NSSecureCoding, NSCopying>

- (instancetype)initWithRecordType:(NSString *)recordType;

- (id)objectForKey:(NSString *)key;
- (void)setObject:(id <CKRecordValue>)object forKey:(NSString *)key;
```

Records

```
@interface CKRecord : NSObject <NSSecureCoding, NSCopying>

- (instancetype)initWithRecordType:(NSString *)recordType;

- (id)objectForKey:(NSString *)key;
- (void)setObject:(id <CKRecordValue>)object forKey:(NSString *)key;

- (id)objectForKeyedSubscript:(NSString *)key;
- (void)setObject:(id <CKRecordValue>)object
forKeyedSubscript:(NSString *)key;
```

Records

```
@interface CKRecord : NSObject <NSSecureCoding, NSCopying>

- (instancetype)initWithRecordType:(NSString *)recordType;

- (id)objectForKey:(NSString *)key;
- (void)setObject:(id <CKRecordValue>)object forKey:(NSString *)key;

- (id)objectForKeyedSubscript:(NSString *)key;
- (void)setObject:(id <CKRecordValue>)object
forKeyedSubscript:(NSString *)key;

- (NSArray /* NSString */ *)allKeys;
```

Records

Records

```
CKRecord *party = [[CKRecord alloc] initWithRecordType:@“Party”];
```

Records

```
CKRecord *party = [[CKRecord alloc] initWithRecordType:@“Party”];
```

```
// setting values
[party setObject:@"Post Presentation Beers"
          forKey:@"summary"];

NSDate *startDate = [NSDate dateWithTimeIntervalSinceNow:30.0 * 60.0];
party[@"start"] = startDate;
```

Records

```
CKRecord *party = [[CKRecord alloc] initWithRecordType:@“Party”];

// setting values
[party setObject:@"Post Presentation Beers"
          forKey:@"summary"];

NSDate *startDate = [NSDate dateWithTimeIntervalSinceNow:30.0 * 60.0];
party[@"start"] = startDate;
```

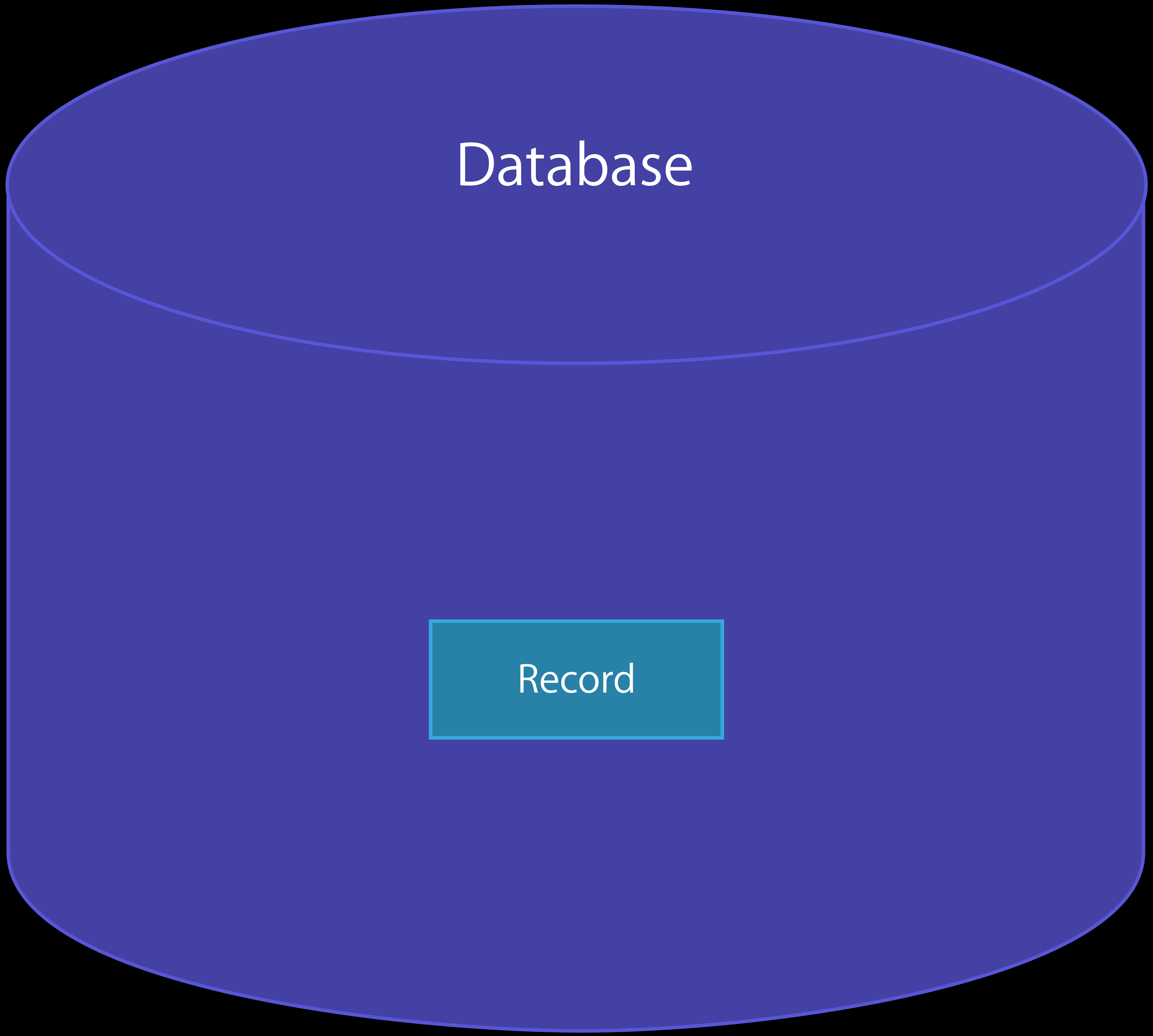
```
// accessing values
NSString *summary = [party objectForKey:@"summary"];

NSDate *startDate = party[@"start"];
```

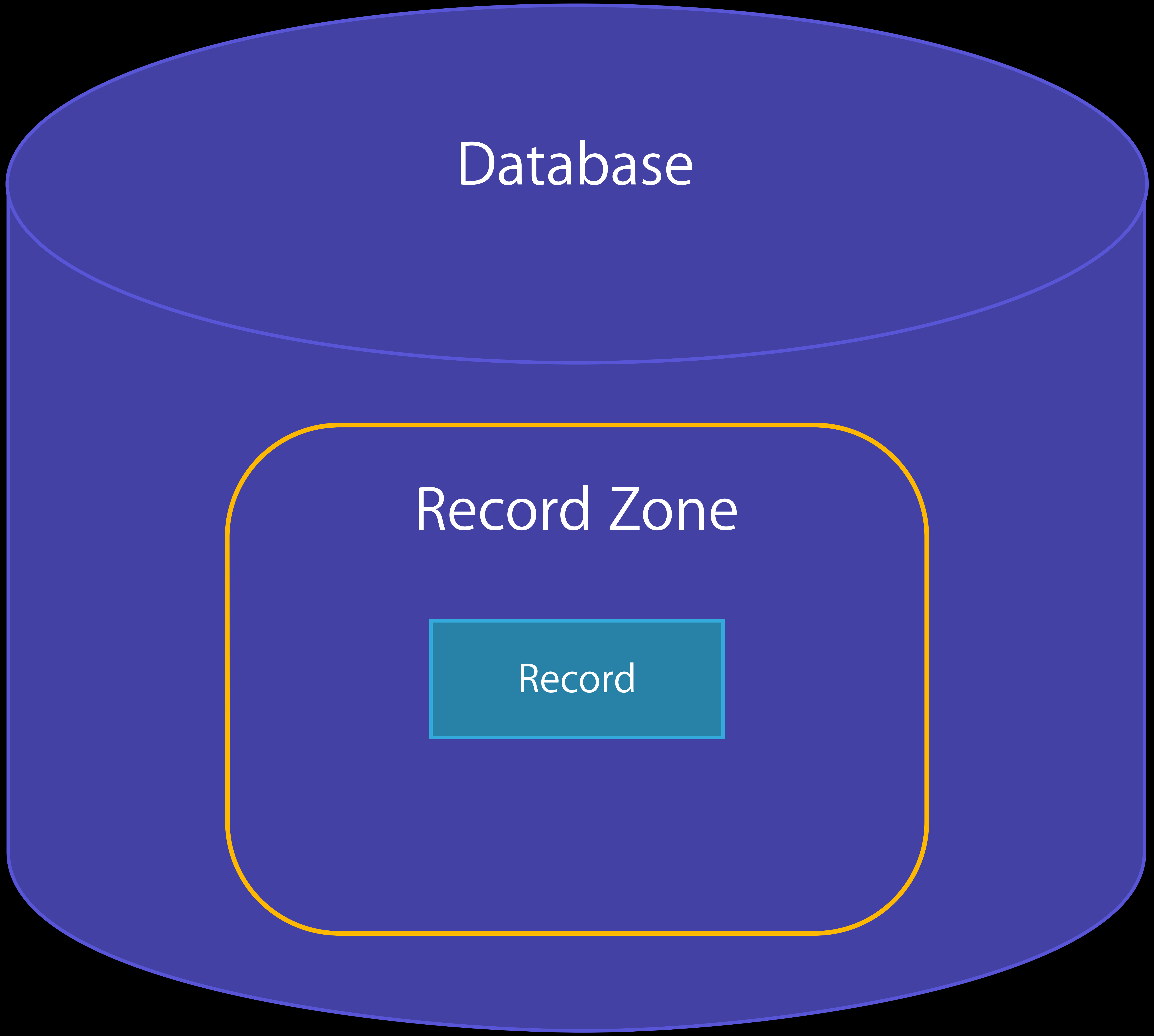
Fundamental CloudKit Objects

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

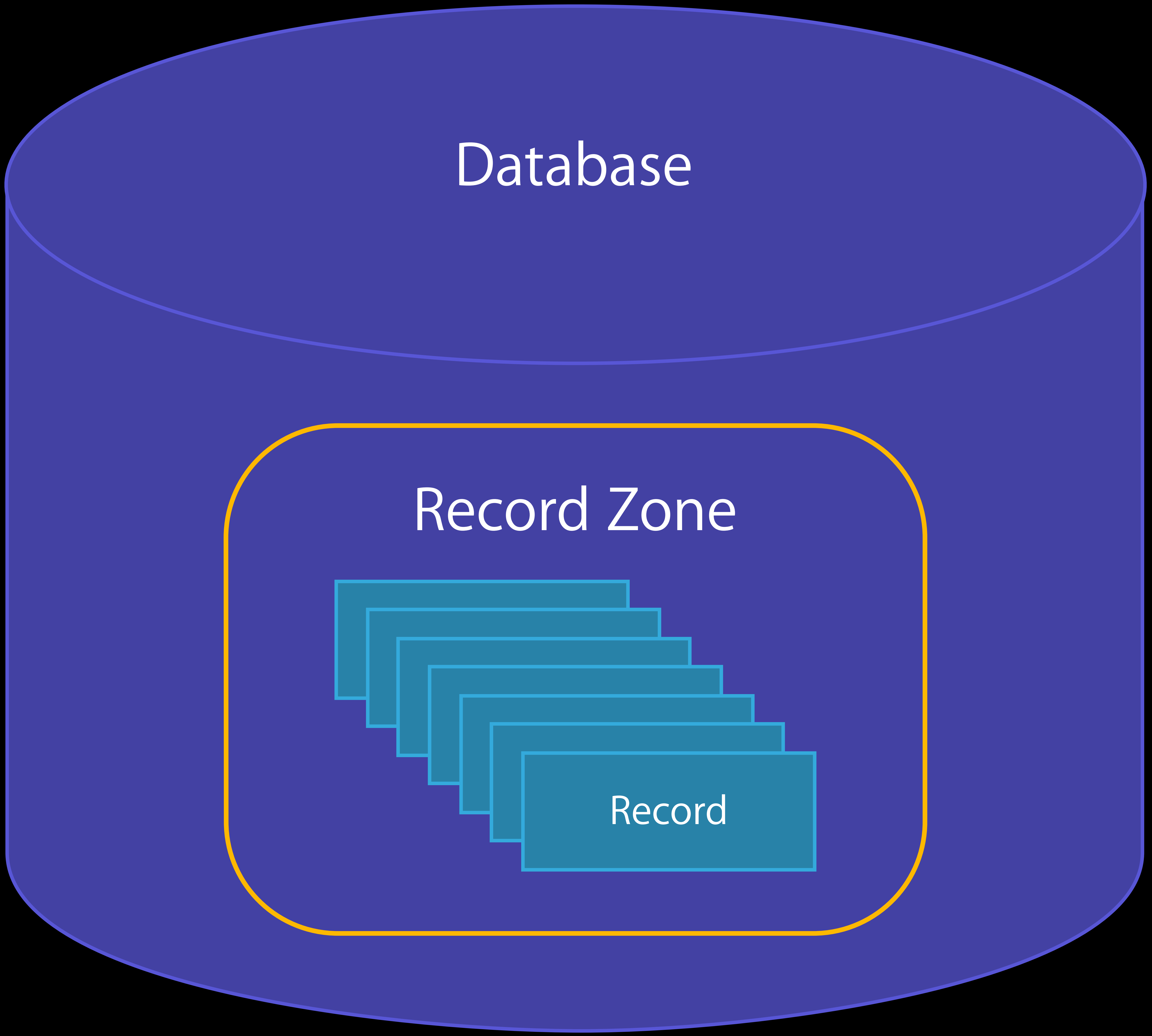
Record Zones



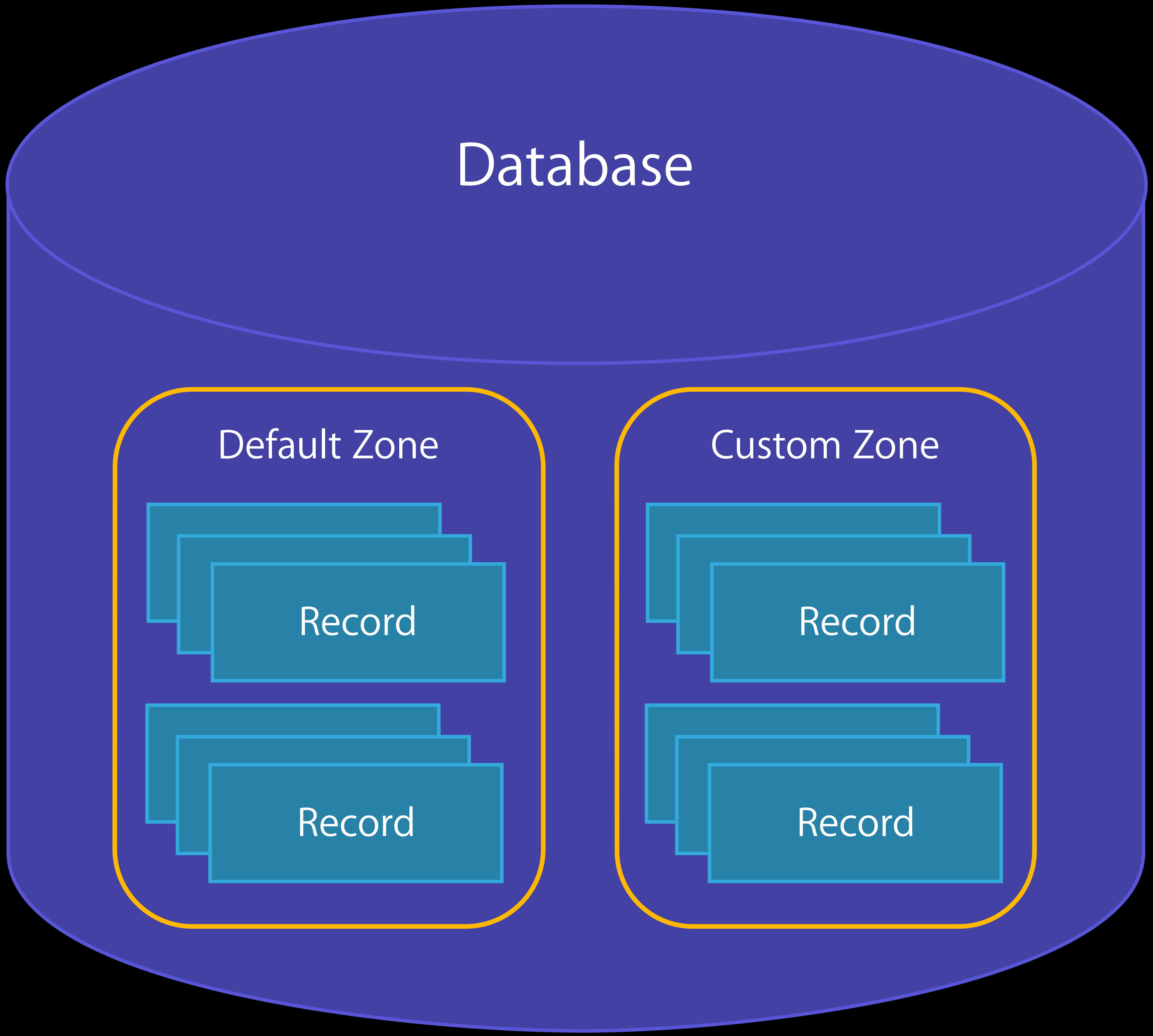
Record Zones



Record Zones



Record Zones



Fundamental CloudKit Objects

Containers

Databases

Records

Record Zones

Record Identifiers

References

Assets

Record Identifiers

Record Identifiers

```
@interface CKRecordID : NSObject <NSSecureCoding, NSCopying>
...
@property (nonatomic, readonly, strong) NSString *recordName;
@property (nonatomic, readonly, strong) CKRecordZoneID *zoneID;

@end
```

Record Identifiers

```
@interface CKRecordID : NSObject <NSSecureCoding, NSCopying>
...
@property (nonatomic, readonly, strong) NSString *recordName;
@property (nonatomic, readonly, strong) CKRecordZoneID *zoneID;

@end
```

- Created by the client

Record Identifiers

```
@interface CKRecordID : NSObject <NSSecureCoding, NSCopying>
...
@property (nonatomic, readonly, strong) NSString *recordName;
@property (nonatomic, readonly, strong) CKRecordZoneID *zoneID;

@end
```

- Created by the client
- Fully normalized: they represent the location of the record

Record Identifiers

```
@interface CKRecordID : NSObject <NSSecureCoding, NSCopying>
...
@property (nonatomic, readonly, strong) NSString *recordName;
@property (nonatomic, readonly, strong) CKRecordZoneID *zoneID;

@end
```

- Created by the client
- Fully normalized: they represent the location of the record
- External data set foreign key

Record Identifiers

Record Identifiers

```
CKRecord *party = [[CKRecord alloc] initWithRecordType:@“Party”];
```

Record Identifiers

```
CKRecord *party = [[CKRecord alloc] initWithRecordType:@“Party”];
```

```
@interface CKRecord : NSObject <NSSecureCoding, NSCopying>
...
- (instancetype)initWithRecordType:(NSString *)recordType;
- (instancetype)initWithRecordType:(NSString *)recordType
                          recordID:(CKRecordID *)recordID;
...
@end
```

Record Identifiers

```
CKRecord *party = [[CKRecord alloc] initWithRecordType:@“Party”];

@interface CKRecord : NSObject <NSSecureCoding, NSCopying>
...
- (instancetype)initWithRecordType:(NSString *)recordType;
- (instancetype)initWithRecordType:(NSString *)recordType
                          recordID:(CKRecordID *)recordID;
...
@end
```

```
CKRecordID *wellKnownID = [[CKRecordID alloc]
                          initWithRecordName:@"WellKnownParty"];
CKRecord *party = [[CKRecord alloc] initWithRecordType:@"Party"
                                              recordID:wellKnownID];
```

Fundamental CloudKit Objects

Containers

Databases

Records

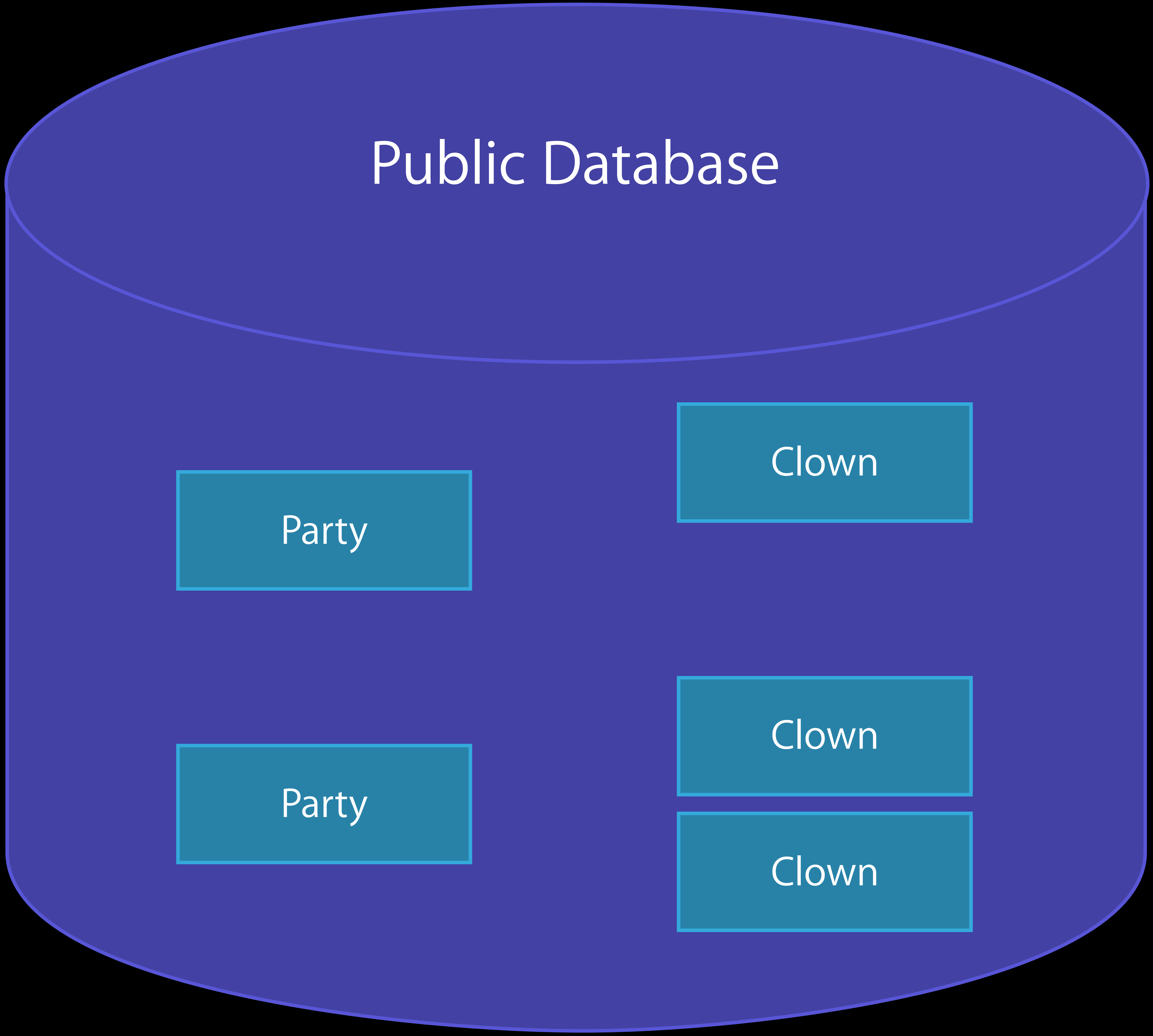
Record Zones

Record Identifiers

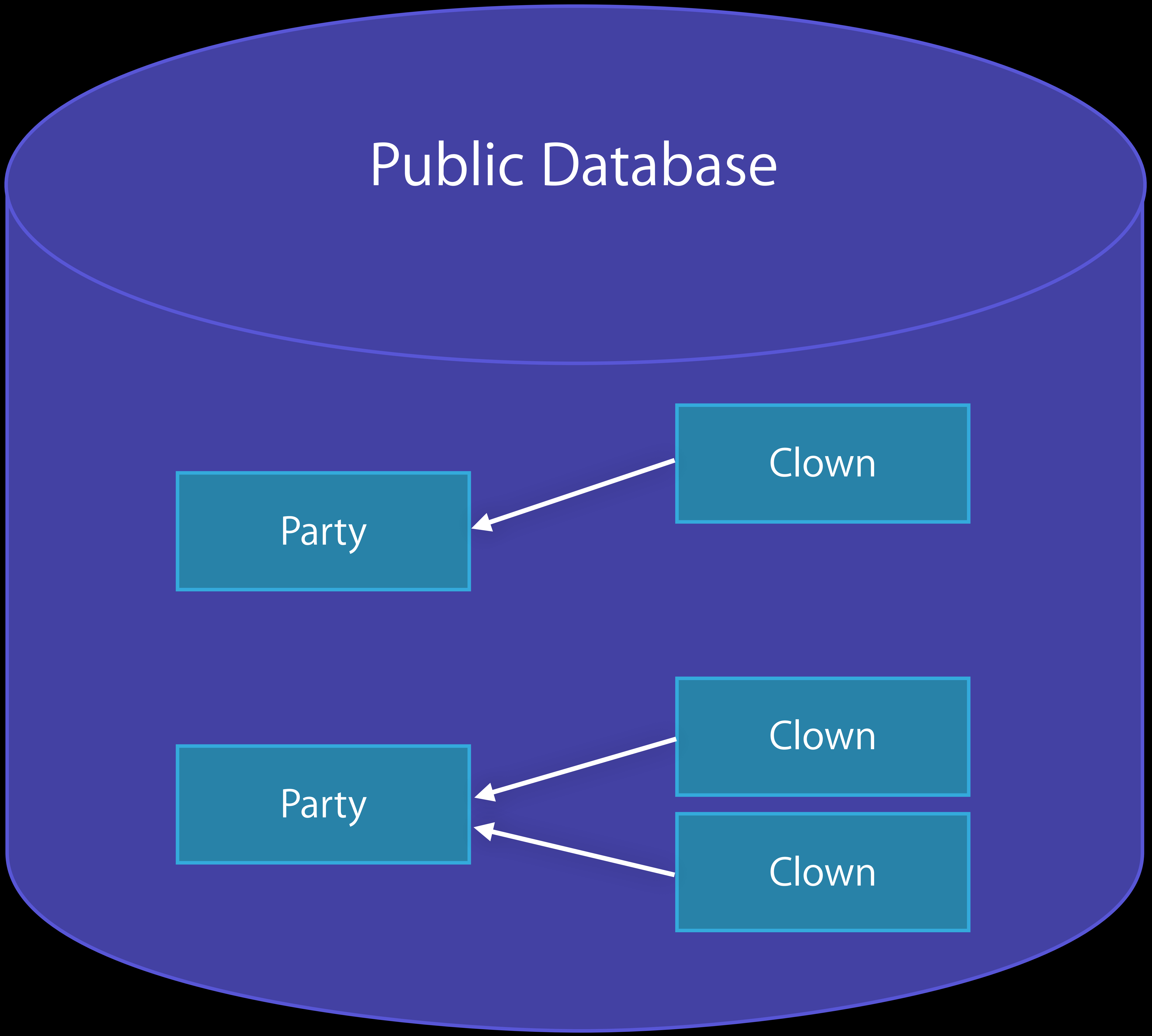
References

Assets

References



References



References

References

CKReference

References

CKReference

Server Understands Relationship

References

CKReference

Server Understands Relationship

Cascade Deletes

References

CKReference

Server Understands Relationship

Cascade Deletes

Dangling Pointers

References

CKReference

Server Understands Relationship

Cascade Deletes

Dangling Pointers

Back References

References

References

```
CKRecord *clown = [[CKRecord alloc] initWithRecordType:@“Clown”];
```

References

```
CKRecord *clown = [[CKRecord alloc] initWithRecordType:@“Clown”];

CKRecord *party = [[CKRecord alloc] initWithRecordType:@“Party”];
CKReference *partyReference = [[CKReference alloc]
                                 initWithRecord:party
                                 action:CKReferenceActionNone];
clown[@"party"] = partyReference;
```

References

```
CKRecord *clown = [[CKRecord alloc] initWithRecordType:@“Clown”];

CKRecord *party = [[CKRecord alloc] initWithRecordType:@“Party”];
CKReference *partyReference = [[CKReference alloc]
                                initWithRecord:party
                                action:CKReferenceActionNone];
clown[@"party"] = partyReference;
```

```
CKRecordID *wellKnownID = [[CKRecordID alloc]
                        initWithRecordName:@"WellKnownParty"];
CKReference *wellKnownReference = [[CKReference alloc]
                                initWithRecordID:wellKnownID
                                action:CKReferenceActionNone];
clown[@"party"] = wellKnownReference;
```

Fundamental CloudKit Objects

Containers

Databases

Records

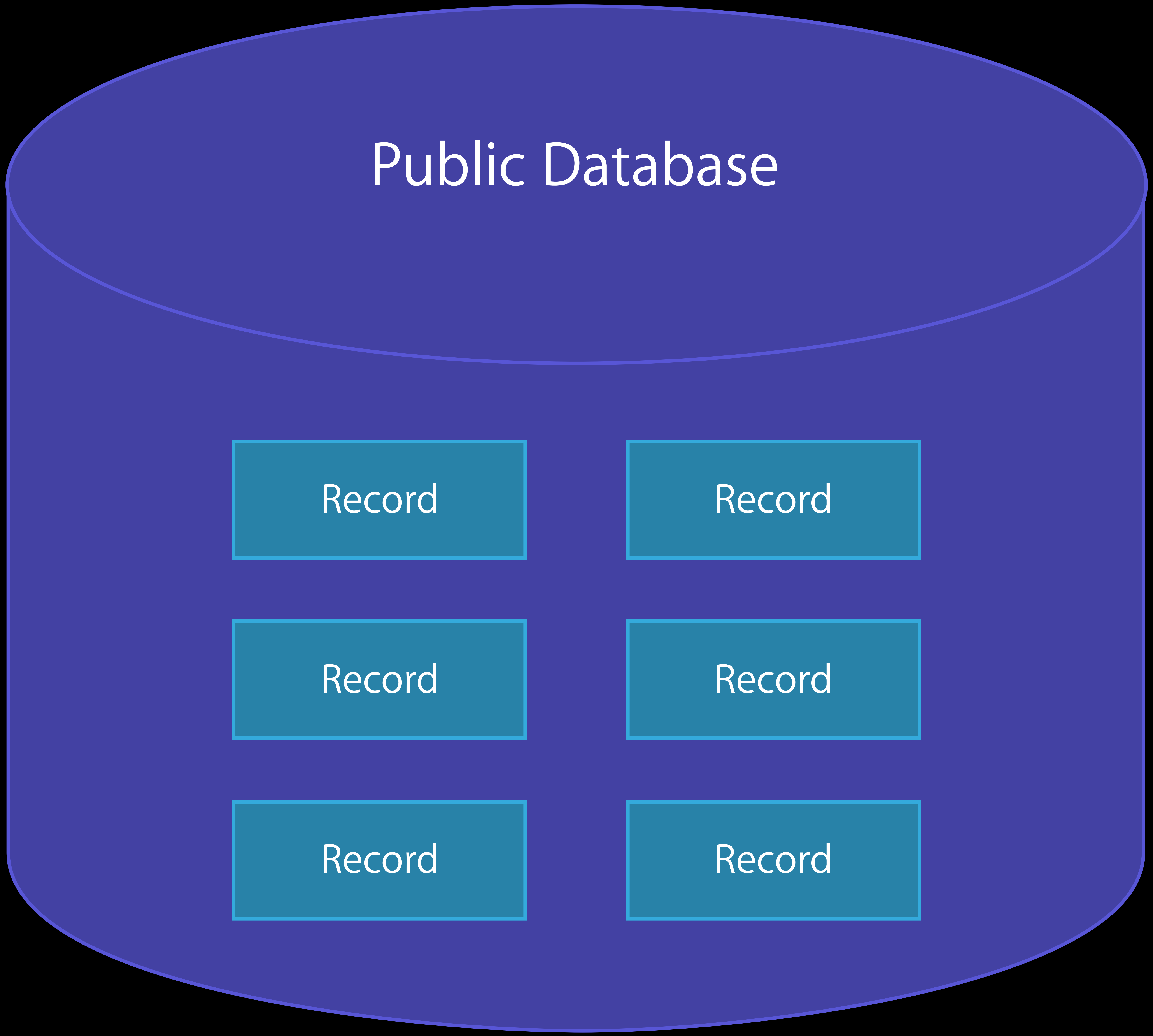
Record Zones

Record Identifiers

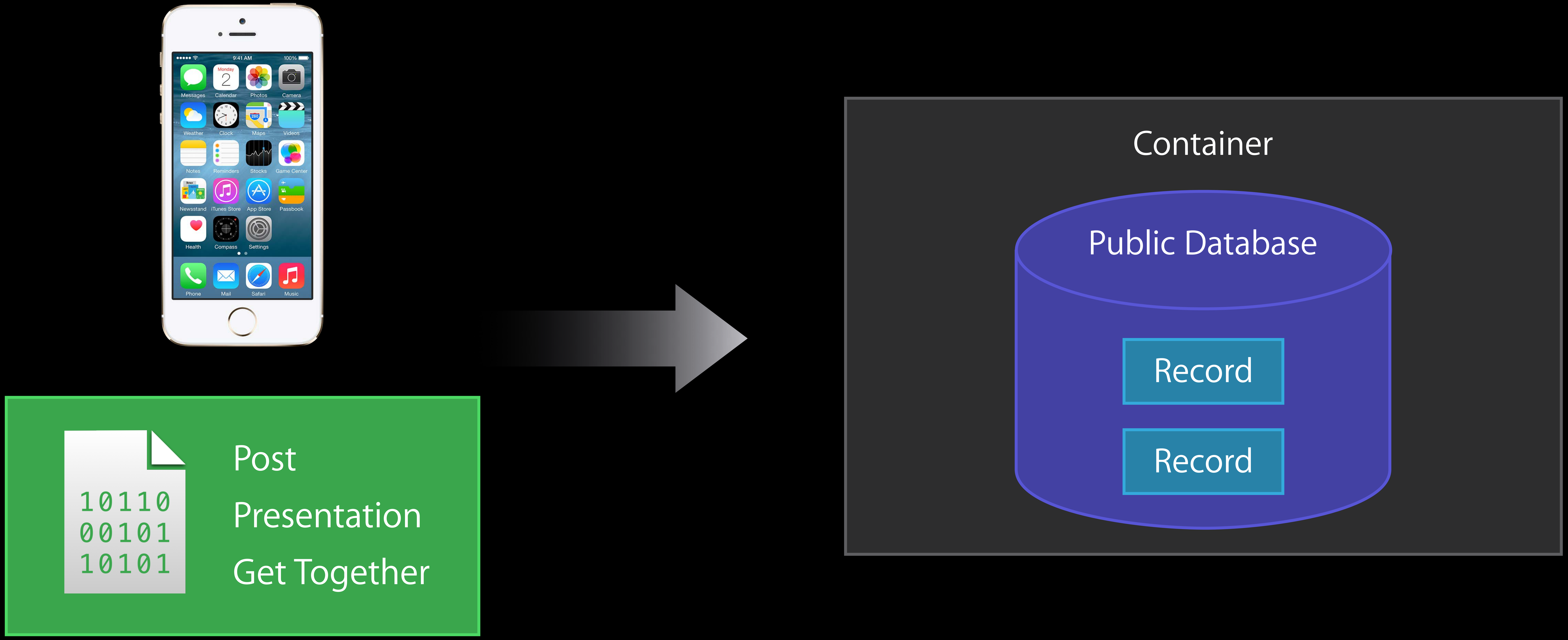
References

Assets

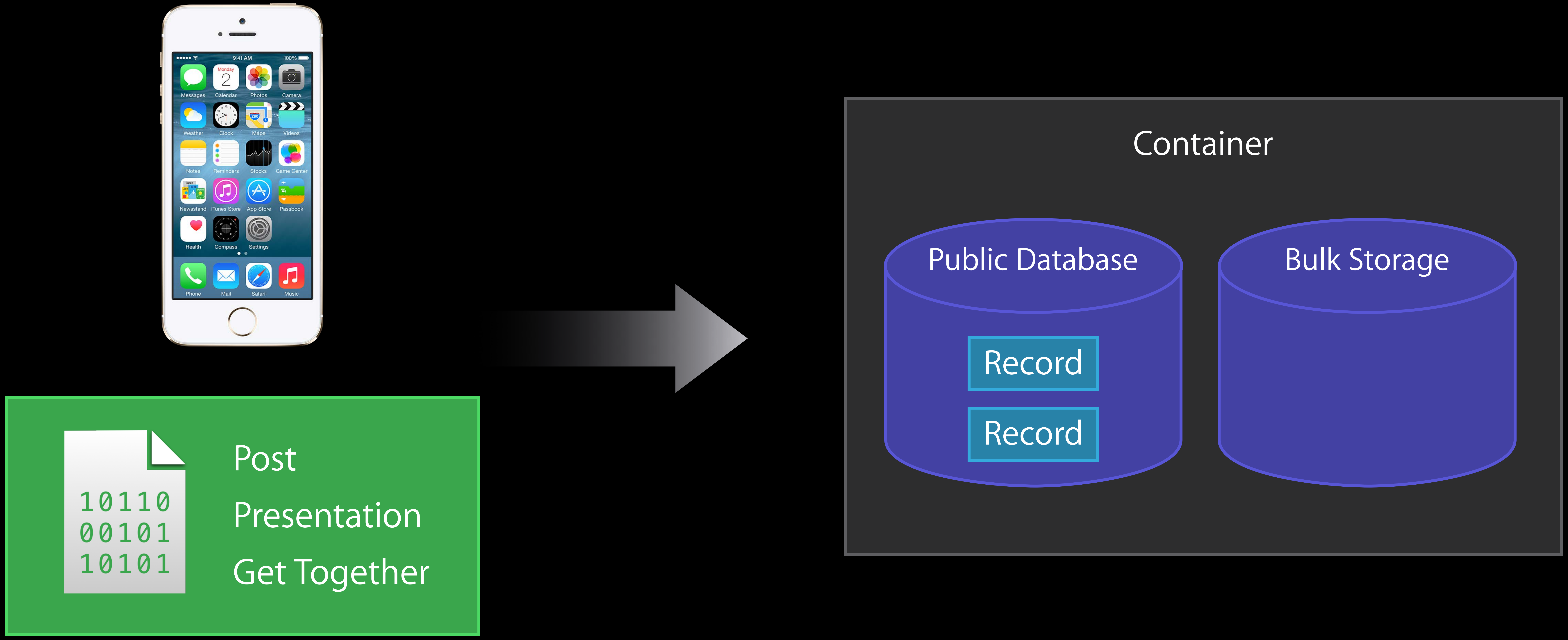
Assets


Public Database
Record
Record
Record
Record
Record
Record

Assets

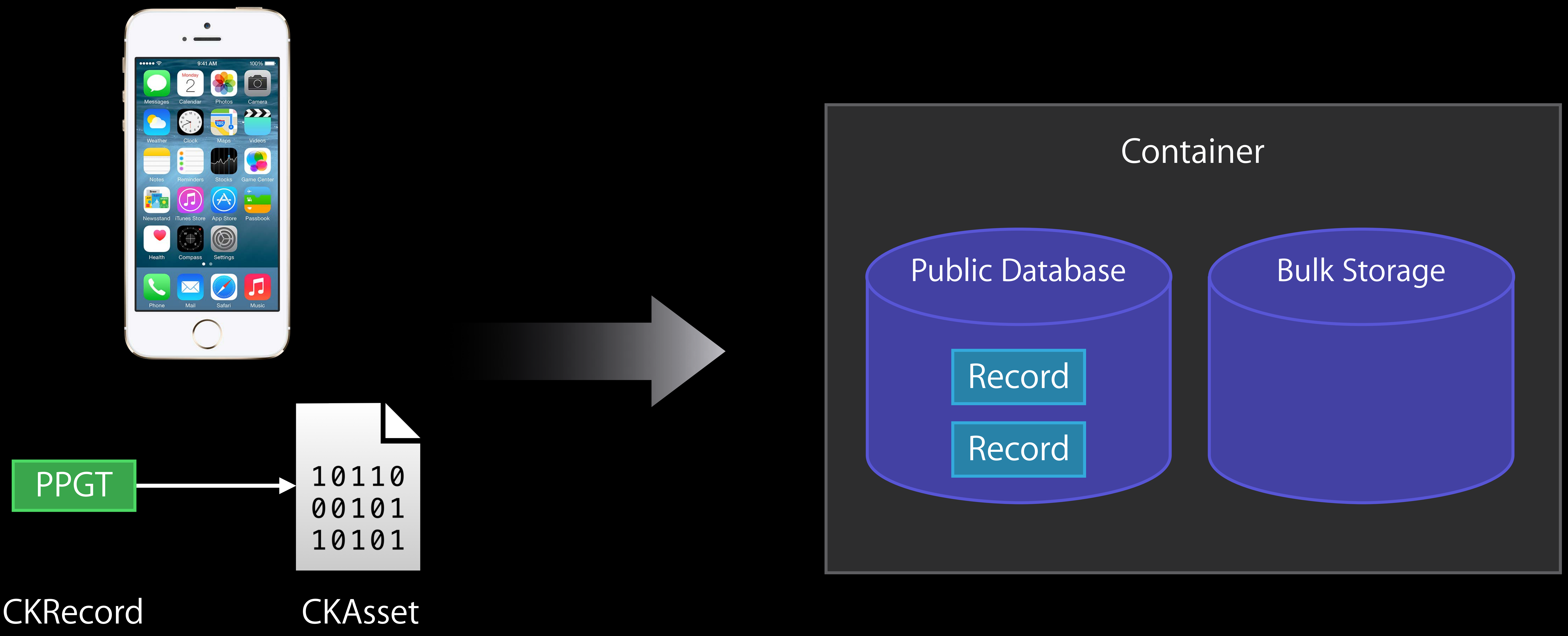

10110
00101
10101
Post
Presentation
Get Together
Container
Public Database
Record
Record

Assets

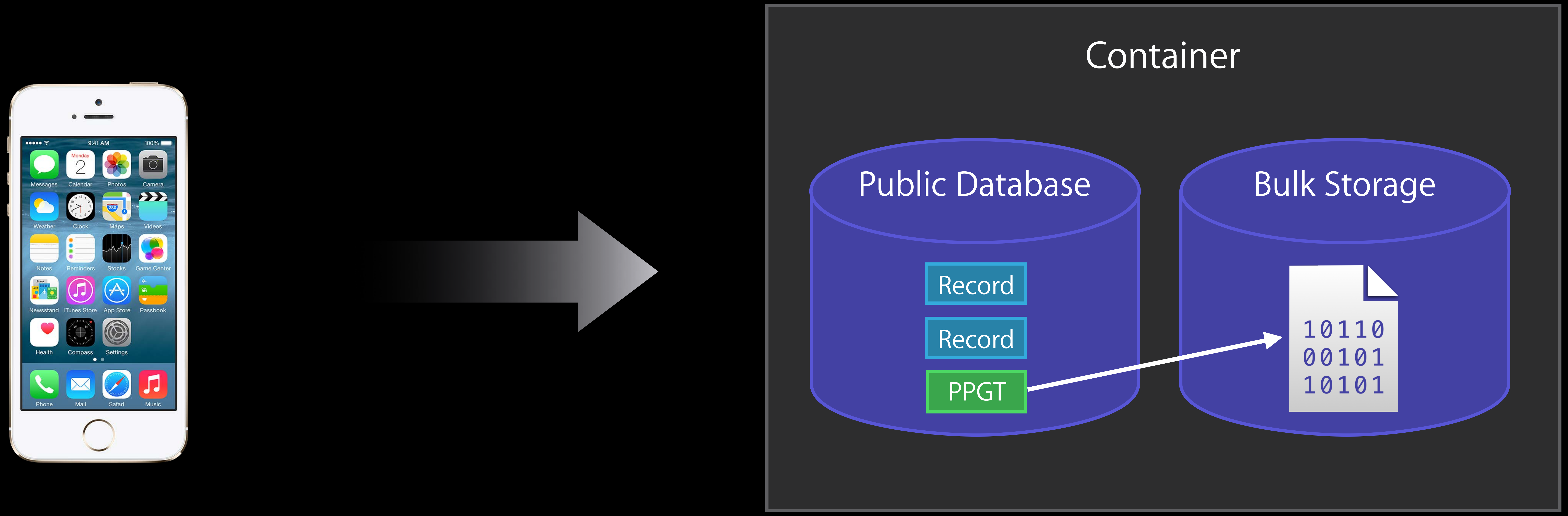


10110
00101
10101
Post
Presentation
Get Together
Container
Public Database
Record
Record
Bulk Storage

Assets


PPGT
10110
00101
10101
CKRecord
CKAsset
Container
Public Database
Record
Record
Bulk Storage

Assets


Container
Public Database
Record
Record
PPGT
Bulk Storage
10110
00101
10101

Assets

Assets

CKAsset

Assets

CKAsset

Large, unstructured data

Assets

CKAsset

Large, unstructured data

Files on disk

Assets

CKAsset

Large, unstructured data

Files on disk

Owned by CKRecords

Assets

CKAsset

Large, unstructured data

Files on disk

Owned by CKRecords

Garbage collected

Assets

CKAsset

Large, unstructured data

Files on disk

Owned by CKRecords

Garbage collected

Efficient uploads and downloads

Assets

Assets

```
NSURL *screenplayURL = [NSURL fileURLWithPath:@“...”];

CKAsset *screenplay = [[CKAsset alloc] initWithFileURL:screenplayURL];
```

Assets

```
NSURL *screenplayURL = [NSURL fileURLWithPath:@“...”];

CKAsset *screenplay = [[CKAsset alloc] initWithFileURL:screenplayURL];

CKRecord *party = [[CKRecord alloc] initWithRecordType:@“Party”];
party[@"screenplay"] = screenplay;
```

Fundamental CloudKit Objects

Containers

Databases

Records

Record Zones

Record Identifiers

References

Assets

CloudKit's Convenience API

Convenience API

Convenience API

Saving a record

Convenience API

Saving a record

Fetching a record

Convenience API

Saving a record

Fetching a record

Saving modified record

Convenience API

Saving a record

Fetching a record

Saving modified record

Saving a Record

Saving a Record

```
CKRecordID *wellKnownID = [[CKRecordID alloc]
                          initWithRecordName:@"WellKnownParty"];
CKRecord *party = [[CKRecord alloc] initWithRecordType:@"Party"
                                              recordID:wellKnownID];
```

Saving a Record

```
CKRecordID *wellKnownID = [[CKRecordID alloc]
                          initWithRecordName:@"WellKnownParty"];
CKRecord *party = [[CKRecord alloc] initWithRecordType:@"Party"
                                              recordID:wellKnownID];

CKDatabase *publicDatabase = [[CKContainer defaultContainer] publicCloudDatabase];
```

Saving a Record

```
CKRecordID *wellKnownID = [[CKRecordID alloc]
                          initWithRecordName:@"WellKnownParty"];
CKRecord *party = [[CKRecord alloc] initWithRecordType:@"Party"
                                              recordID:wellKnownID];

CKDatabase *publicDatabase = [[CKContainer defaultContainer] publicCloudDatabase];

[publicDatabase saveRecord:party
         completionHandler:^(CKRecord *savedParty, NSError *error) {

   // appropriate error handling when (error != nil)
}];
```

Convenience API

Saving a record

Fetching a record

Saving modified record

Fetching a Record

Fetching a Record

```
CKContainer *defaultContainer =[CKContainer defaultContainer];
CKDatabase *publicDatabase = [defaultContainer publicCloudDatabase];
```

Fetching a Record

```
CKContainer *defaultContainer =[CKContainer defaultContainer];
CKDatabase *publicDatabase = [defaultContainer publicCloudDatabase];

CKRecordID *wellKnownID = [[CKRecordID alloc]
                           initWithRecordName:@"WellKnownParty"];
```

Fetching a Record

```
CKContainer *defaultContainer =[CKContainer defaultContainer];
CKDatabase *publicDatabase = [defaultContainer publicCloudDatabase];

CKRecordID *wellKnownID = [[CKRecordID alloc]
                           initWithRecordName:@"WellKnownParty"];
```

```
[publicDatabase fetchRecordWithID:wellKnownID
                completionHandler:^(CKRecord *fetchedParty, NSError *error) {

   // truly marvelous error handling when (error != nil)
}];
```

Convenience API

Saving a record

Fetching a record

Saving modified record

Saving Modified Record

Saving Modified Record

```
CKDatabase *publicDatabase = ...;
CKRecordID *wellKnownID = ...;
[publicDatabase fetchRecordWithID:wellKnownID
                completionHandler:^(CKRecord *fetchedParty, NSError *error) {
    if (error) { ... } else {
```

Saving Modified Record

```
CKDatabase *publicDatabase = ...;
CKRecordID *wellKnownID = ...;
[publicDatabase fetchRecordWithID:wellKnownID
                completionHandler:^(CKRecord *fetchedParty, NSError *error) {
    if (error) { ... } else {

        NSDate *endDate = fetchedParty[@"end"];
        fetchedParty[@"end"] = [endDate dateByAddingTimeInterval:30.0 * 60.0];
```

Saving Modified Record

```
CKDatabase *publicDatabase = ...;
CKRecordID *wellKnownID = ...;
[publicDatabase fetchRecordWithID:wellKnownID
                completionHandler:^(CKRecord *fetchedParty, NSError *error) {
    if (error) { ... } else {

        NSDate *endDate = fetchedParty[@"end"];
        fetchedParty[@"end"] = [endDate dateByAddingTimeInterval:30.0 * 60.0];

        [publicDatabase saveRecord:fetchedParty
               completionHandler:^(CKRecord *savedParty, NSError *saveError) {

            // error handling to make your mother proud when (error != nil)
        }];
    }
}];
```

Convenience API

Saving a record

Fetching a record

Saving modified record

Big Data, Tiny Phone

Big Data, Tiny Phone

Big Data, Tiny Phone

Keep your large data in the cloud

Big Data, Tiny Phone

Keep your large data in the cloud

Client views slice of that data

Big Data, Tiny Phone

Keep your large data in the cloud

Client views slice of that data

Client view can change

Big Data, Tiny Phone

Keep your large data in the cloud

Client views slice of that data

Client view can change

Clients use queries to focus their viewpoint

Queries

Queries

CKQuery

Queries

CKQuery

Combine a RecordType, a NSPredicate, and NSSortDescriptors

Queries

CKQuery

Combine a RecordType, a NSPredicate, and NSSortDescriptors

- CloudKit supports a subset of NSPredicate

Queries

Predicates

Queries

Predicates

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

Queries

Predicates

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

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

Queries

Predicates

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

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

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

Queries

Predicates

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

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

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

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

Queries

Predicates

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

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

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

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

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

Queries

Predicates

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

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

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

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

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

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

Queries

Creating

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

CKQuery *query = [[CKQuery alloc] initWithRecordType:@"Party"
                                           predicate:predicate];
```

Queries

Performing

Queries

Performing

```
CKQuery *query = ...;

[[publicDatabase performQuery:query
                 inZoneWithID:nil
            completionHandler:^(NSArray *results, NSError *error) {
```

Queries

Performing

```
CKQuery *query = ...;

[[publicDatabase performQuery:query
                 inZoneWithID:nil
            completionHandler:^(NSArray *results, NSError *error) {
    // astounding error handling when (error != nil)

    if (!error) {
        NSLog(@"Fetch %ld results", (long)[results count]);
        for (CKRecord *record in results) {
            NSLog(@"Found matching party %@", record);
        }
    }
}];
```

Big Data, Tiny Phone

Big Data, Tiny Phone

Queries are polls

Big Data, Tiny Phone

Queries are polls

Great for slicing through large server data

Big Data, Tiny Phone

Queries are polls

Great for slicing through large server data

Bad for large, mostly static data set

Big Data, Tiny Phone

Queries are polls

Great for slicing through large server data

Bad for large, mostly static data set

- Battery life

Big Data, Tiny Phone

Queries are polls

Great for slicing through large server data

Bad for large, mostly static data set

- Battery life
- Networking traffic

Big Data, Tiny Phone

Queries are polls

Great for slicing through large server data

Bad for large, mostly static data set

- Battery life
- Networking traffic
- User experience

Big Data, Tiny Phone

Queries are polls

Great for slicing through large server data

Bad for large, mostly static data set

- Battery life
- Networking traffic
- User experience

What you want is the server running your query

Big Data, Tiny Phone

Queries are polls

Great for slicing through large server data

Bad for large, mostly static data set

- Battery life
- Networking traffic
- User experience

What you want is the server running your query

- ... in the background

Big Data, Tiny Phone

Queries are polls

Great for slicing through large server data

Bad for large, mostly static data set

- Battery life
- Networking traffic
- User experience

What you want is the server running your query

- ... in the background
- ... after every record save

Big Data, Tiny Phone

Queries are polls

Great for slicing through large server data

Bad for large, mostly static data set

- Battery life
- Networking traffic
- User experience

What you want is the server running your query

- ... in the background
- ... after every record save
- ... and you want push

Subscriptions

Subscriptions

CKSubscription

Subscriptions

CKSubscription

Combine a RecordType, a NSPredicate, and Push

Subscriptions

CKSubscription

Combine a RecordType, a NSPredicate, and Push

- Push via Apple Push Service

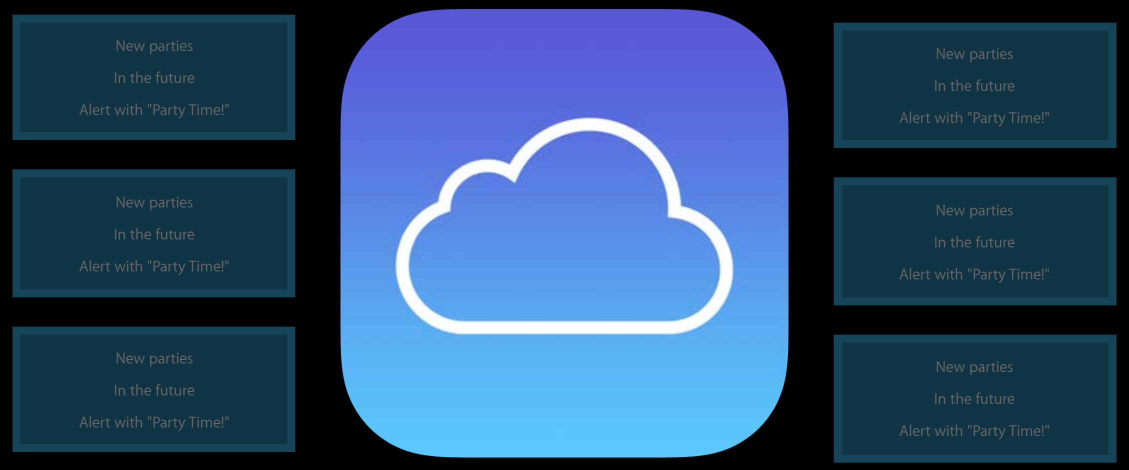
Subscriptions

CKSubscription

Combine a RecordType, a NSPredicate, and Push

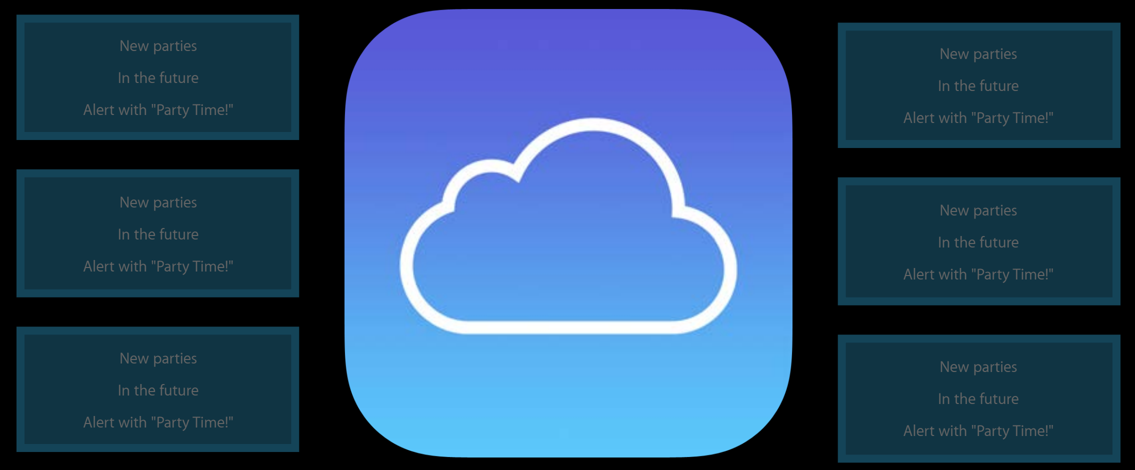
- Push via Apple Push Service
- Augmented payload

Subscriptions



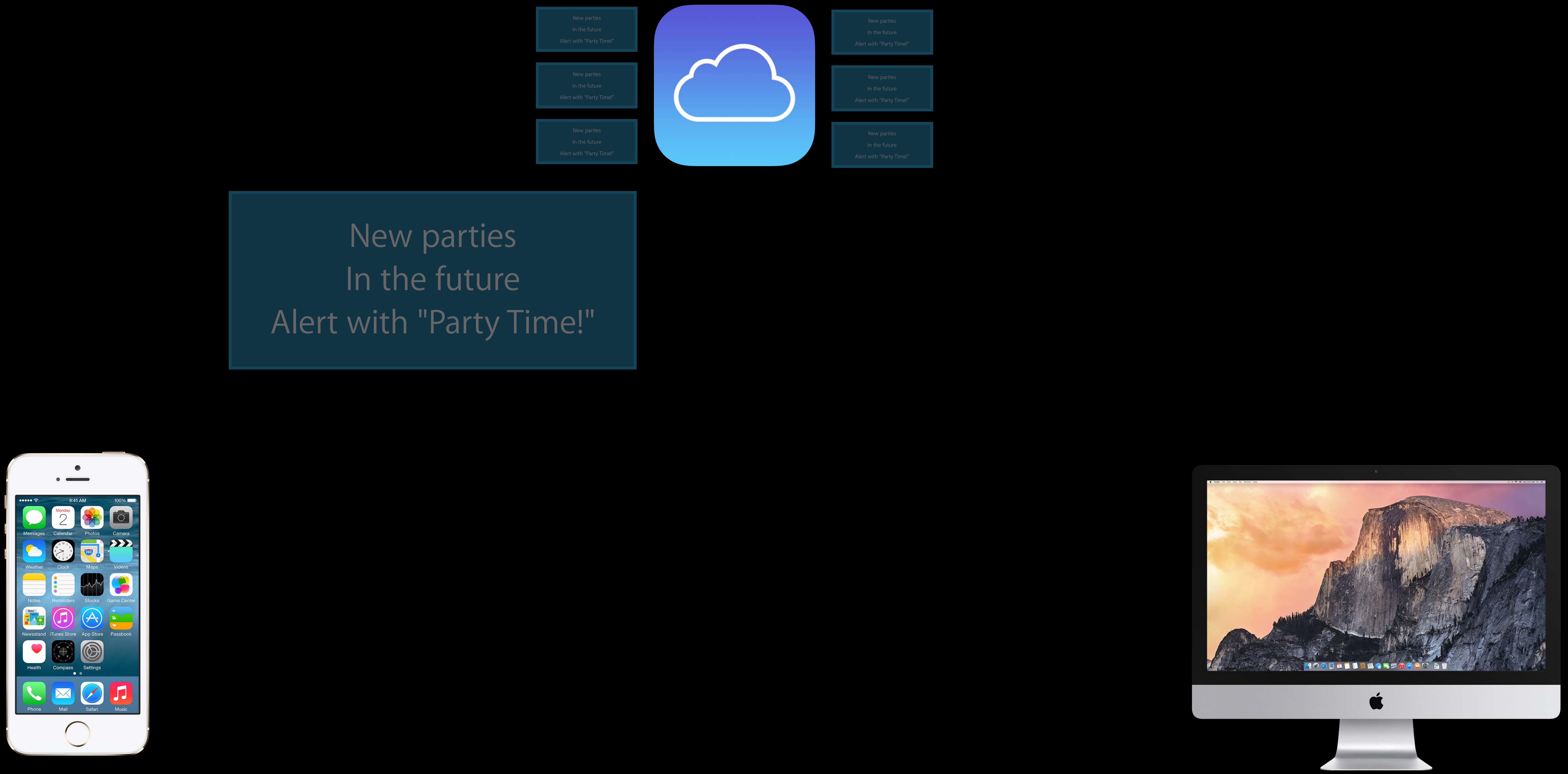
New parties
In the future
Alert with "Party Time!"

Subscriptions

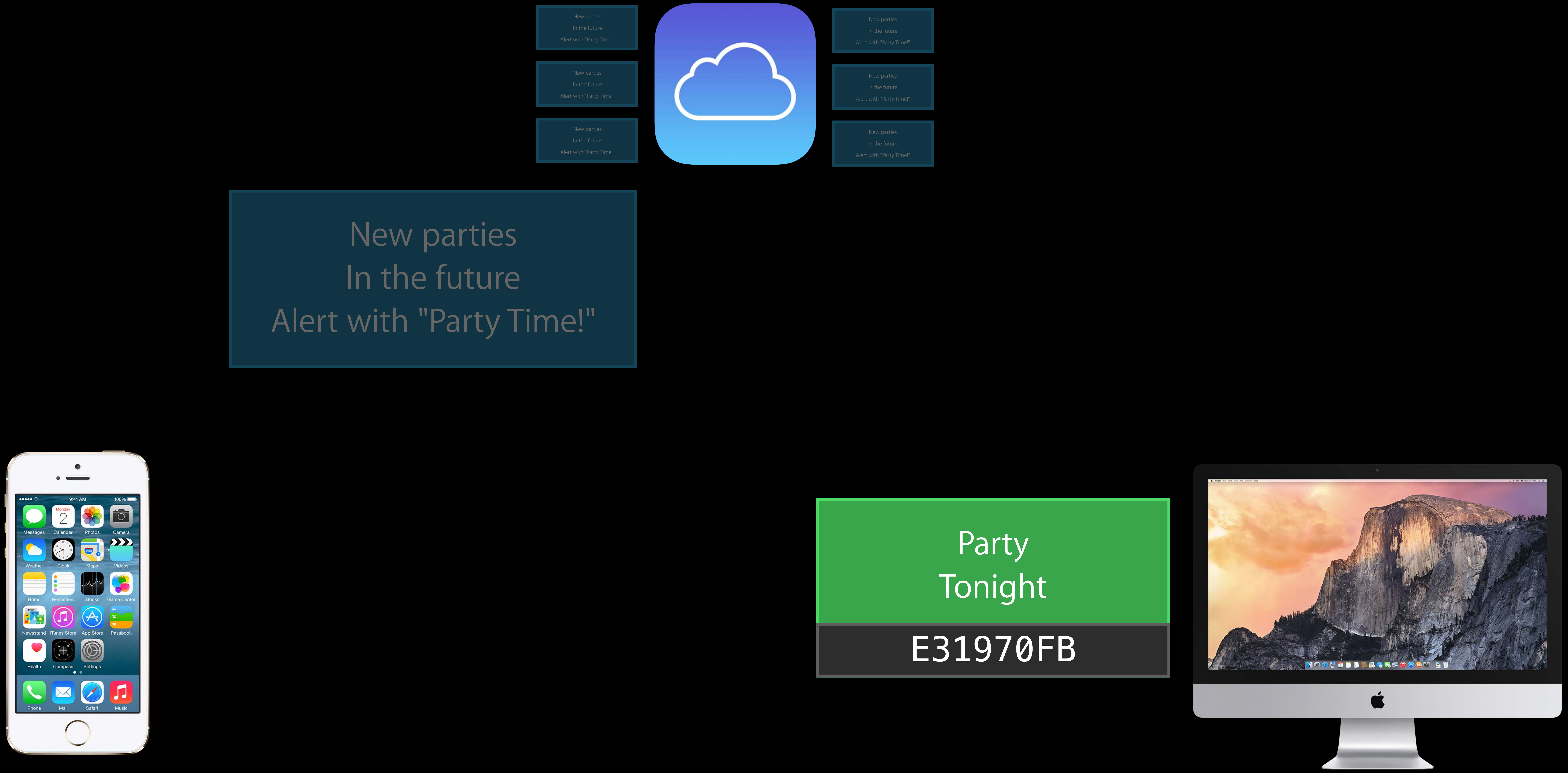


New parties
In the future
Alert with "Party Time!"

Subscriptions



Subscriptions

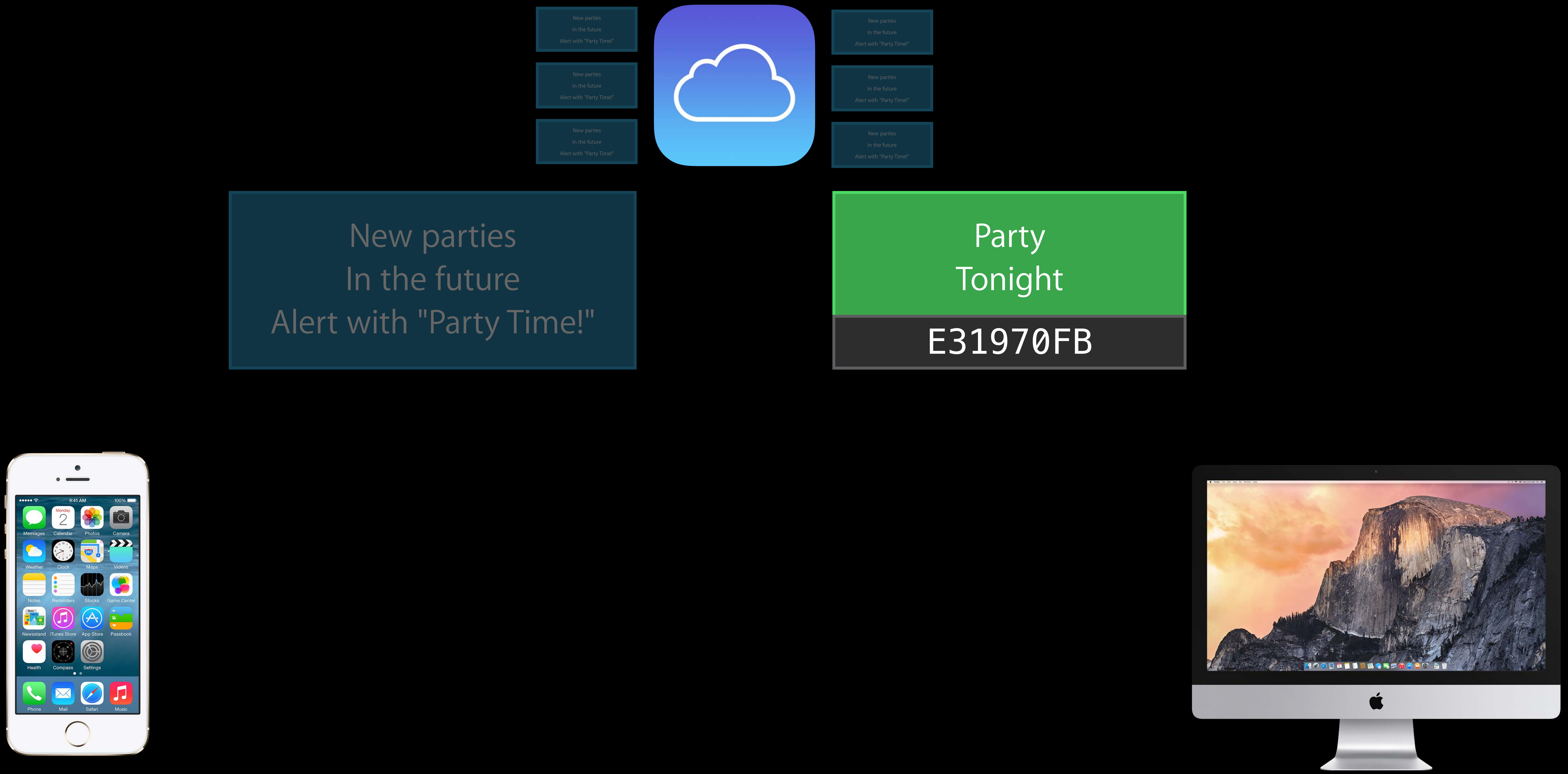


New parties
In the future
Alert with "Party Time!"

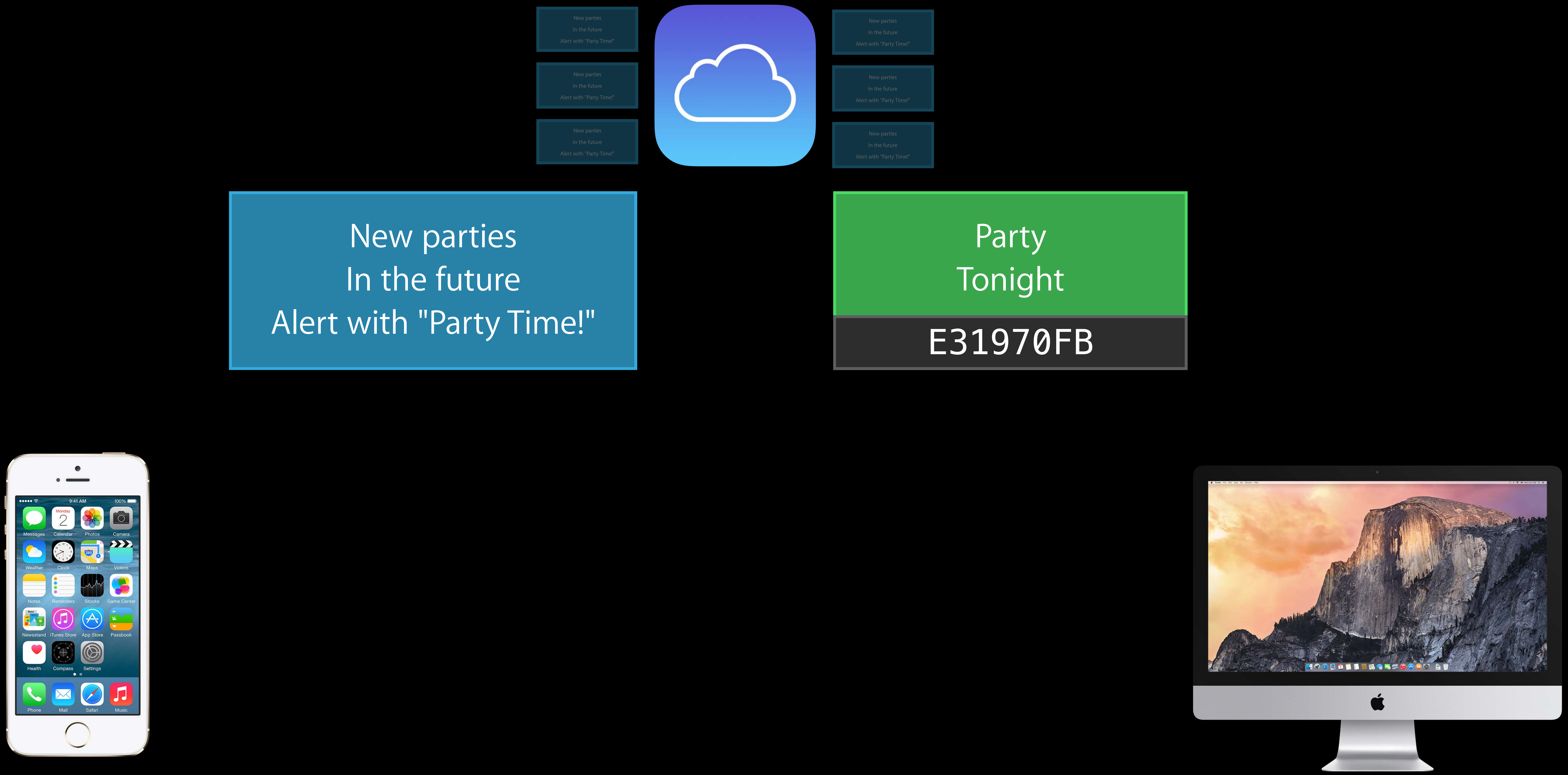
Party
Tonight

E31970FB

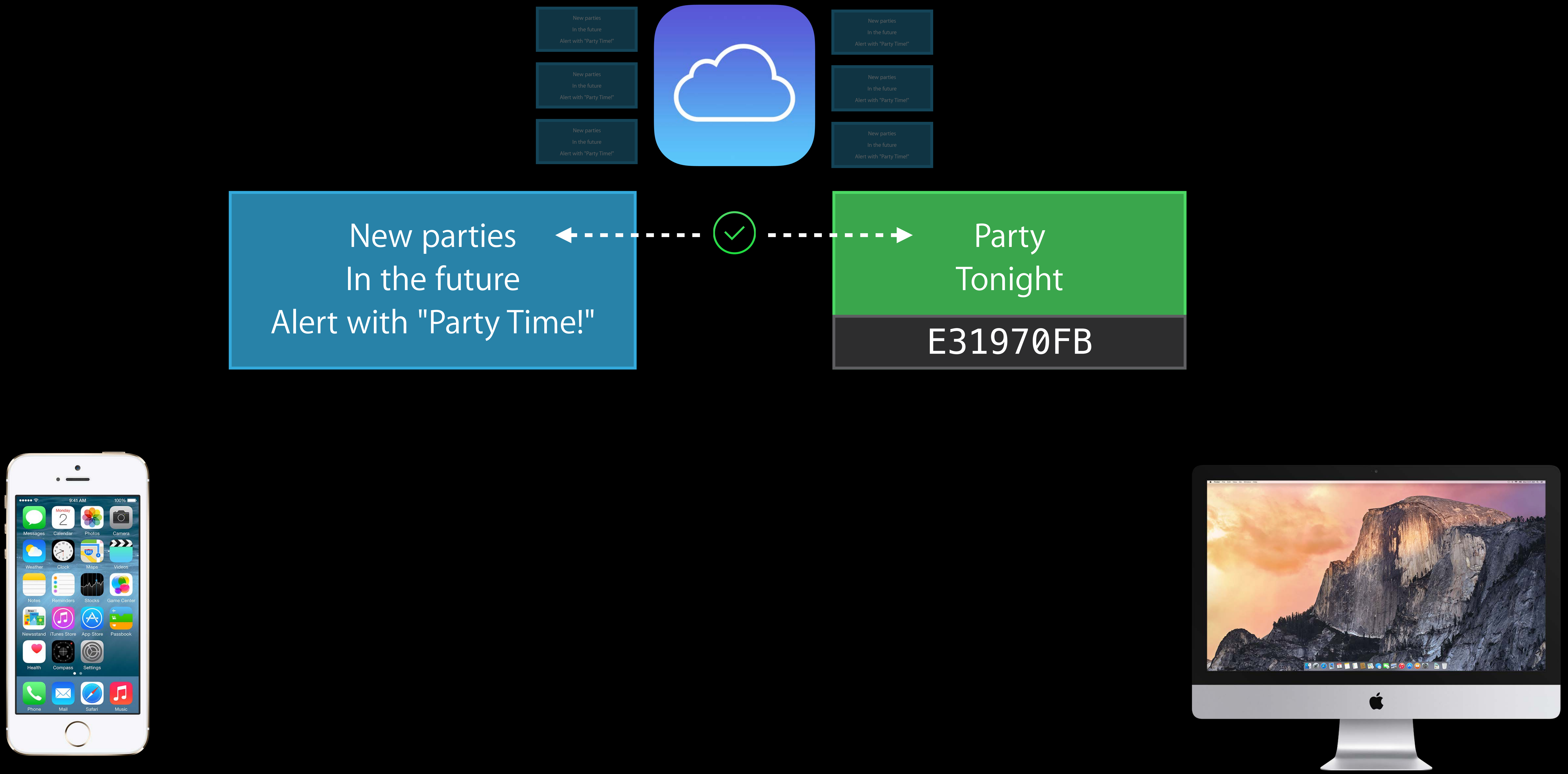
Subscriptions



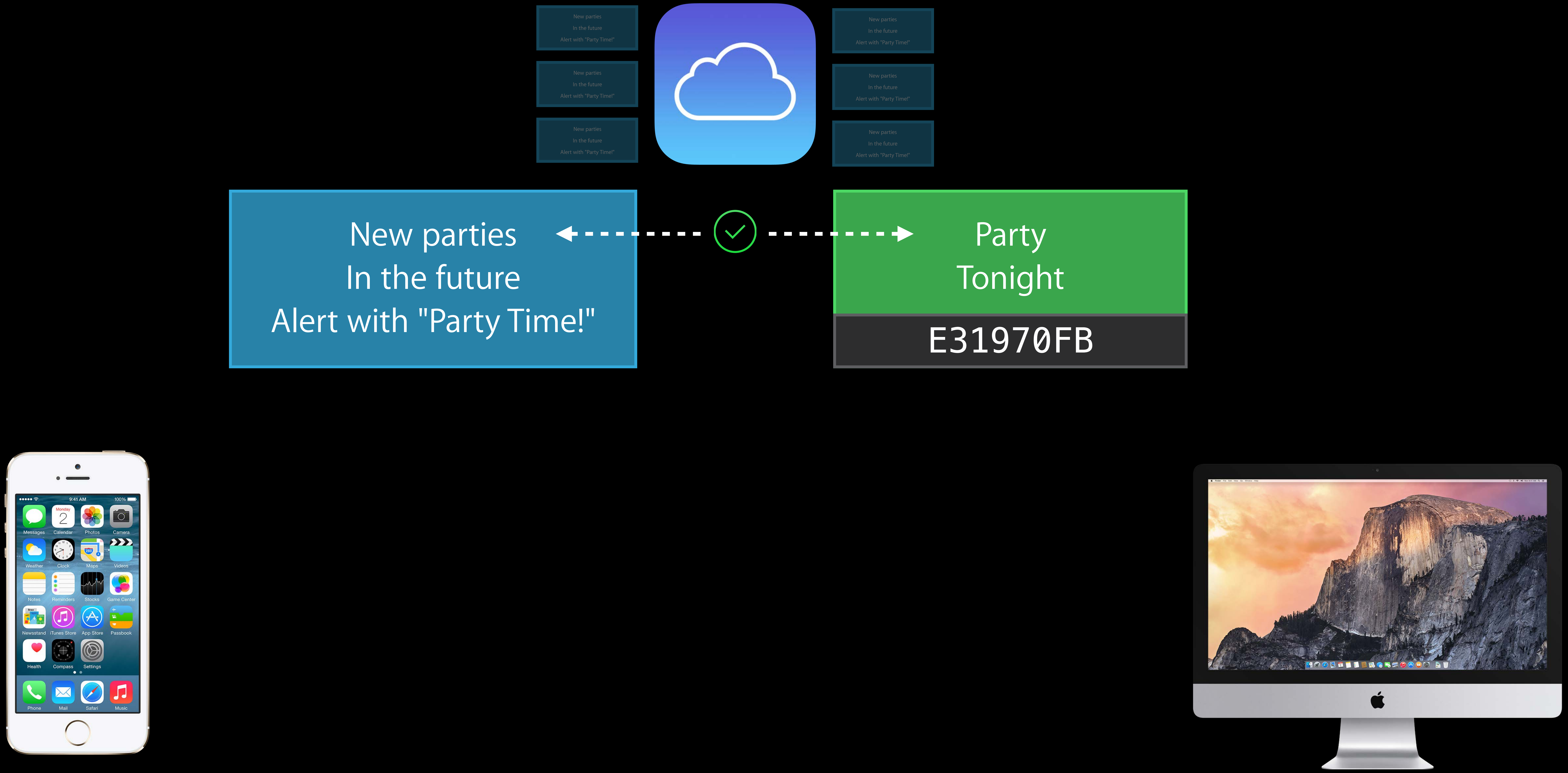
Subscriptions



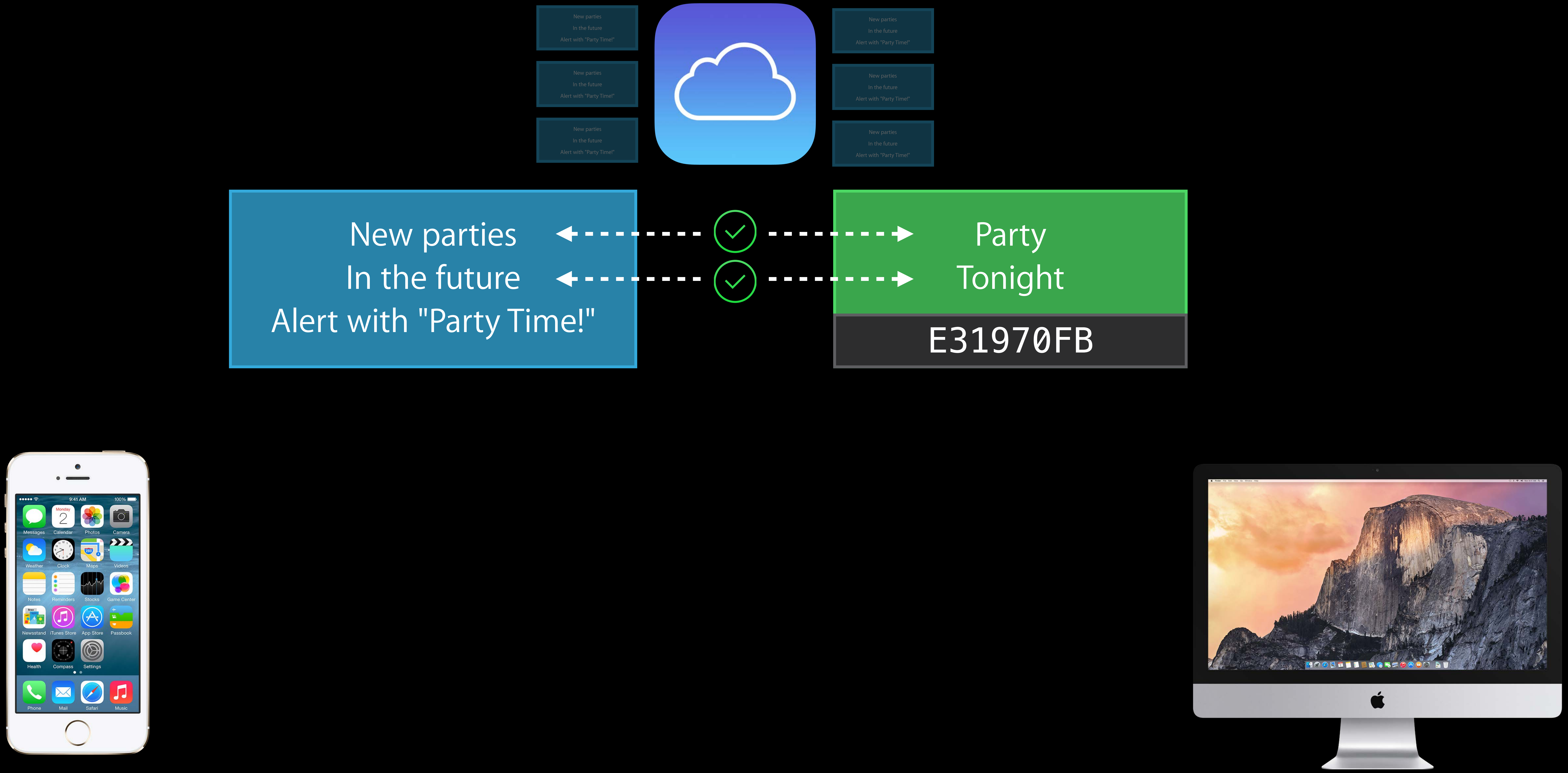
Subscriptions



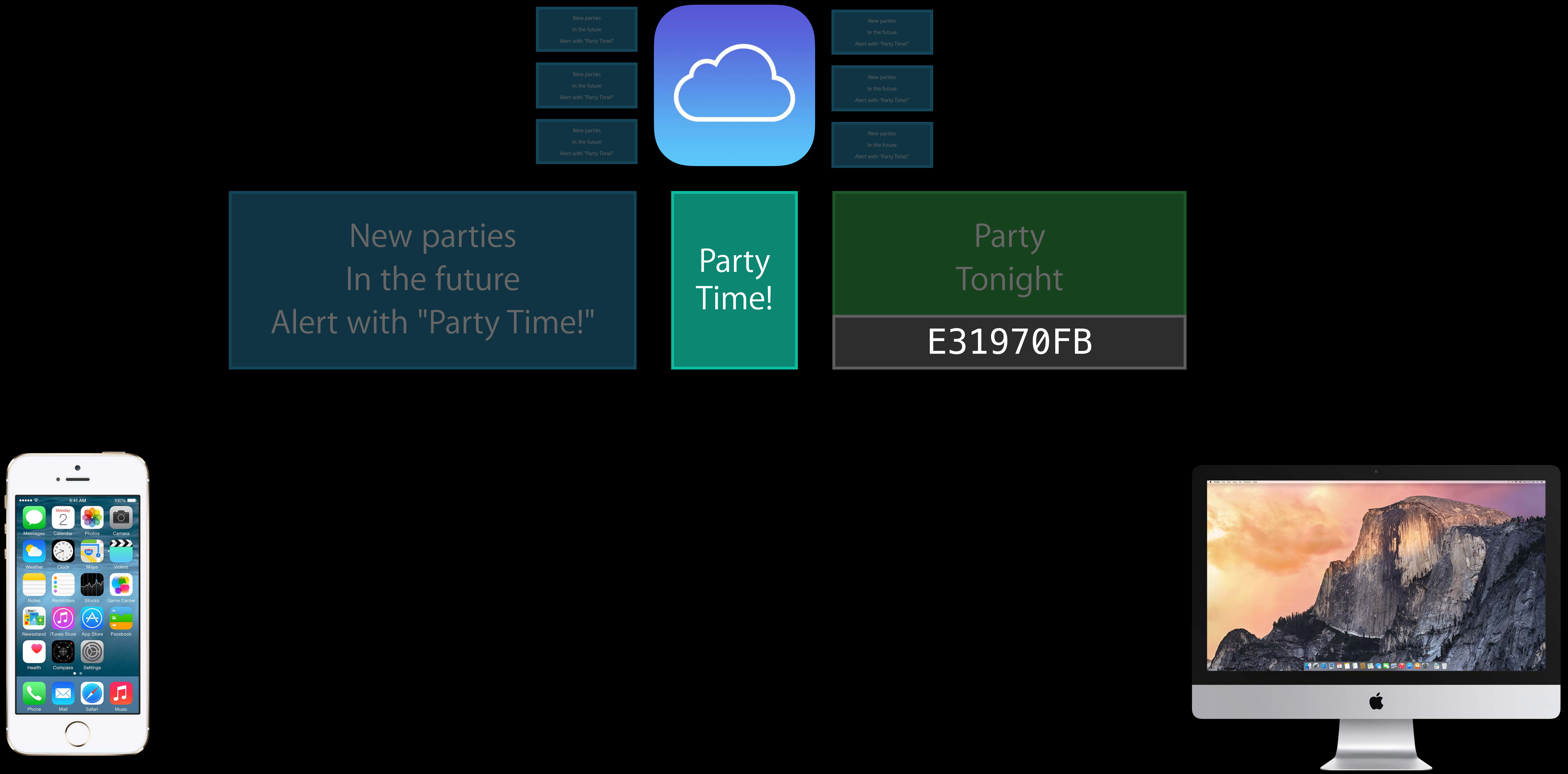
Subscriptions



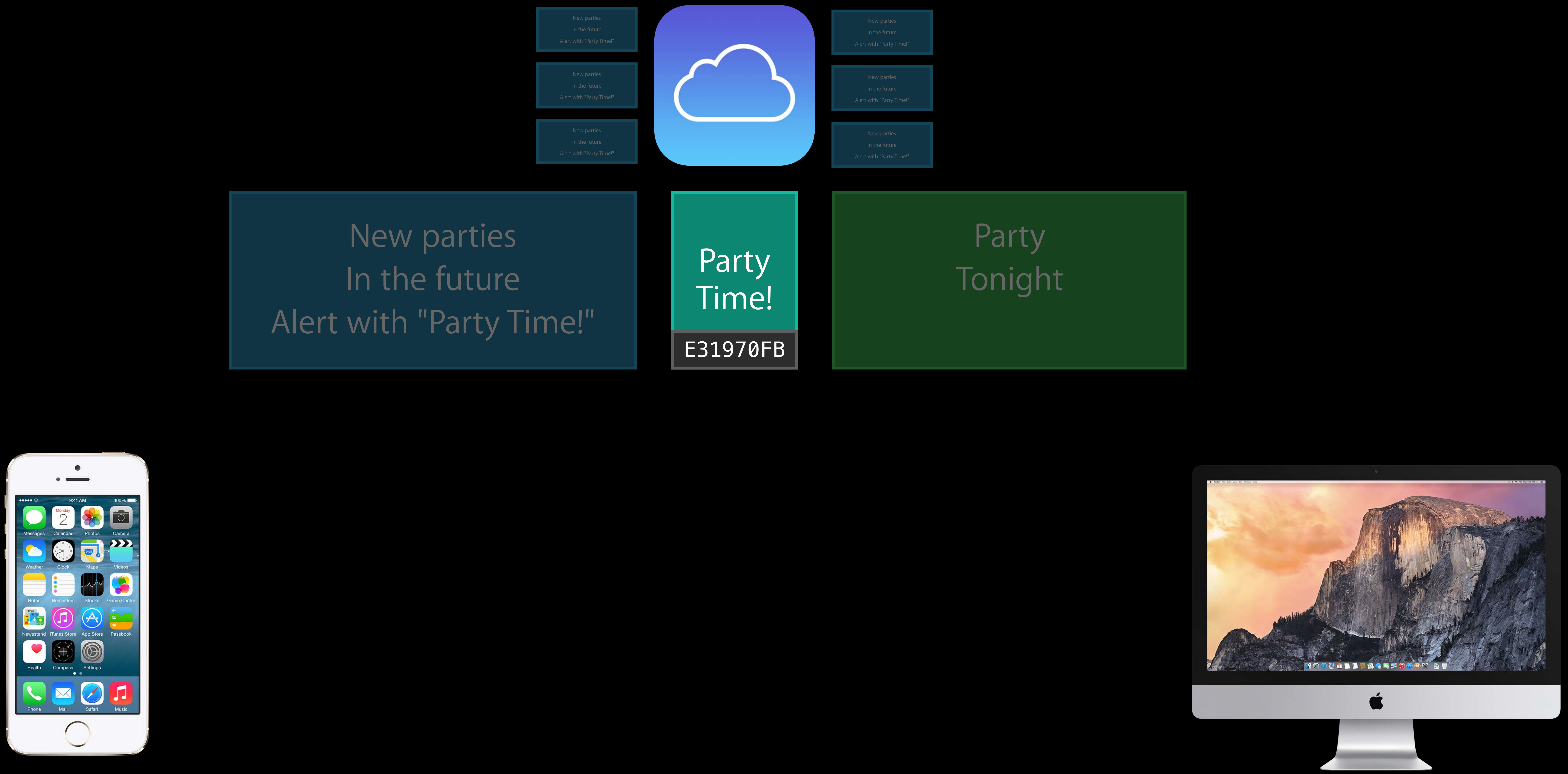
Subscriptions



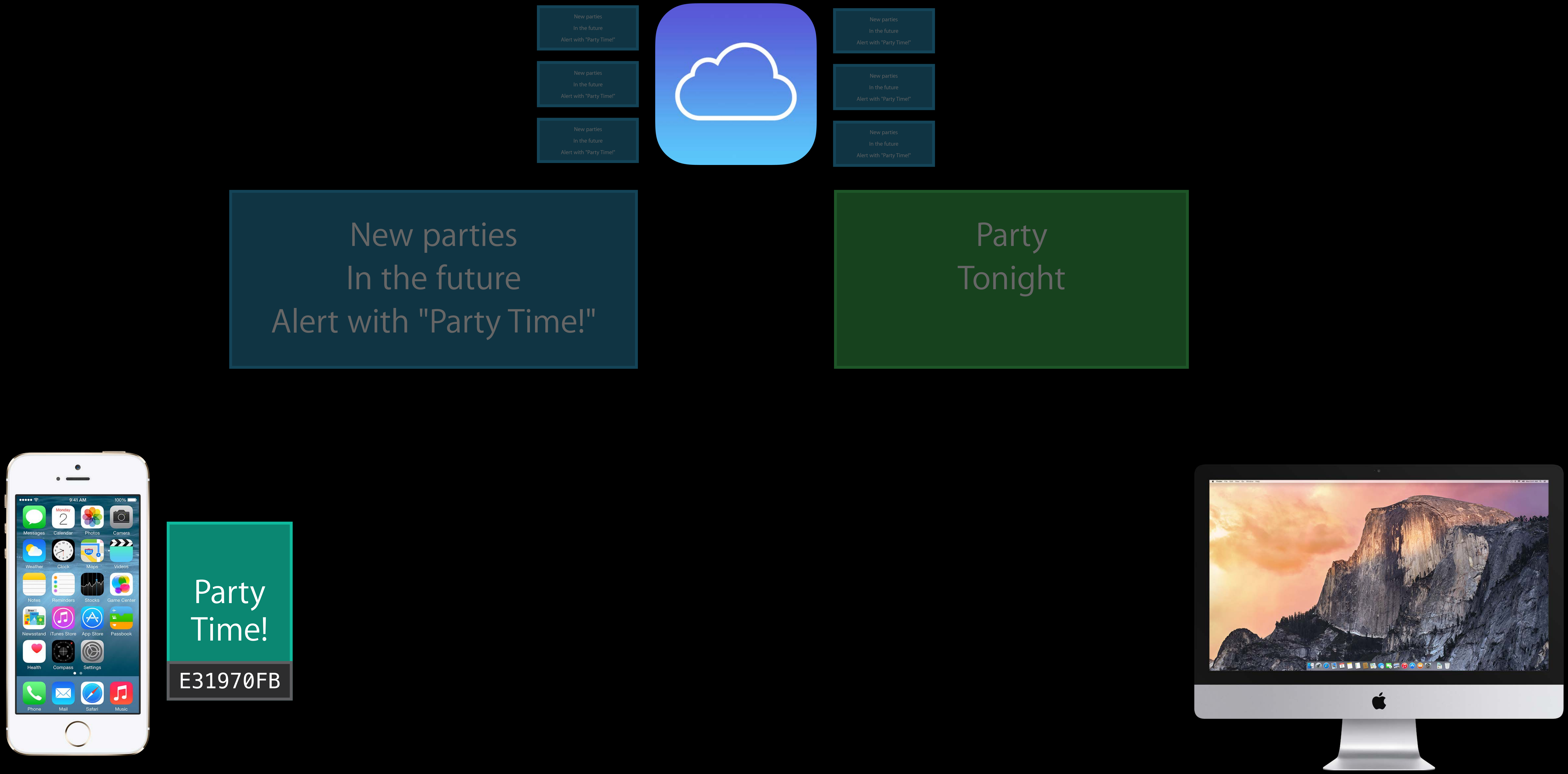
Subscriptions



Subscriptions



Subscriptions



Subscriptions

Creating

Subscriptions

Creating

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

Subscriptions

Creating

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

```
CKSubscription *subscription = [[CKSubscription alloc]
       initWithRecordType:@"Party"
                predicate:predicate
                  options:CKSubscriptionOptionsFiresOnRecordCreation];
```

Subscriptions

Creating

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

 CKSubscription *subscription = [[CKSubscription alloc]
        initWithRecordType:@"Party"
                 predicate:predicate
                   options:CKSubscriptionOptionsFiresOnRecordCreation];

 CKNotificationInfo *notificationInfo = [CKNotificationInfo new];
 notificationInfo.alertLocalizationKey = @"LOCAL_NOTIFICATION_KEY";
 notificationInfo.soundName = @"Party.aiff";
 notificationInfo.shouldBadge = YES;
```

Subscriptions

Creating

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

CKSubscription *subscription = [[CKSubscription alloc]
       initWithRecordType:@"Party"
                predicate:predicate
                  options:CKSubscriptionOptionsFiresOnRecordCreation];

CKNotificationInfo *notificationInfo = [CKNotificationInfo new];
notificationInfo.alertLocalizationKey = @"LOCAL_NOTIFICATION_KEY";
notificationInfo.soundName = @"Party.aiff";
notificationInfo.shouldBadge = YES;

subscription.notificationInfo = notificationInfo;
```

Subscriptions

Saving

```
CKSubscription *subscription = ...;

[[publicDatabase saveSubscription:subscription
   completionHandler:^(CKSubscription *subscription, NSError *error) {

     // labor-of-love error handling when (error != nil)
}];
```

Subscriptions

Handling push

Subscriptions

Handling push

```
@implementation AppDelegate

 - (void)application:(UIApplication *)application
                     didReceiveRemoteNotification:(NSDictionary *)userInfo {
```

Subscriptions

Handling push

```
@implementation AppDelegate

 - (void)application:(UIApplication *)application
                    didReceiveRemoteNotification:(NSDictionary *)userInfo {

      CKNotification *cloudKitNotification = [CKNotification
                    notificationFromRemoteNotificationDictionary:userInfo];
```

Subscriptions

Handling push

```
@implementation AppDelegate

 - (void)application:(UIApplication *)application
                     didReceiveRemoteNotification:(NSDictionary *)userInfo {

     CKNotification *cloudKitNotification = [CKNotification
                     notificationFromRemoteNotificationDictionary:userInfo];

     NSString *alertBody = cloudKitNotification.alertBody;
```

Subscriptions

Handling push

```
@implementation AppDelegate

 - (void)application:(UIApplication *)application
                     didReceiveRemoteNotification:(NSDictionary *)userInfo {

     CKNotification *cloudKitNotification = [CKNotification
                     notificationFromRemoteNotificationDictionary:userInfo];

     NSString *alertBody = cloudKitNotification.alertBody;

     if (cloudKitNotification.notificationType == CKNotificationTypeQuery) {
         CKQueryNotification *queryNotification = cloudKitNotification;
         CKRecordID *recordID = [queryNotification recordID];
    }
 }
```

CloudKit User Accounts

CloudKit User Accounts

CloudKit User Accounts

Identity

CloudKit User Accounts

- Identity
- Metadata

CloudKit User Accounts

Identity

Metadata

Privacy

CloudKit User Accounts

Identity

Metadata

Privacy

Discovery

CloudKit User Accounts

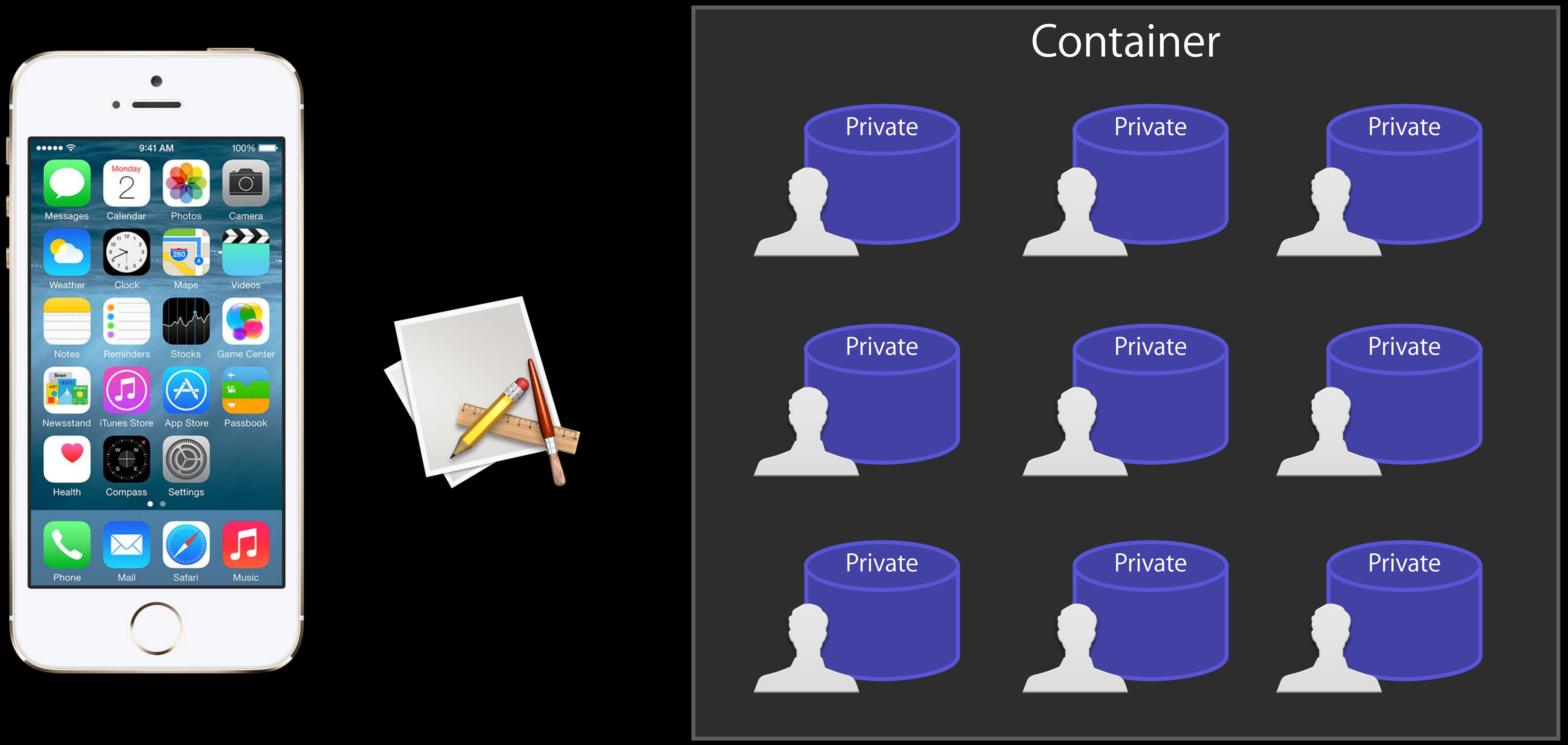
Identity

Metadata

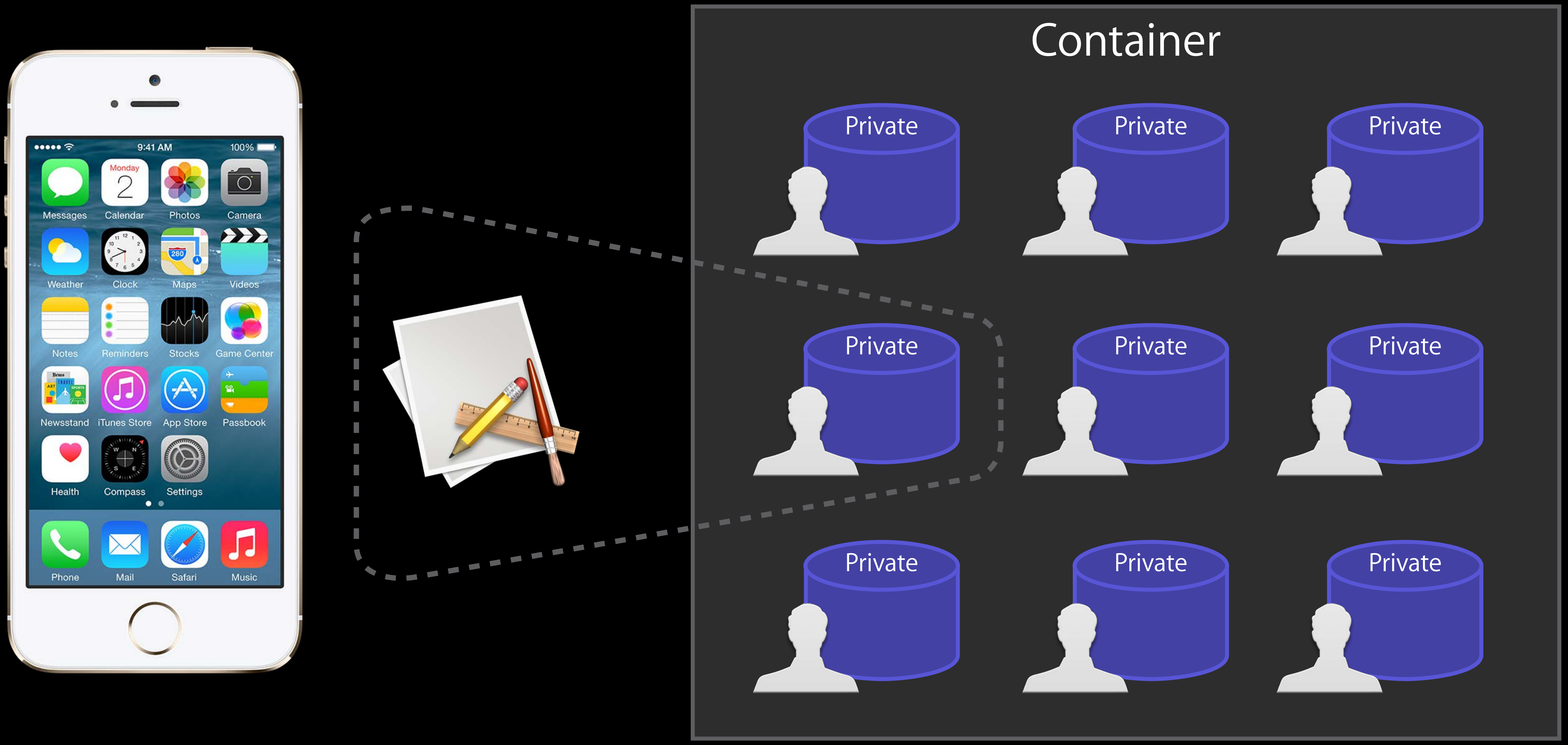
Privacy

Discovery

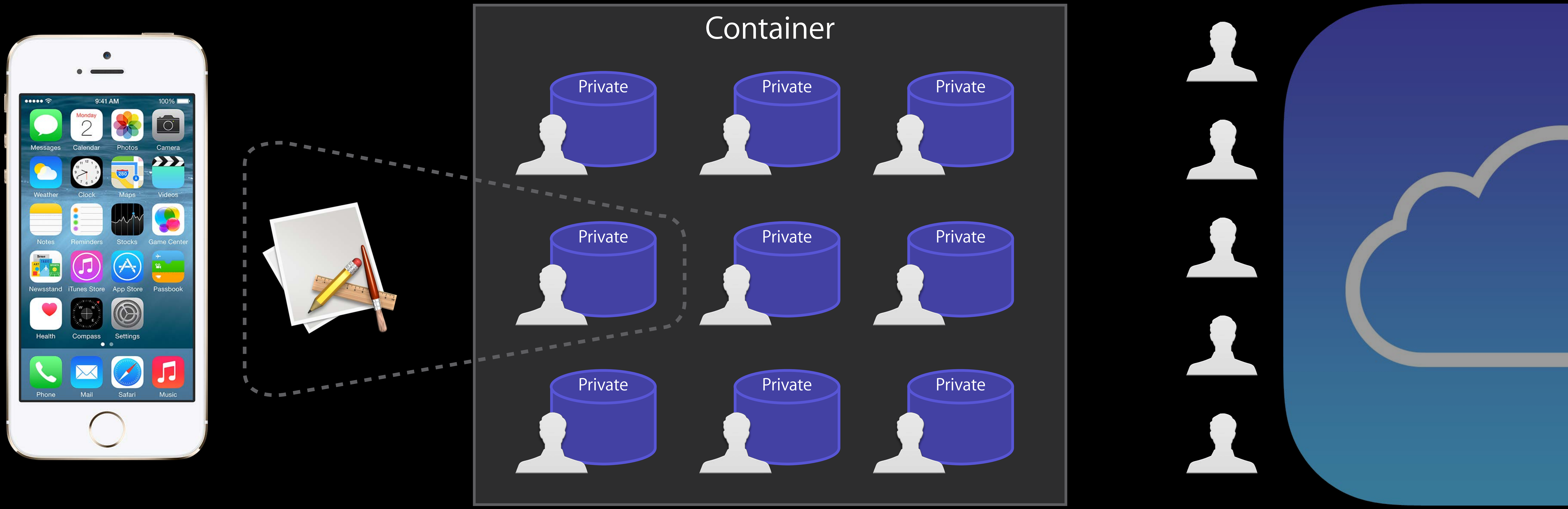
User Identity



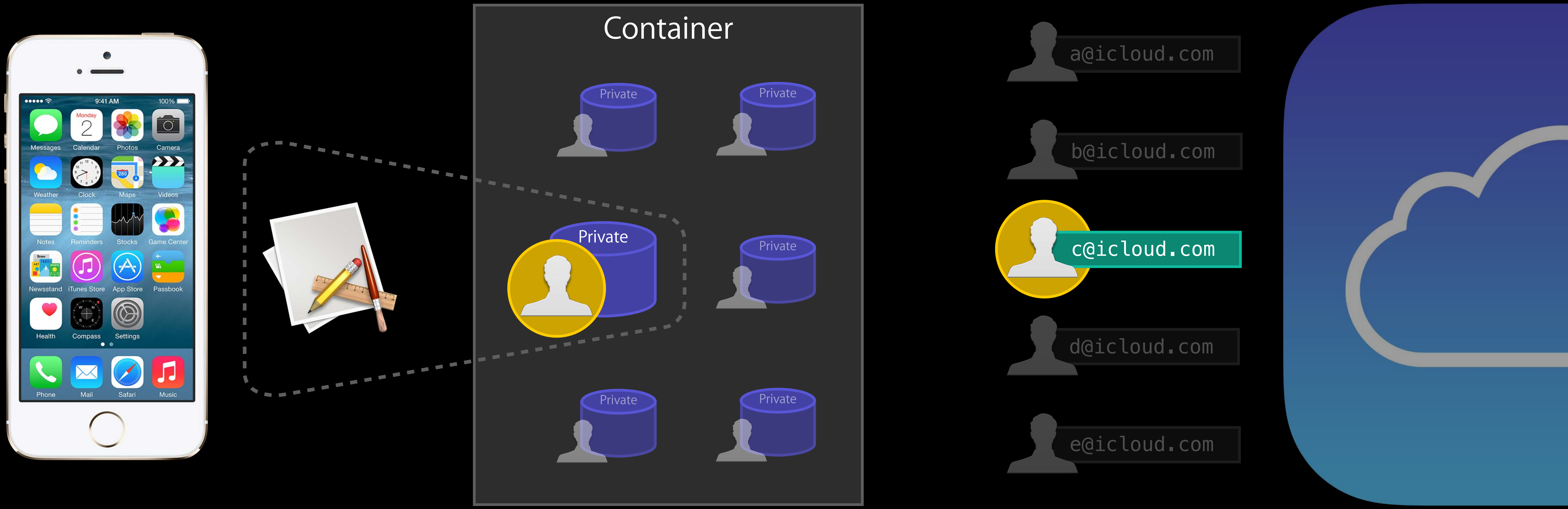
User Identity



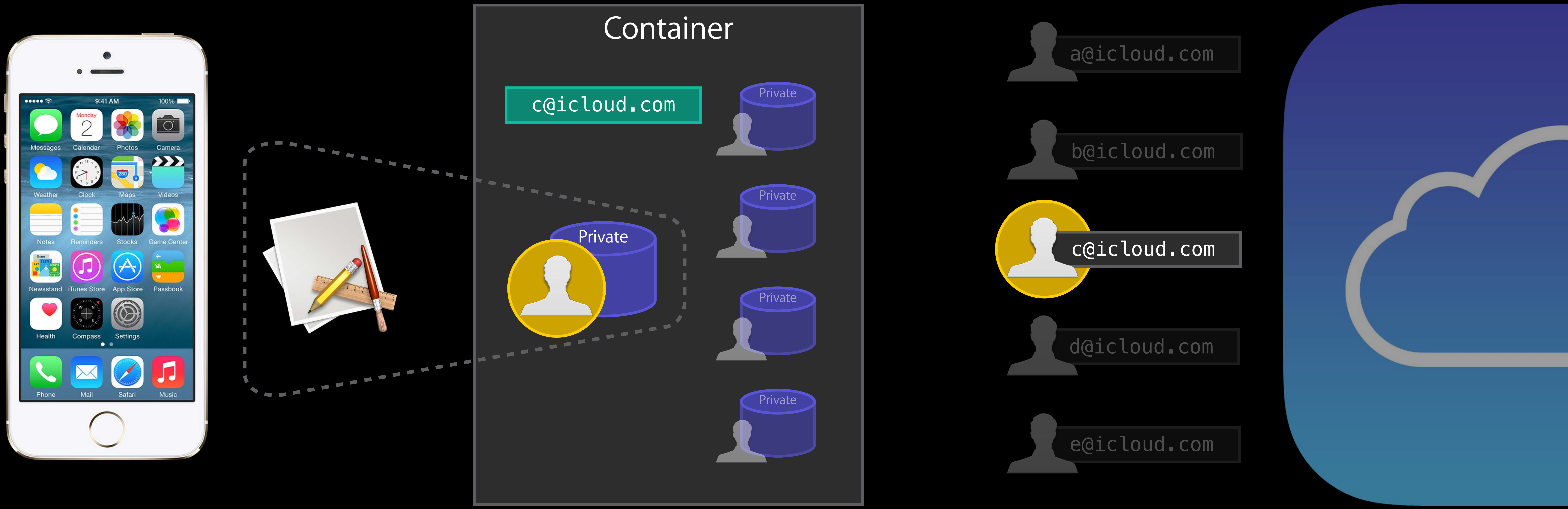
User Identity



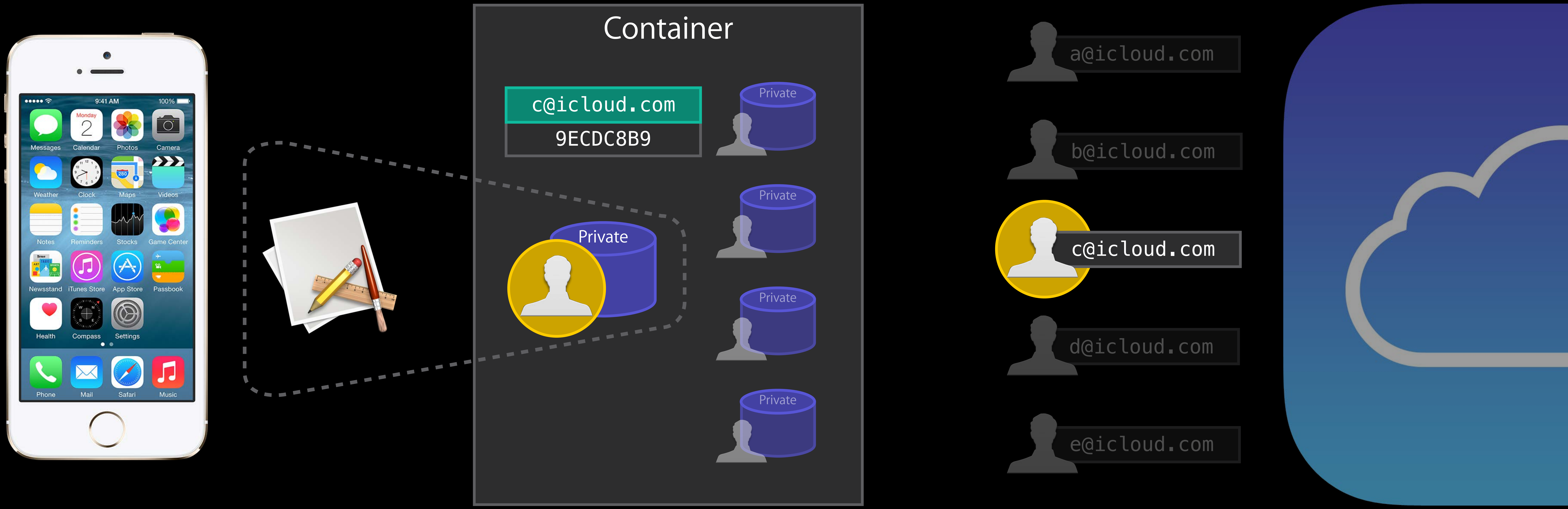
User Identity



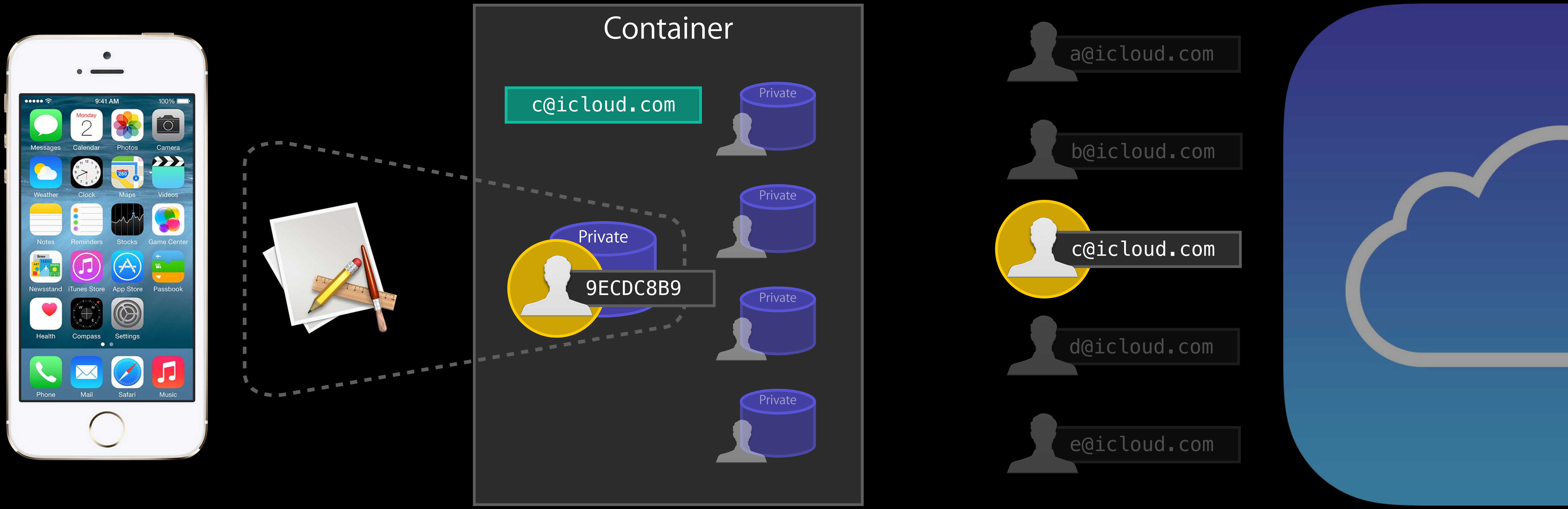
User Identity



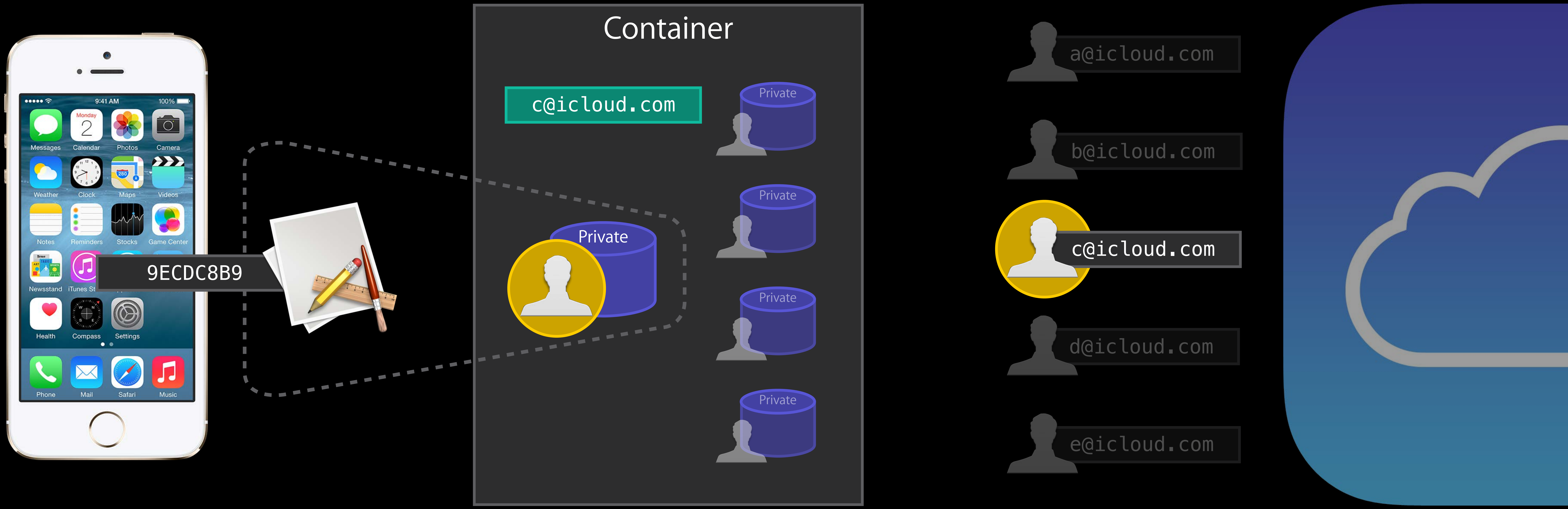
User Identity



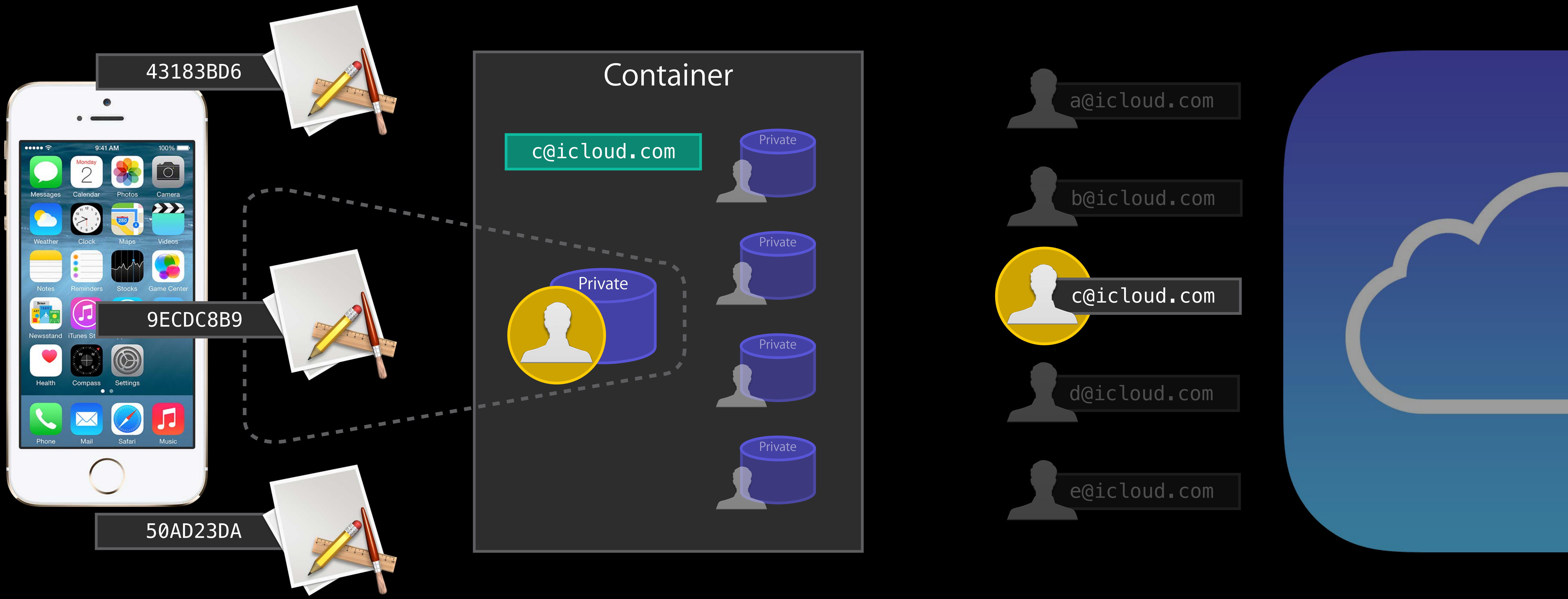
User Identity



User Identity



User Identity



User Identity

User Identity

User Record ID

User Identity

User Record ID

Stable identifier for this user

User Identity

User Record ID

Stable identifier for this user

Scoped to the container

User Identity

User Record ID

Stable identifier for this user

Scoped to the container

Independent API

User Identity

```
 [[CKContainer defaultContainer] fetchUserRecordIDWithCompletionHandler:
          ^(CKRecordID *userRecordID, NSError *error) {

     // ostentatious error handling when (error != nil)
}];
```

CloudKit User Accounts

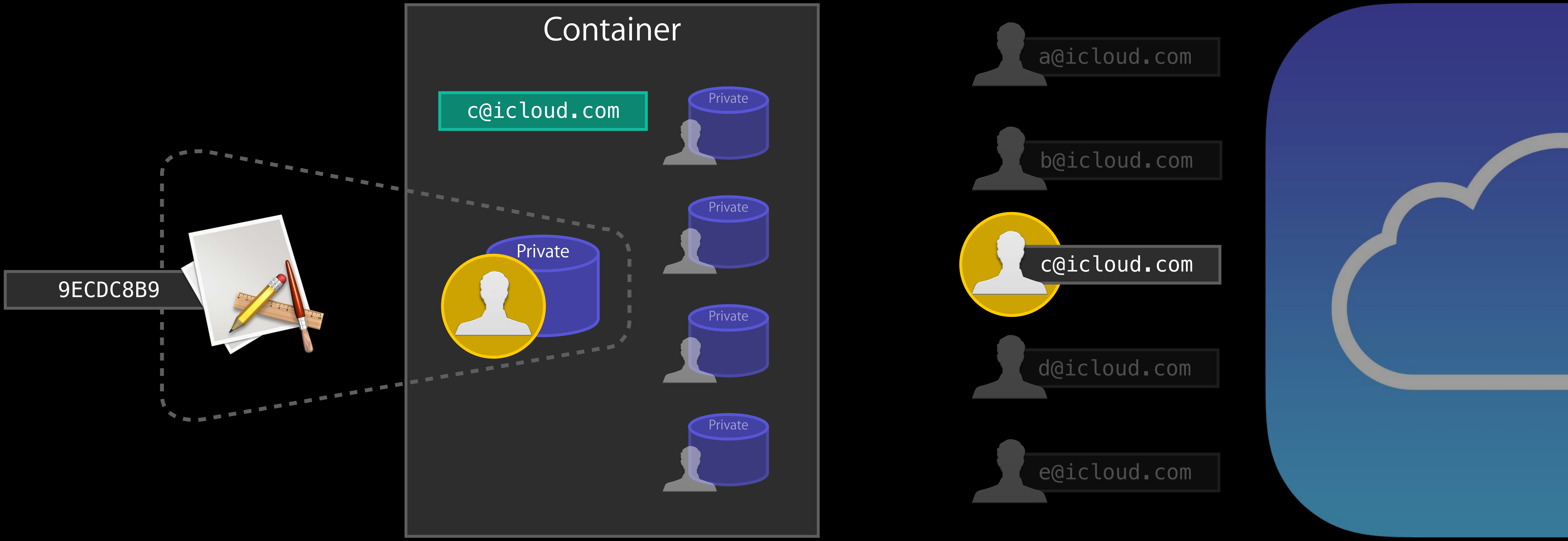
Identity

Metadata

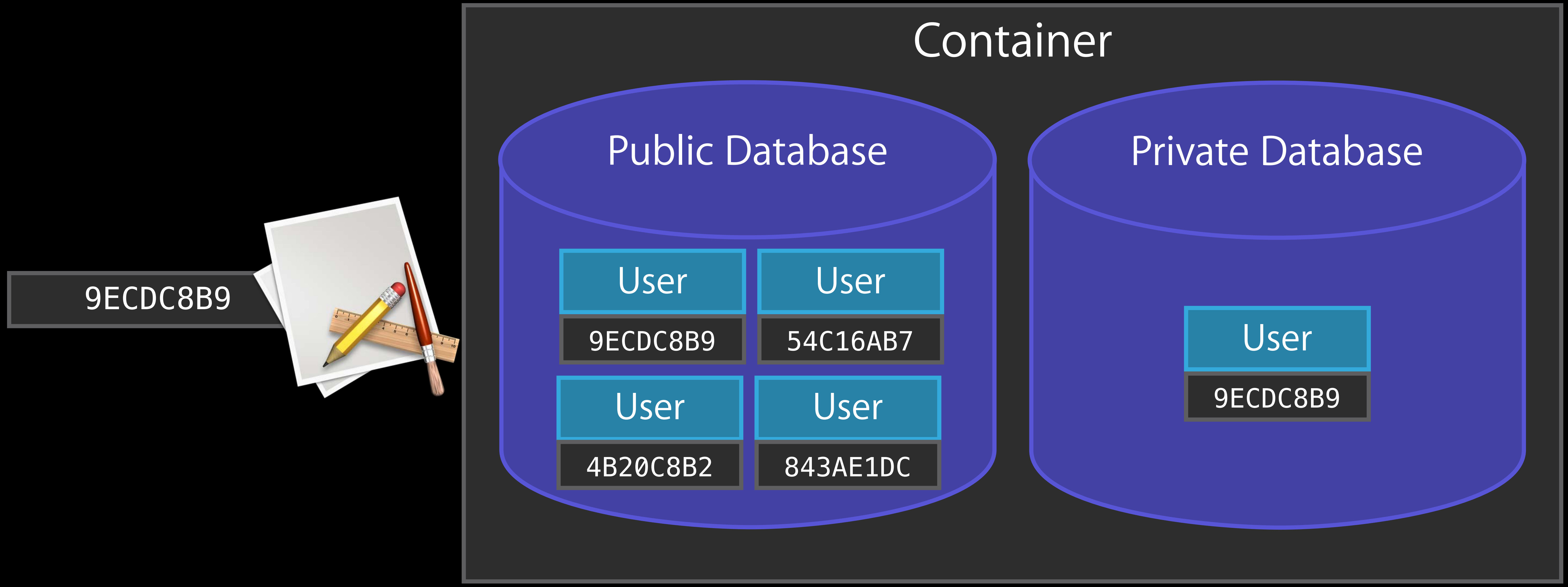
Privacy

Discovery

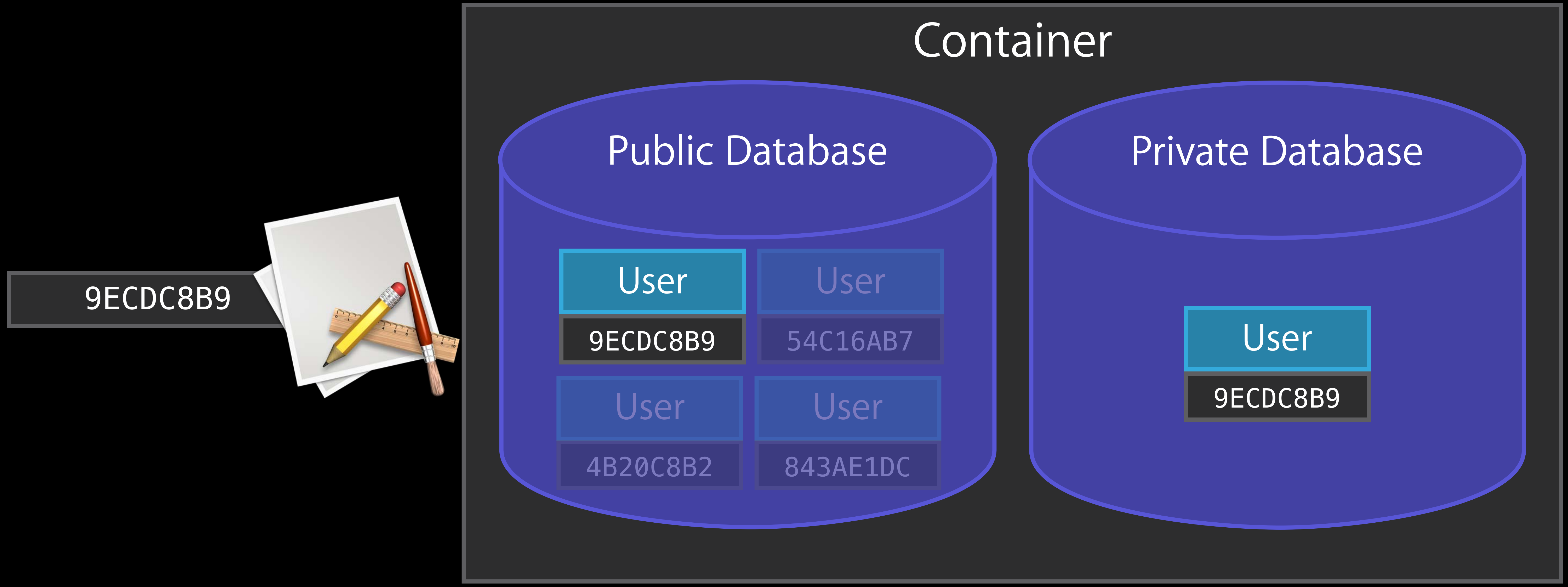
User Metadata



User Metadata

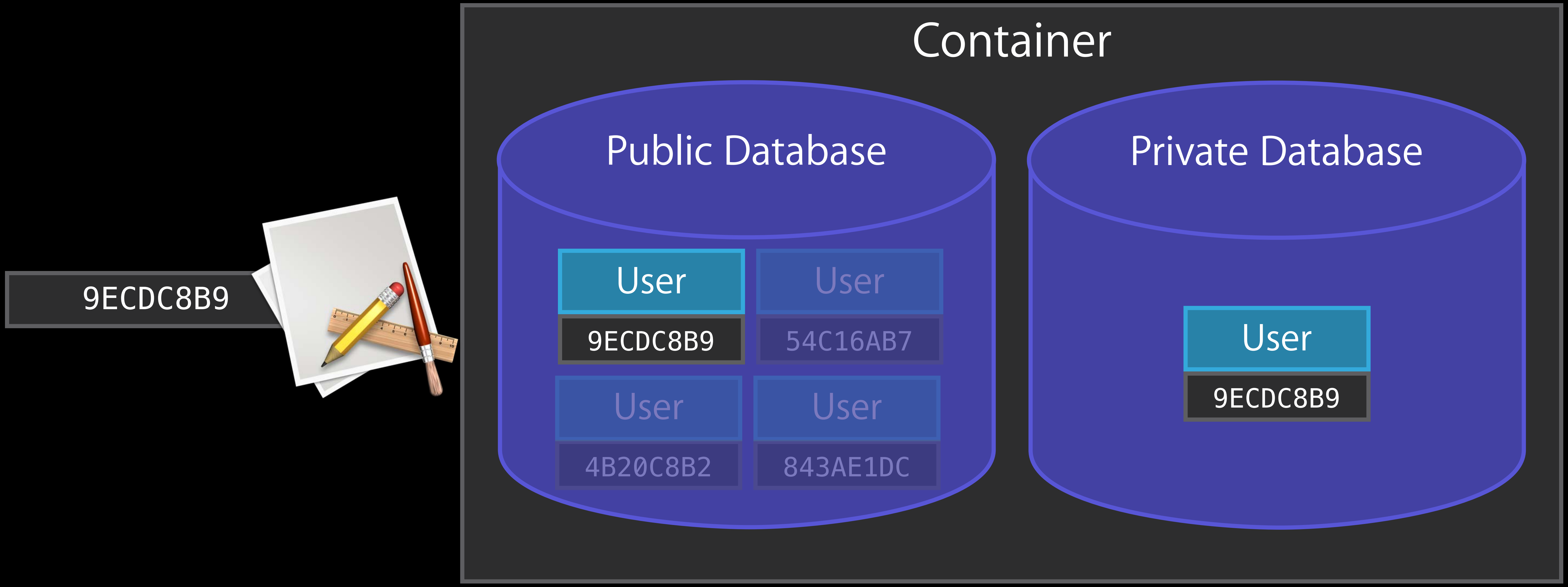


User Metadata



User Metadata

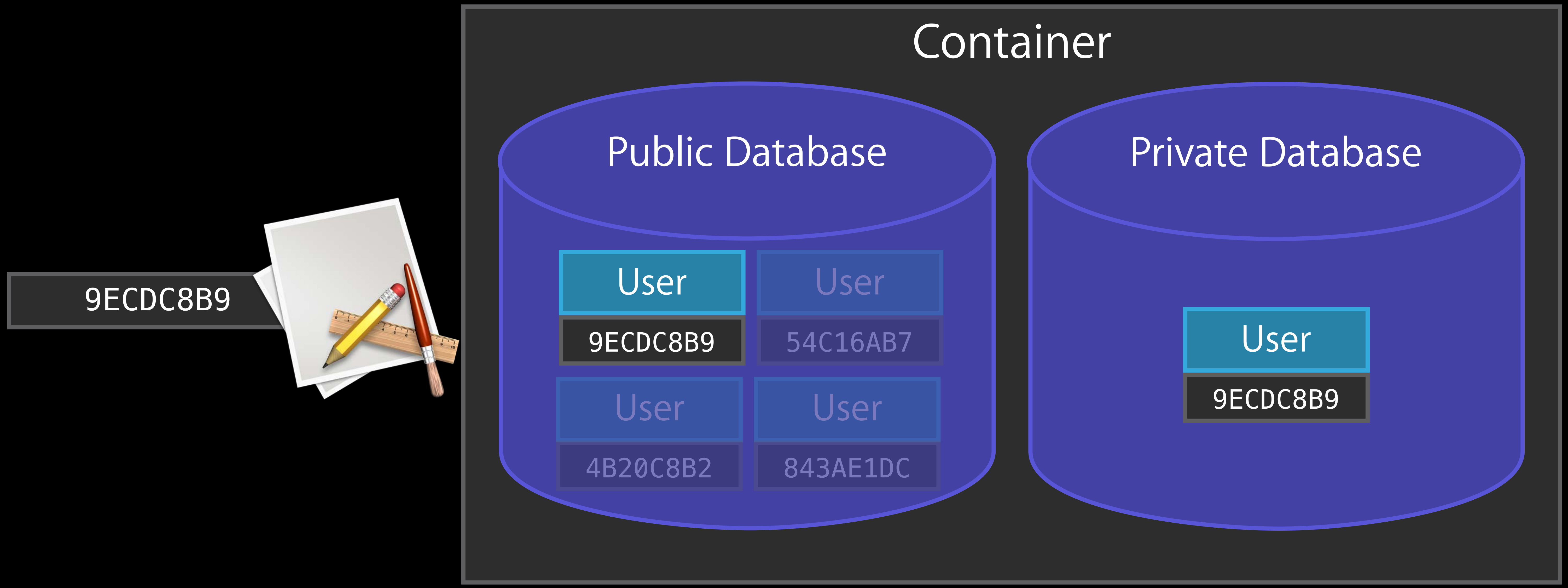
User Record



User Metadata

User Record

One per database

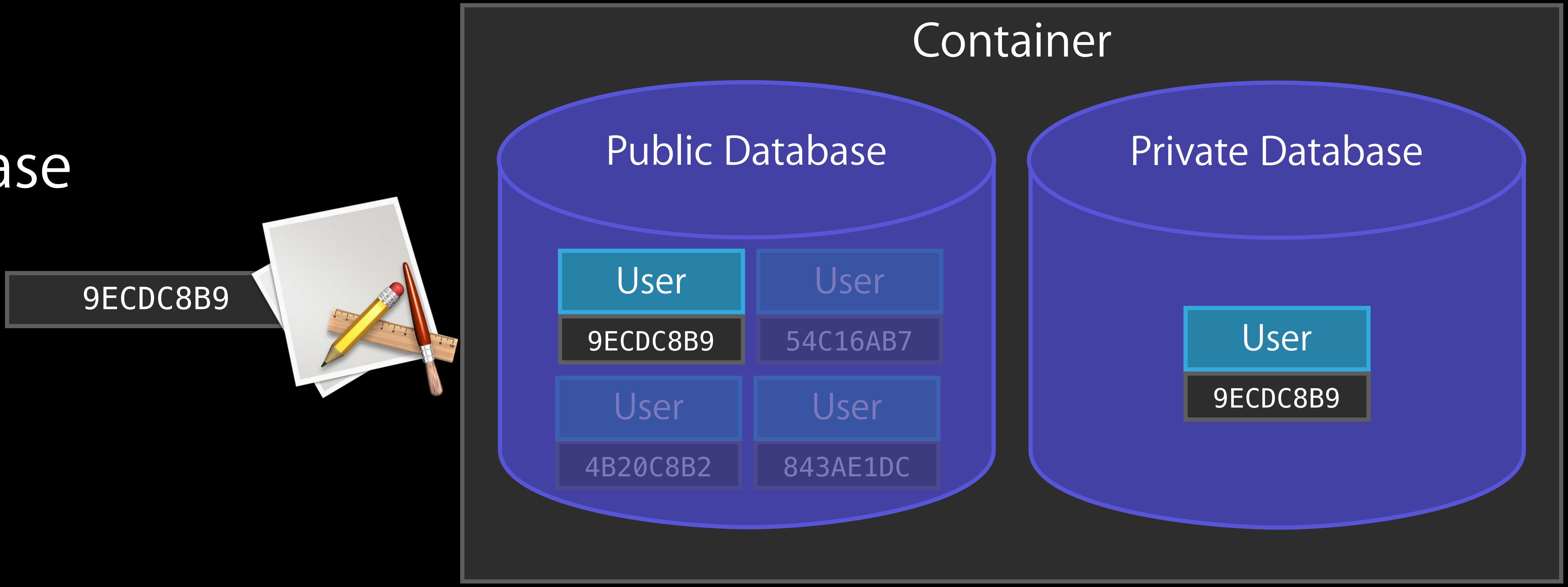


User Metadata

User Record

One per database

World readable in public database



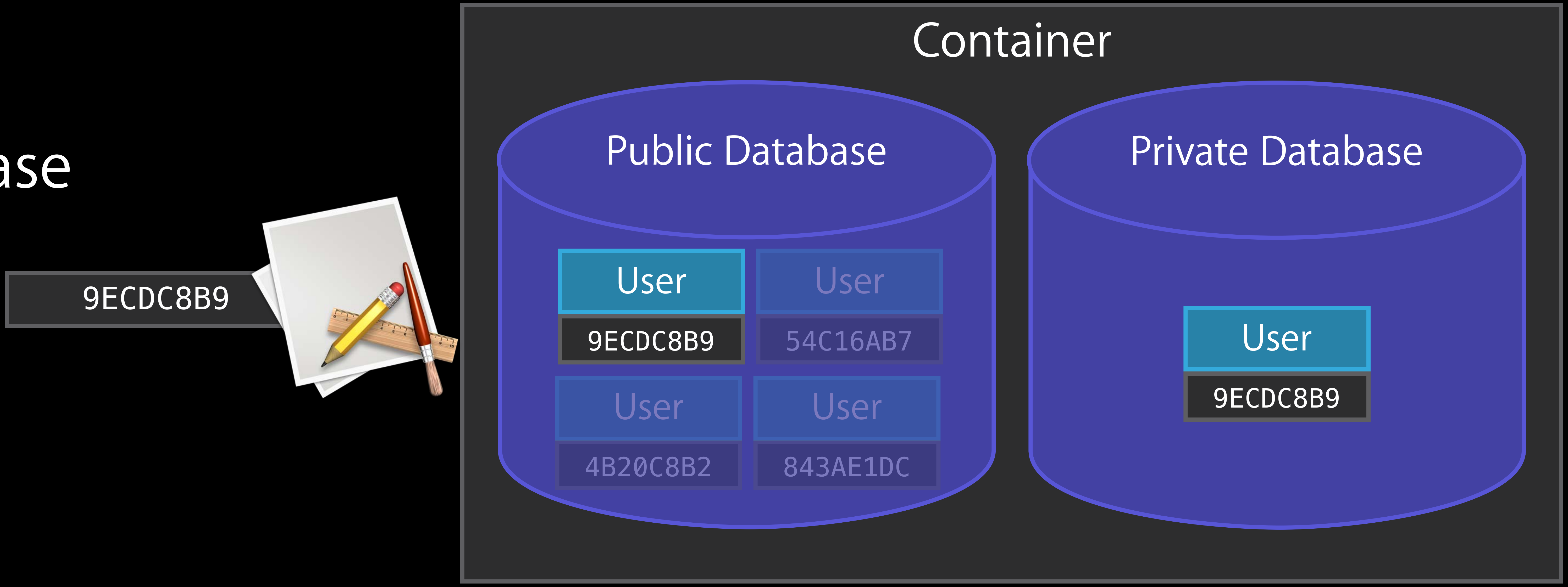
User Metadata

User Record

One per database

World readable in public database

Treated like ordinary record



User Metadata

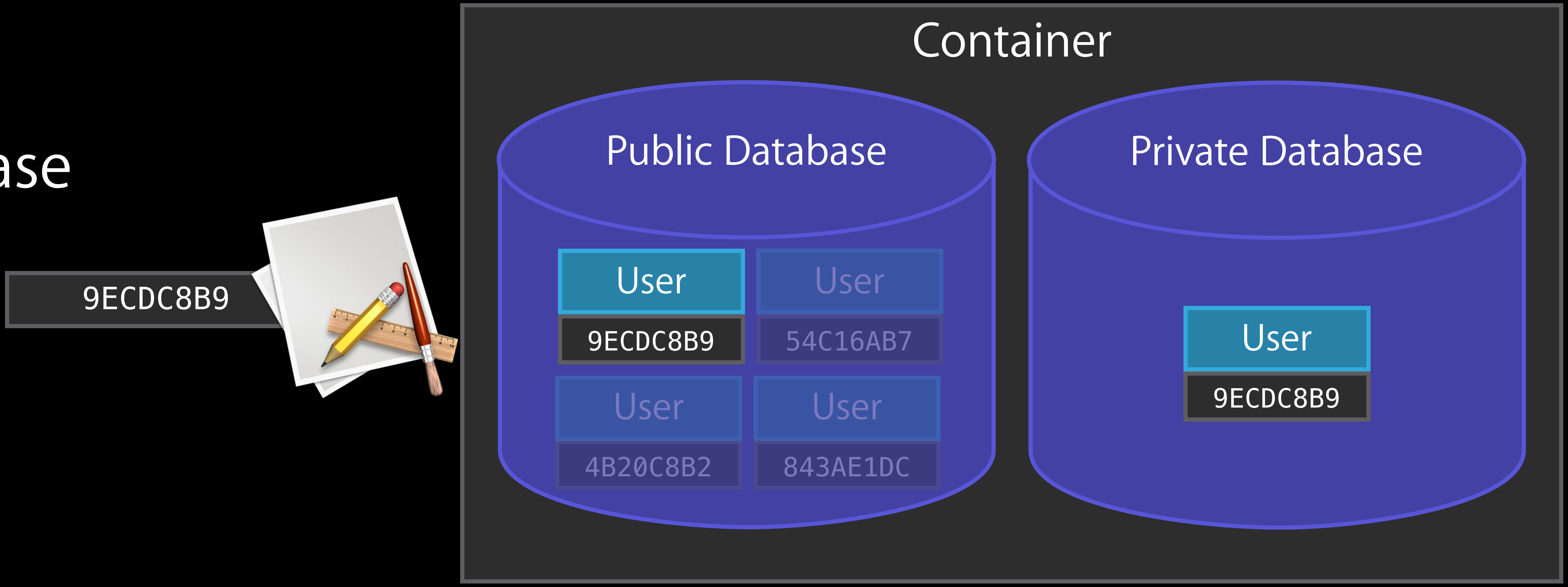
User Record

One per database

World readable in public database

Treated like ordinary record

- `CKRecordTypeUserRecord`



User Metadata

User Record

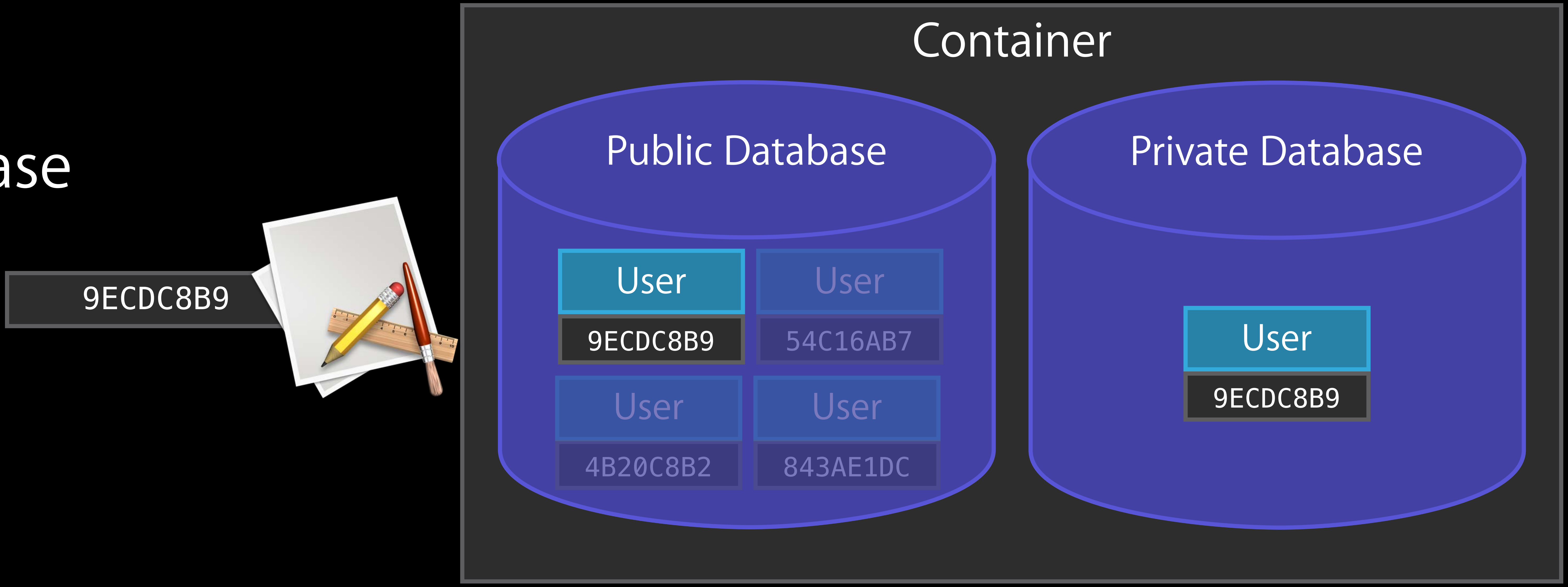
One per database

World readable in public database

Treated like ordinary record

- `CKRecordTypeUserRecord`

… mostly



User Metadata

User Record

One per database

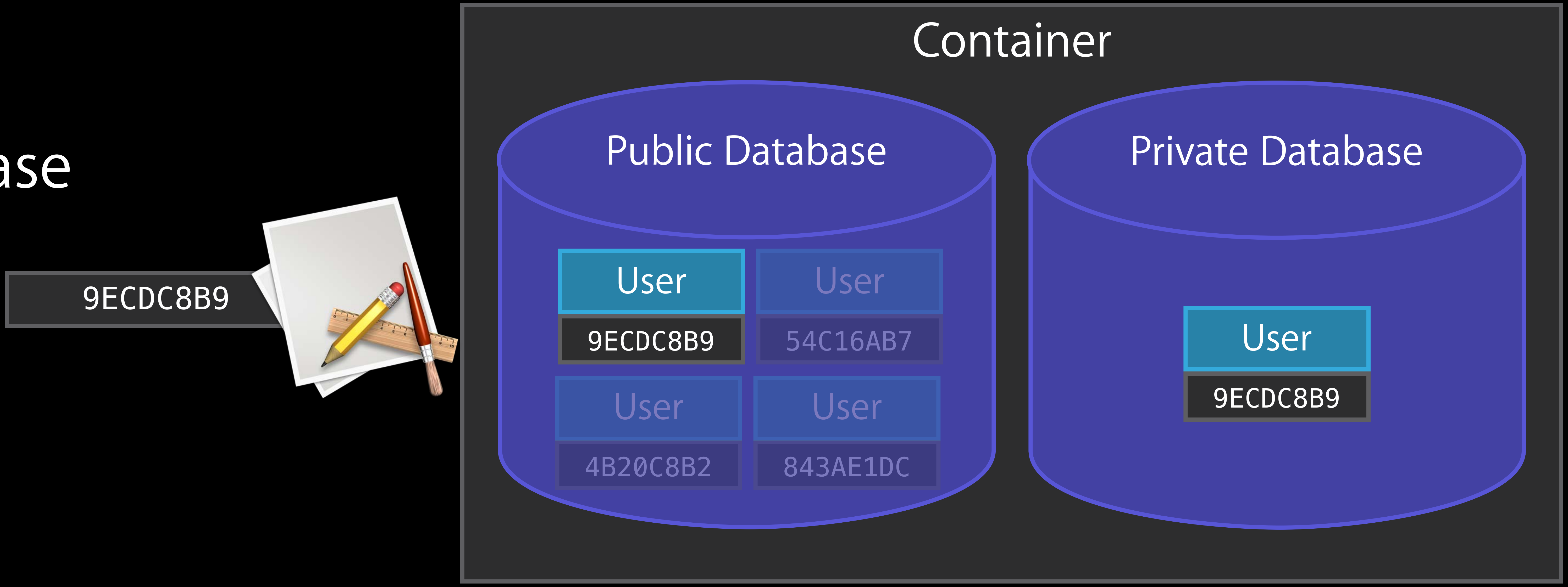
World readable in public database

Treated like ordinary record

- `CKRecordTypeUserRecord`

… mostly

- Reserved by system



User Metadata

User Record

One per database

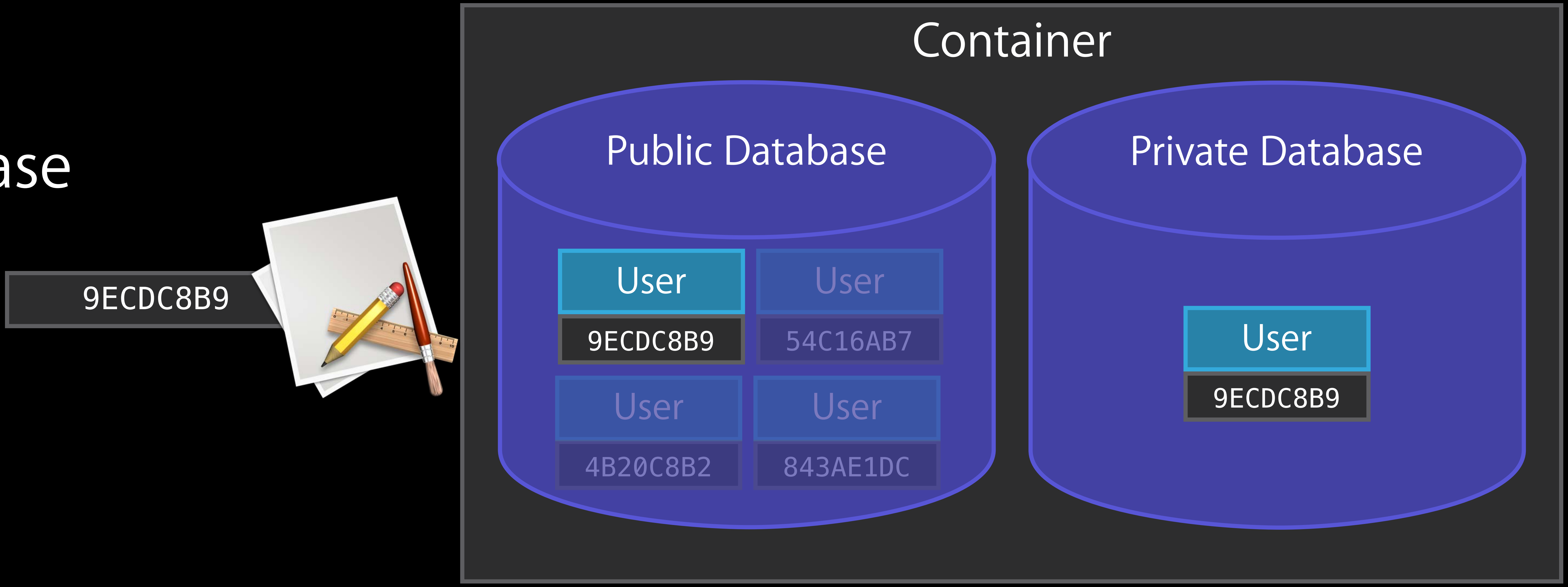
World readable in public database

Treated like ordinary record

- `CKRecordTypeUserRecord`

… mostly

- Reserved by system
- Cannot be queried



User Metadata

```
CKContainer *defaultContainer =[CKContainer defaultContainer];
CKDatabase *publicDatabase = [defaultContainer publicCloudDatabase];

[defaultContainer fetchUserRecordIDWithCompletionHandler:
         ^(CKRecordID *userRecordID, NSError *error) {
    if (error) { ... } else {
```

User Metadata

```
CKContainer *defaultContainer =[CKContainer defaultContainer];
CKDatabase *publicDatabase = [defaultContainer publicCloudDatabase];

[defaultContainer fetchUserRecordIDWithCompletionHandler:
         ^(CKRecordID *userRecordID, NSError *error) {
    if (error) { ... } else {

        [publicDatabase fetchRecordWithID:userRecordID
              completionHandler:^(CKRecord *userRecord, NSError *error) {
```

User Metadata

```
CKContainer *defaultContainer =[CKContainer defaultContainer];
CKDatabase *publicDatabase = [defaultContainer publicCloudDatabase];

[defaultContainer fetchUserRecordIDWithCompletionHandler:
         ^(CKRecordID *userRecordID, NSError *error) {
    if (error) { ... } else {

        [publicDatabase fetchRecordWithID:userRecordID
              completionHandler:^(CKRecord *userRecord, NSError *error) {
            if (error) { ... } else {
                NSString *partyName = userRecord[@"partyName"];
                NSLog(@"Fetched record for %@:%@", partyName, userRecord);
            }
        }];
    }
}];
```

CloudKit User Accounts

- Identity
- Metadata
- Privacy
- Discovery

User Privacy

User Privacy

No disclosure by default

User Privacy

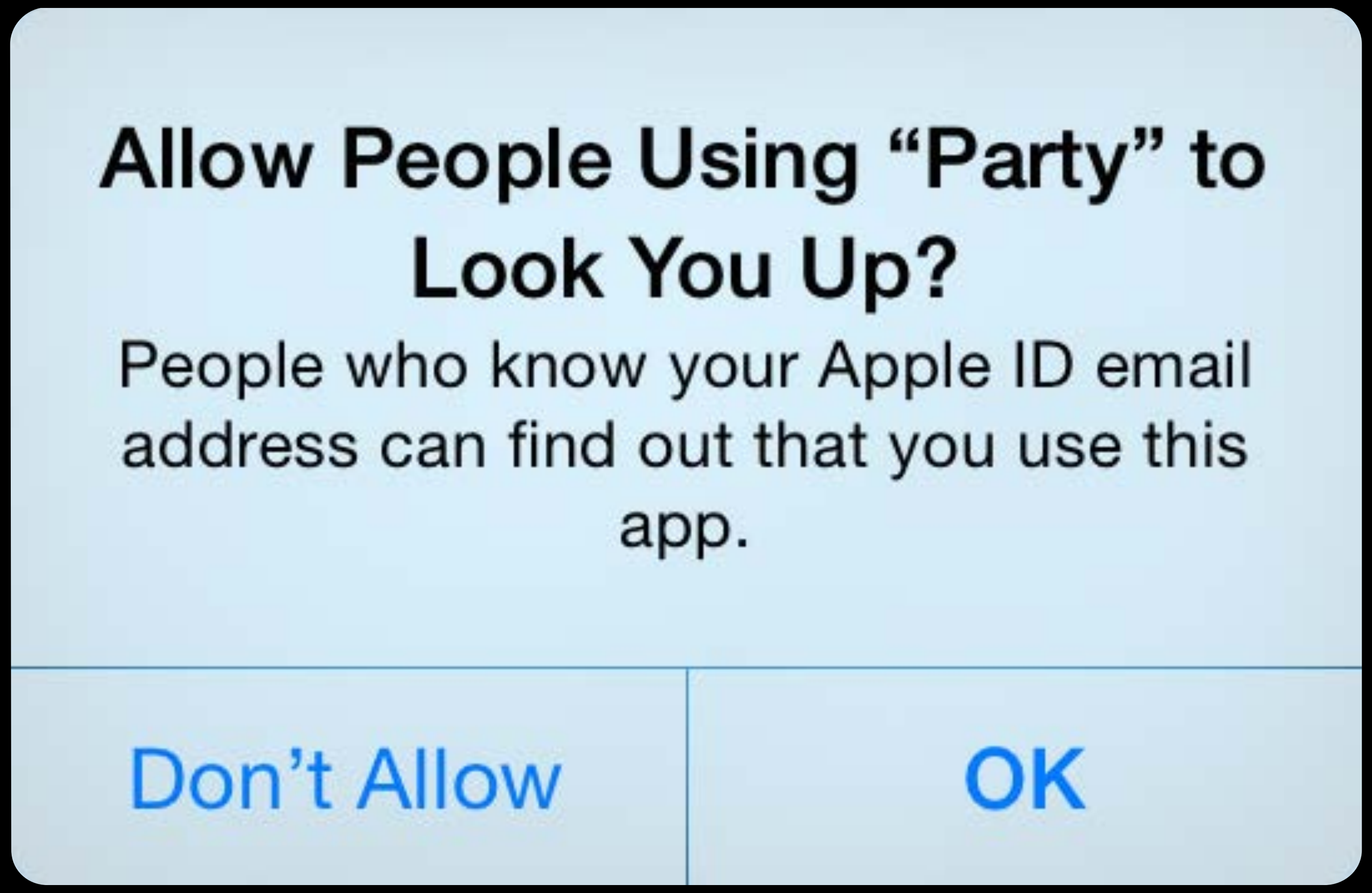
No disclosure by default

Disclosure requested by application

User Privacy

No disclosure by default

Disclosure requested by application



Allow People Using “Party” to Look You Up?

People who know your Apple ID email address can find out that you use this app.

CloudKit User Accounts

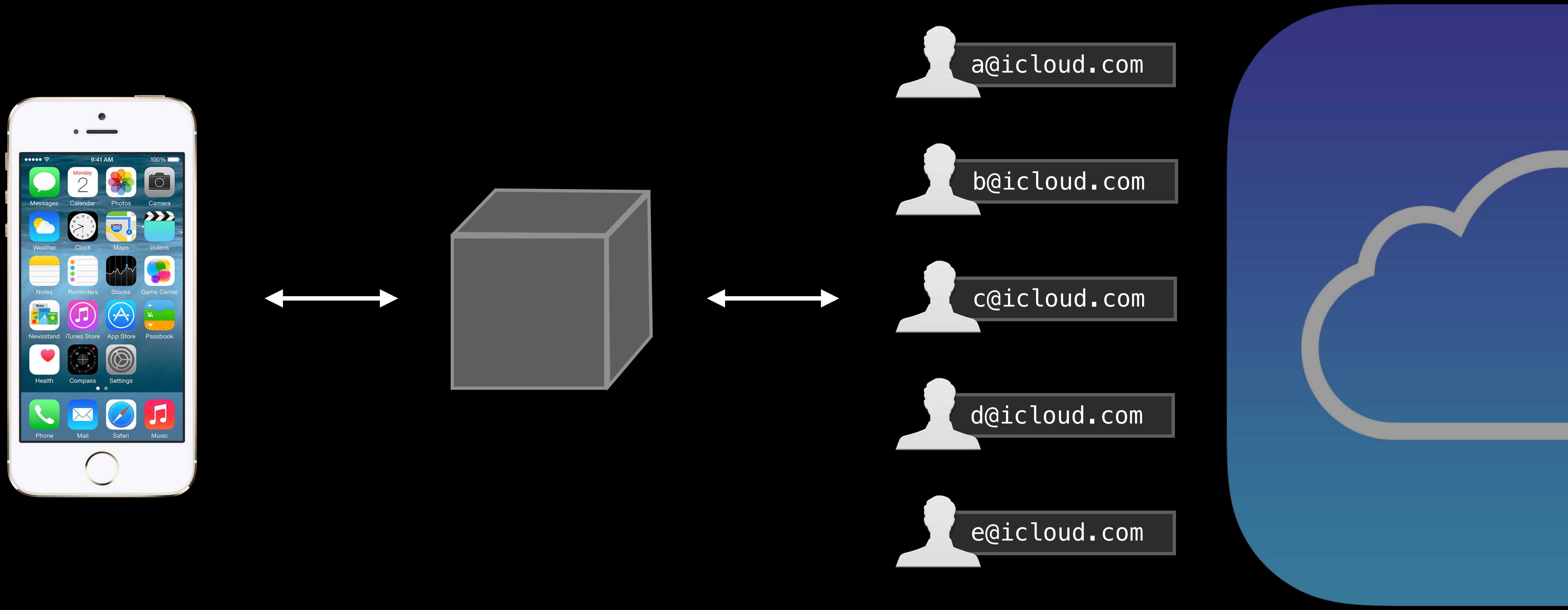
Identity

Metadata

Privacy

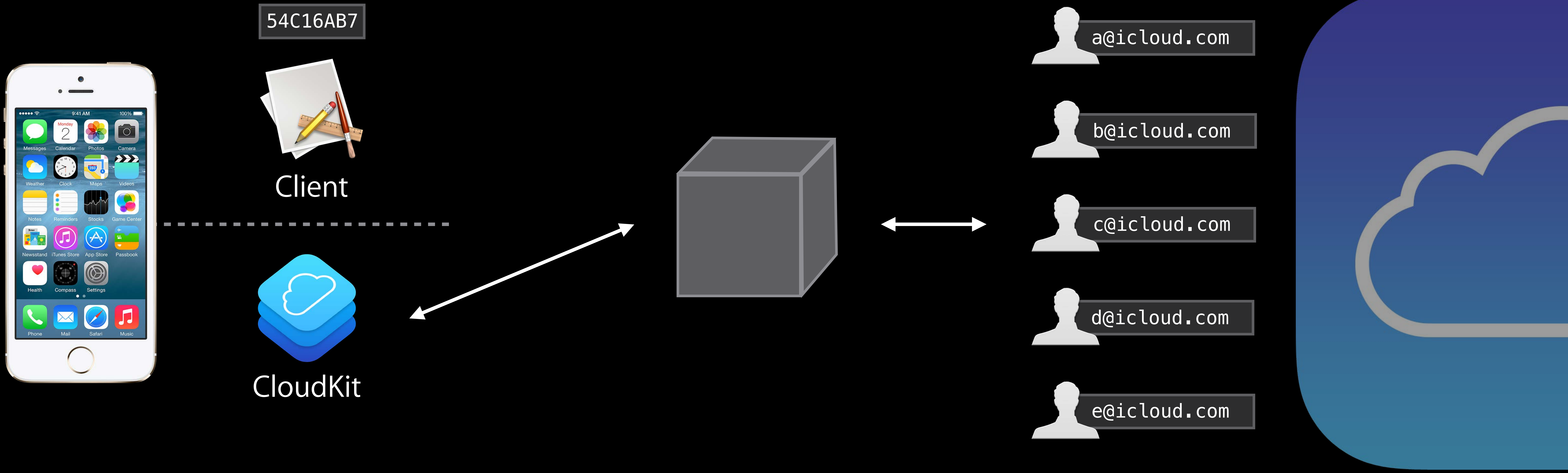
Discovery

User Discovery



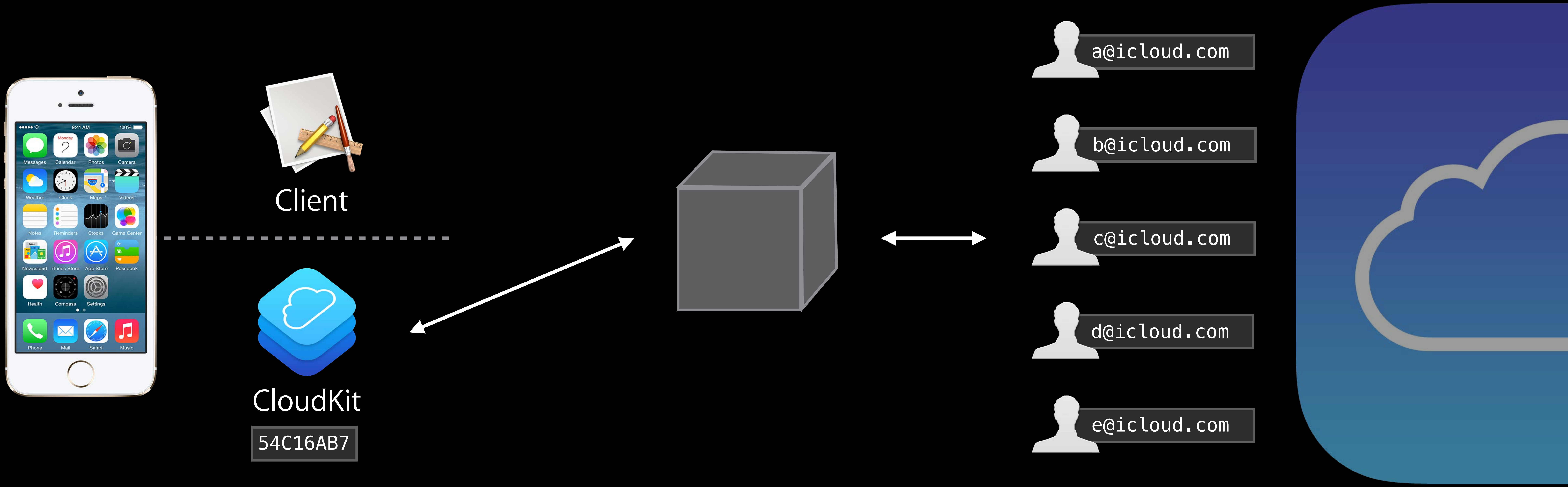
User Discovery

Record ID



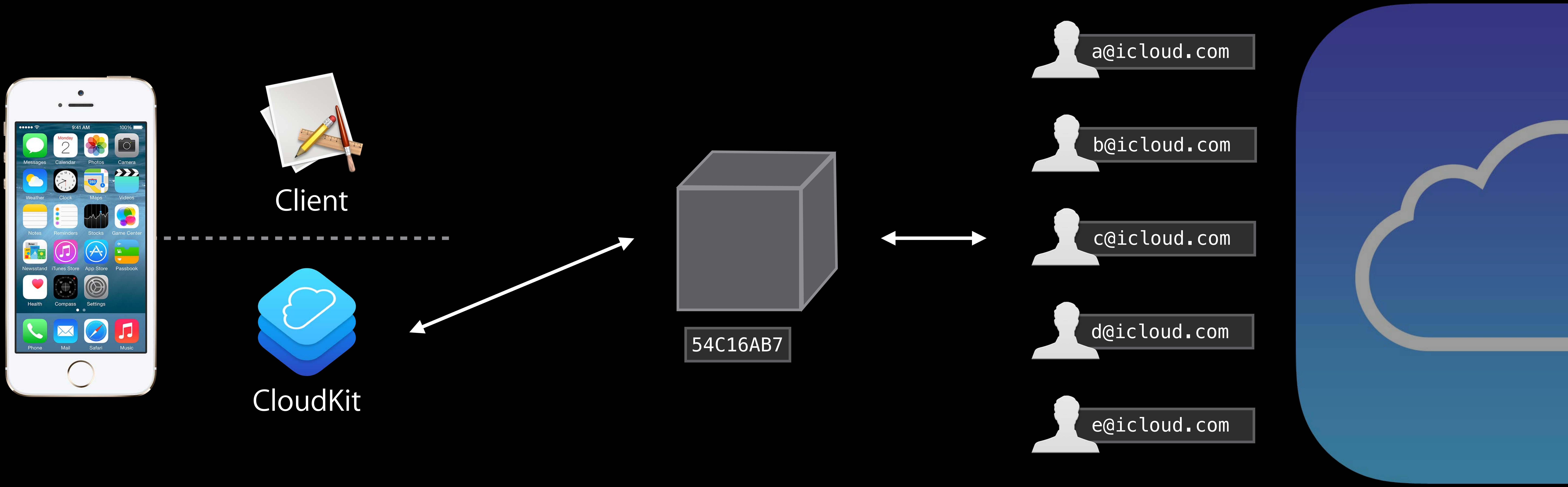
User Discovery

Record ID



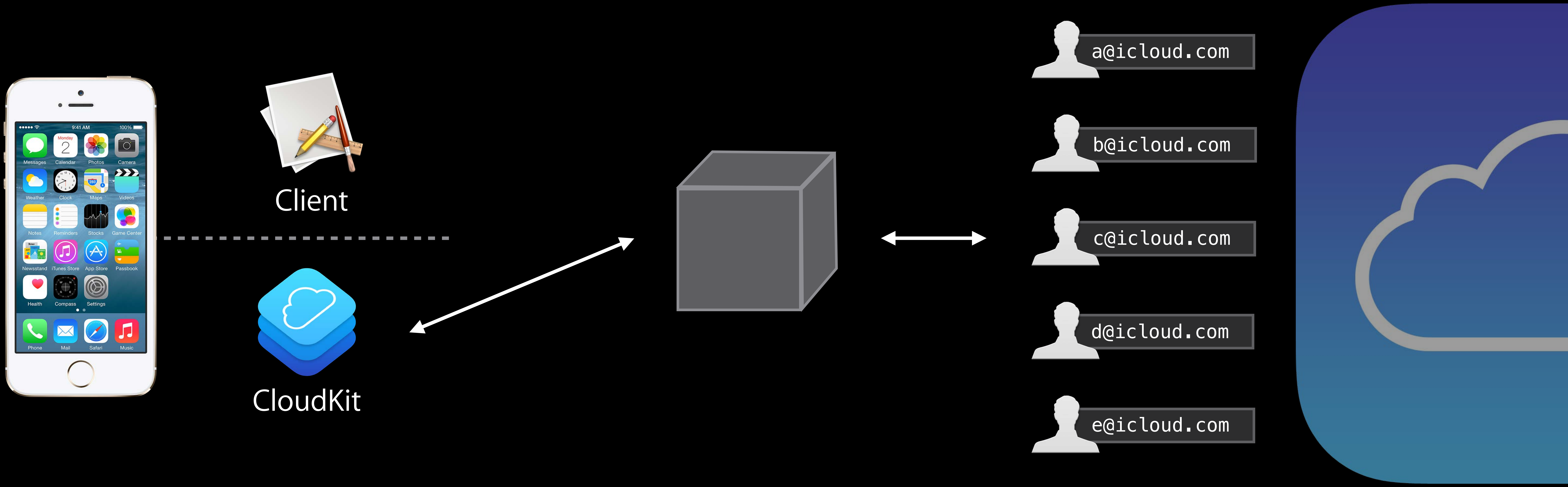
User Discovery

Record ID



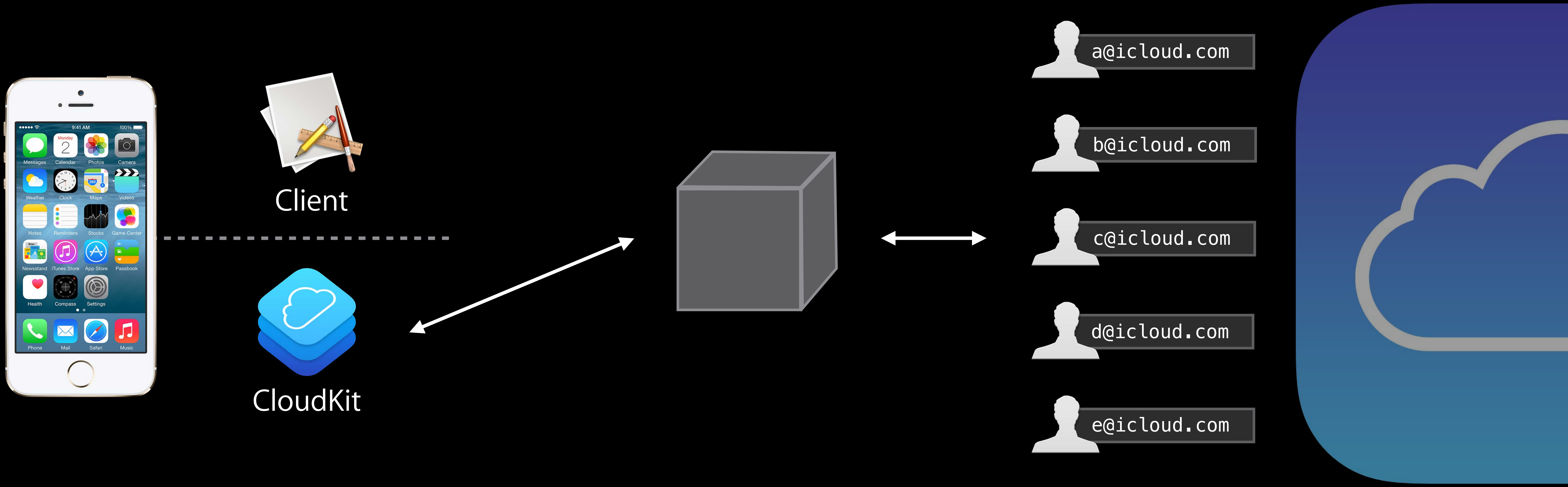
User Discovery

Record ID



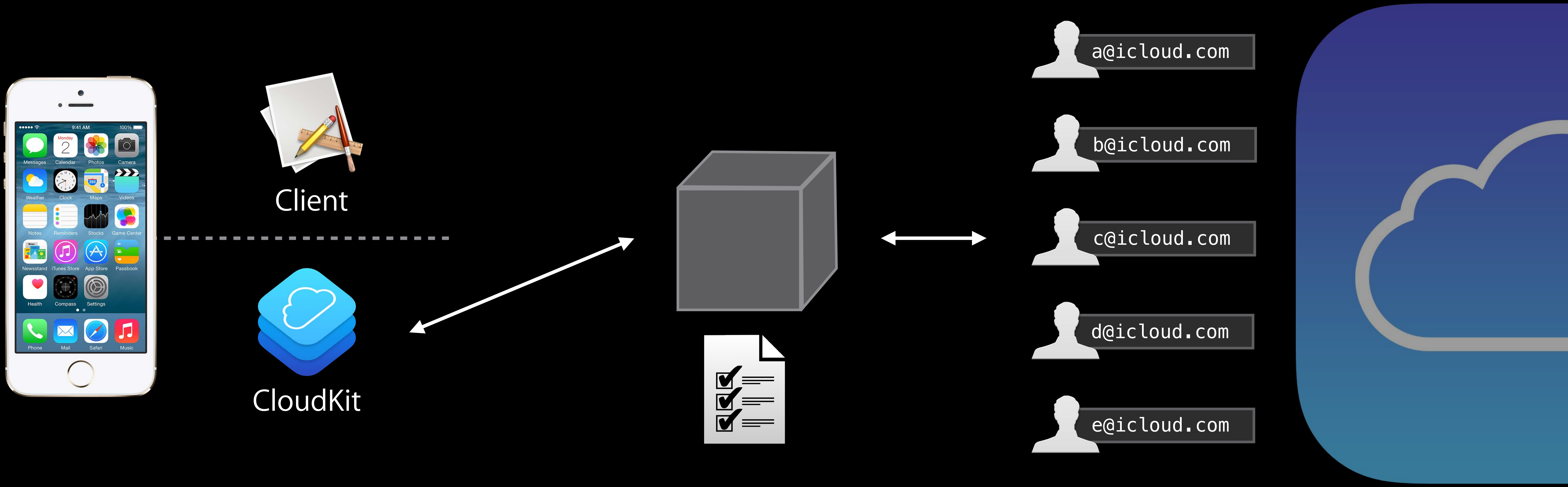
User Discovery

Record ID



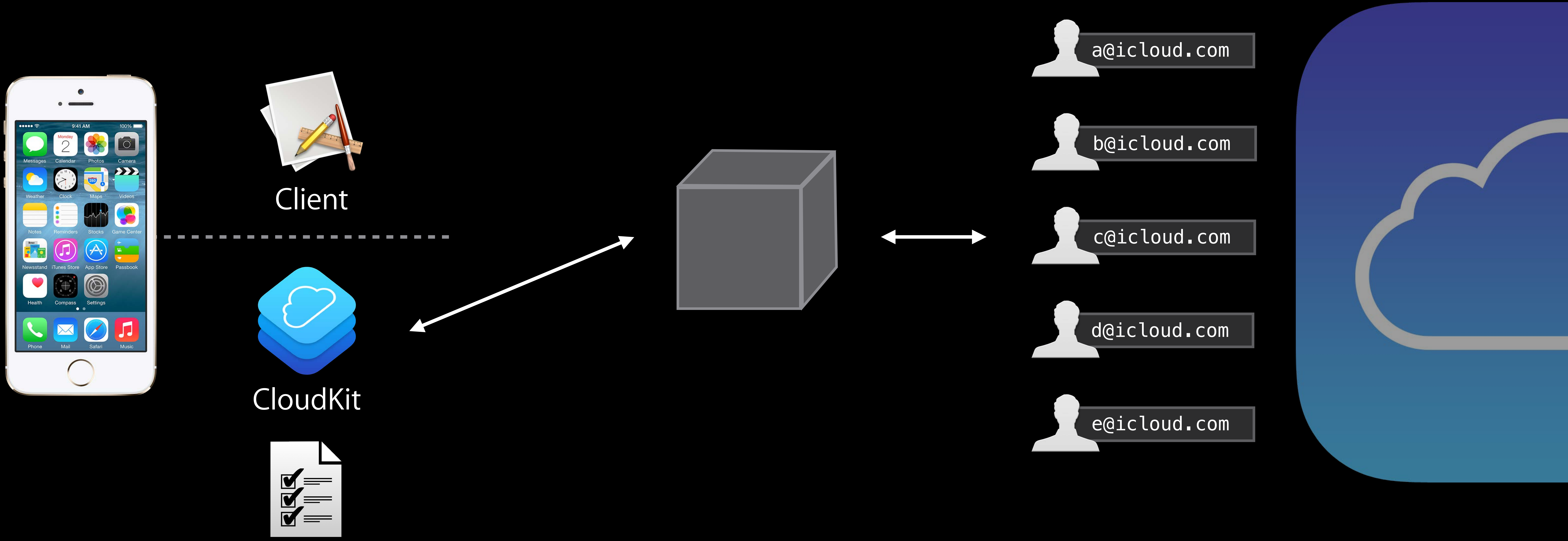
User Discovery

Record ID



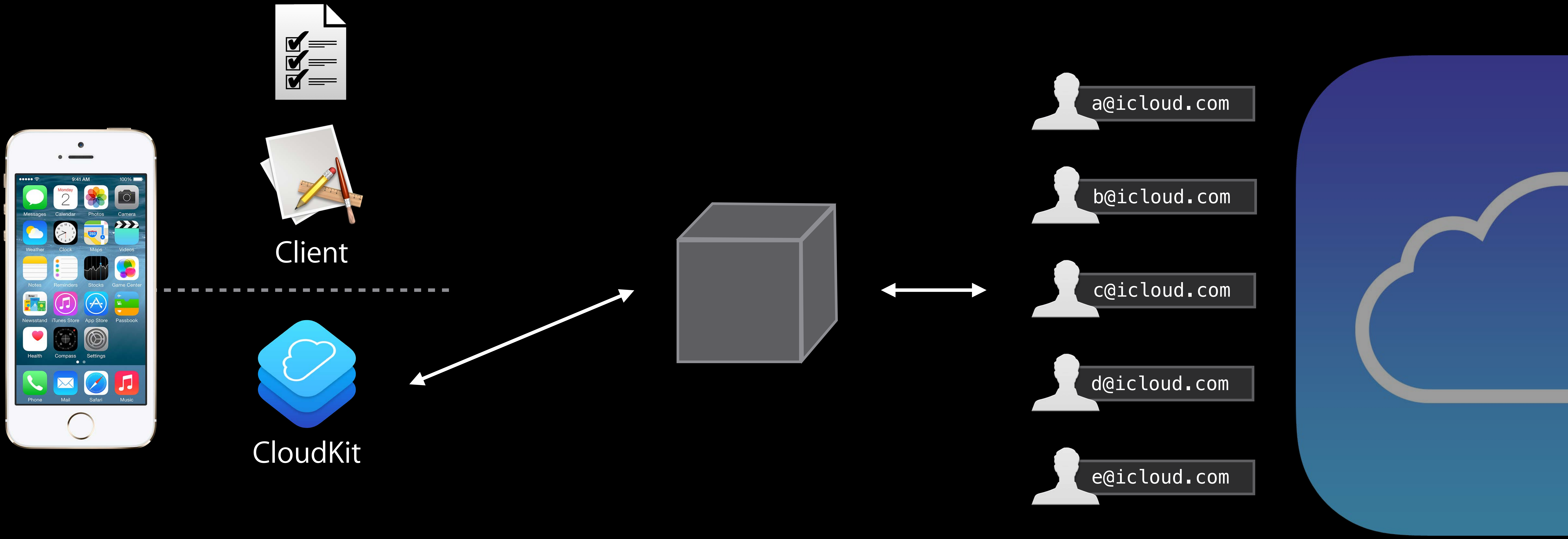
User Discovery

Record ID



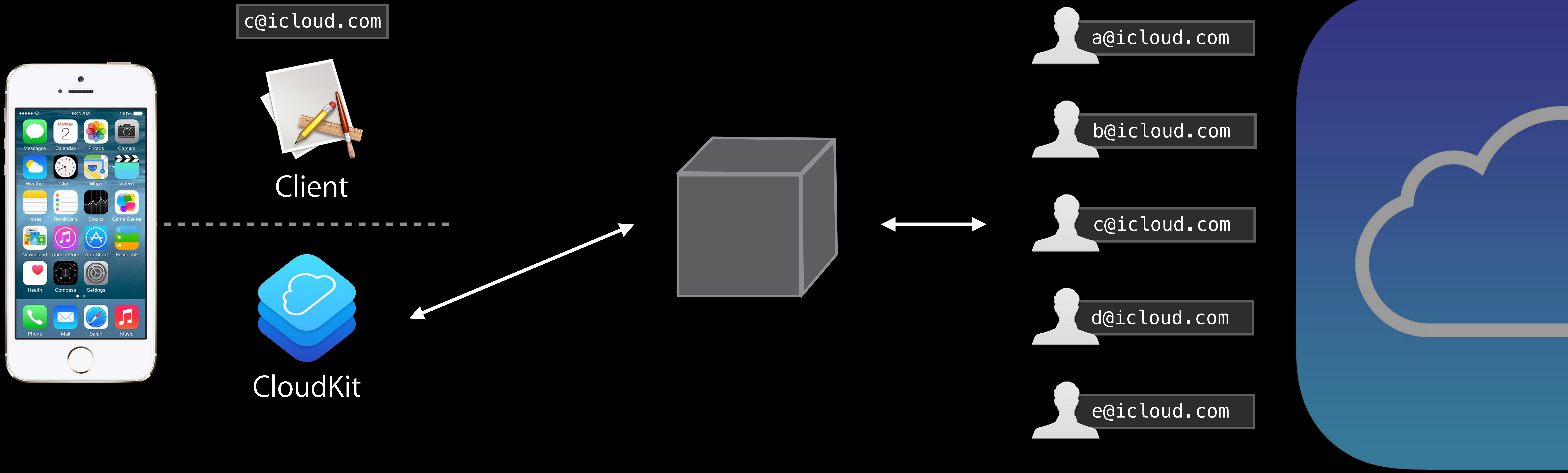
User Discovery

Record ID



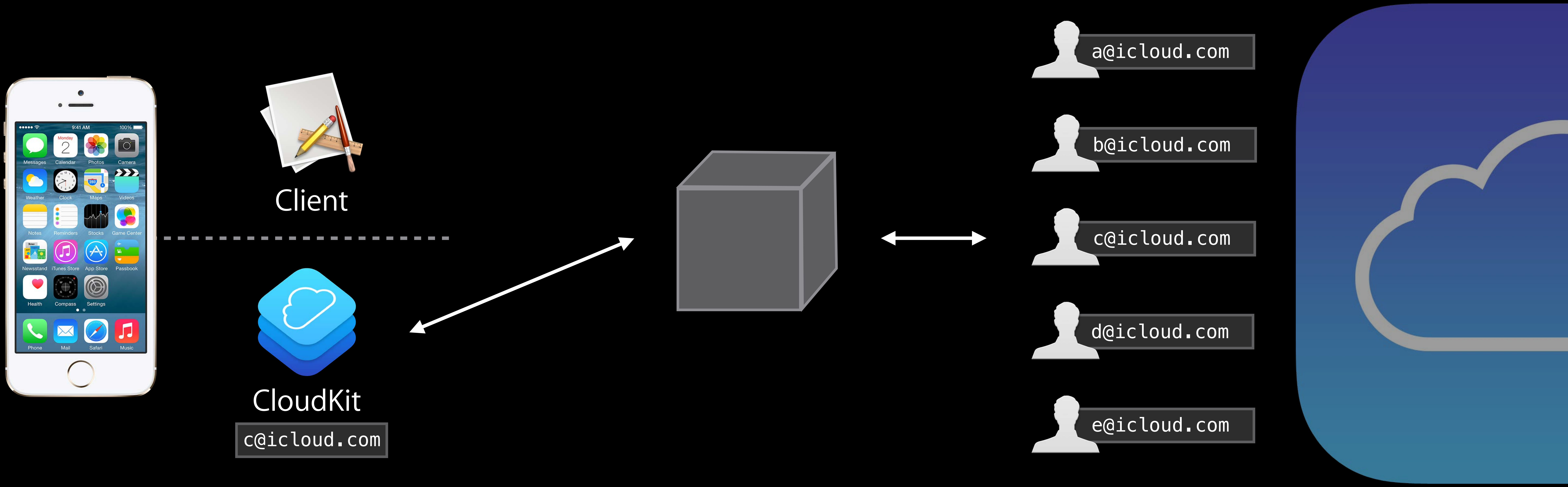
User Discovery

Email address



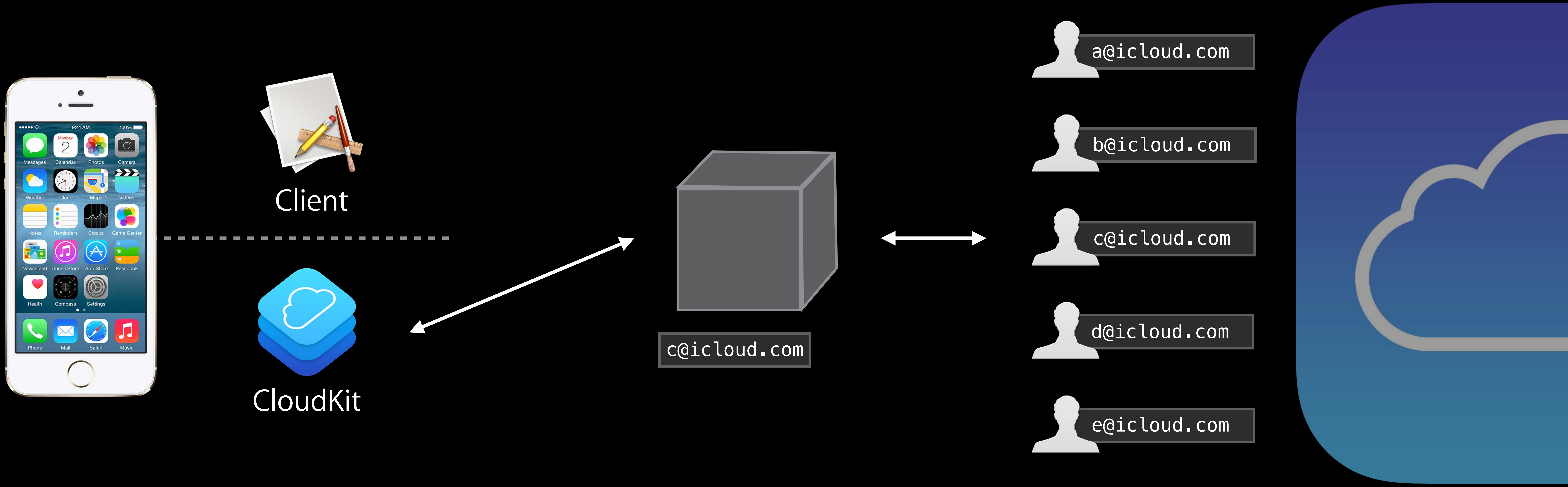
User Discovery

Email address



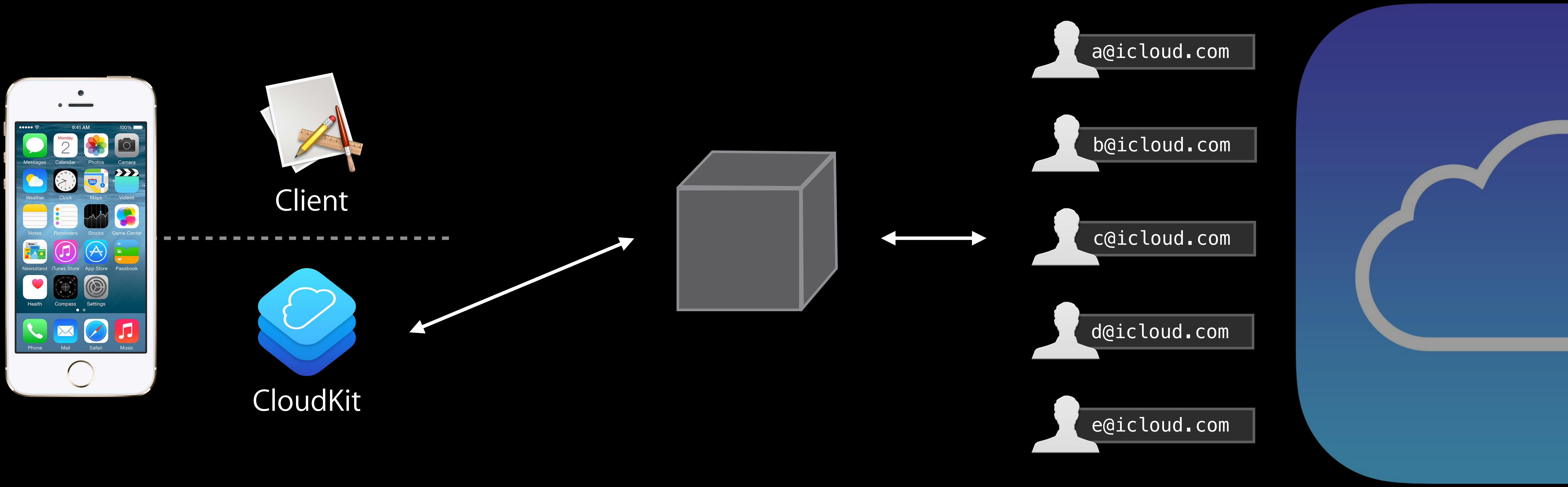
User Discovery

Email address



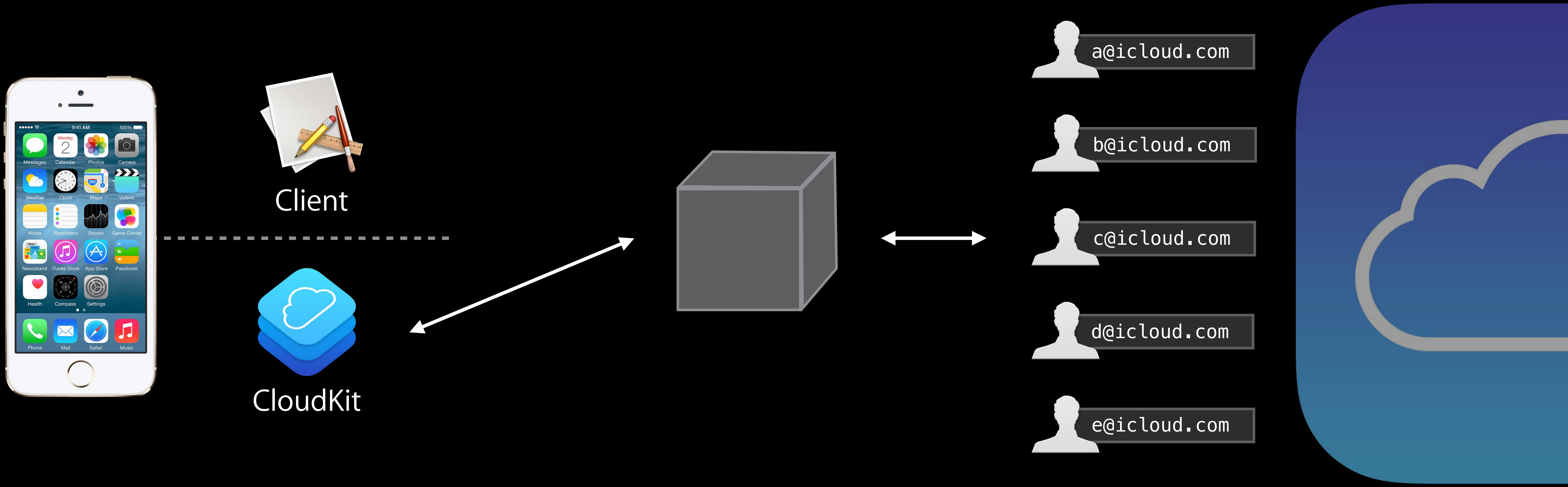
User Discovery

Email address



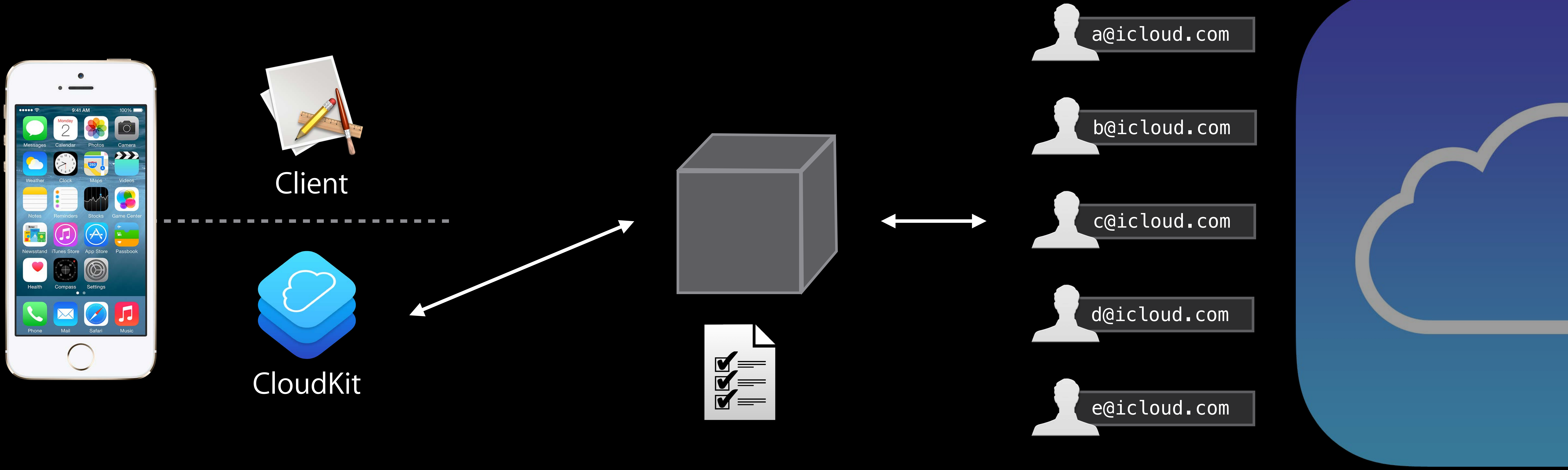
User Discovery

Email address



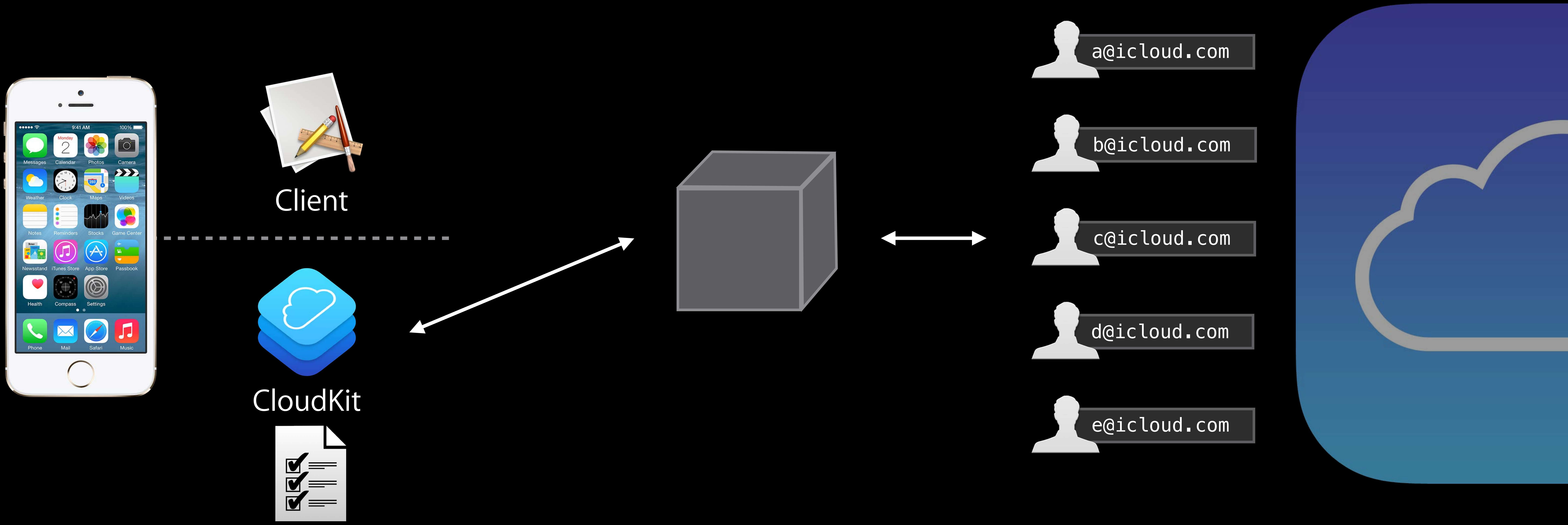
User Discovery

Email address



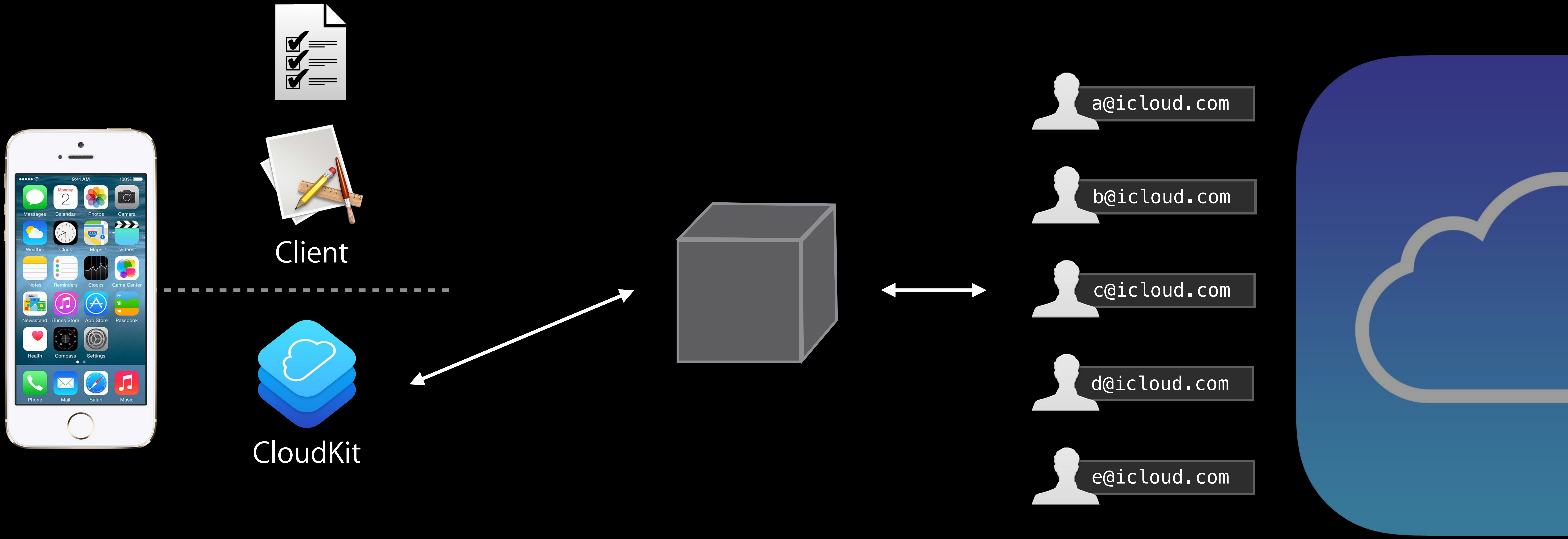
User Discovery

Email address



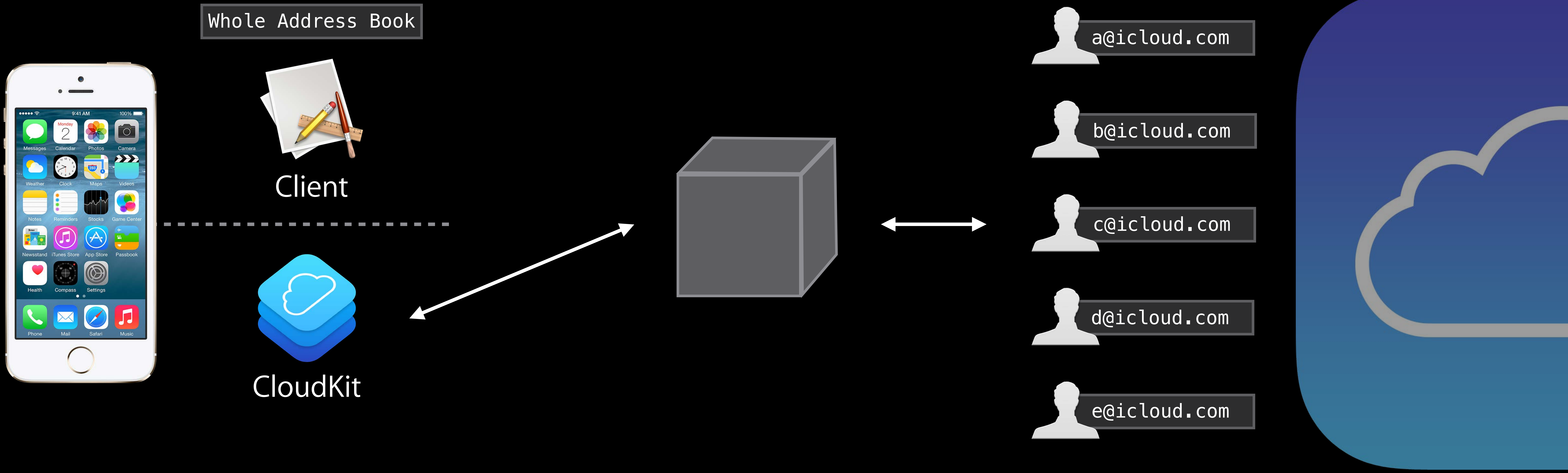
User Discovery

Email address



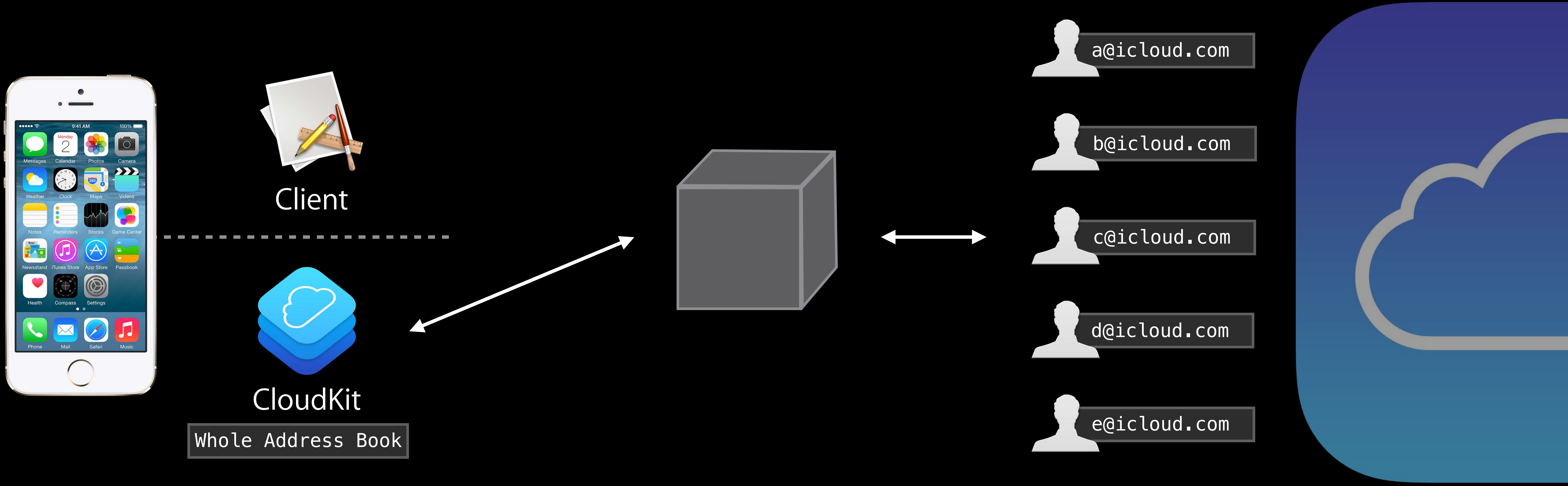
User Discovery

Entire address book



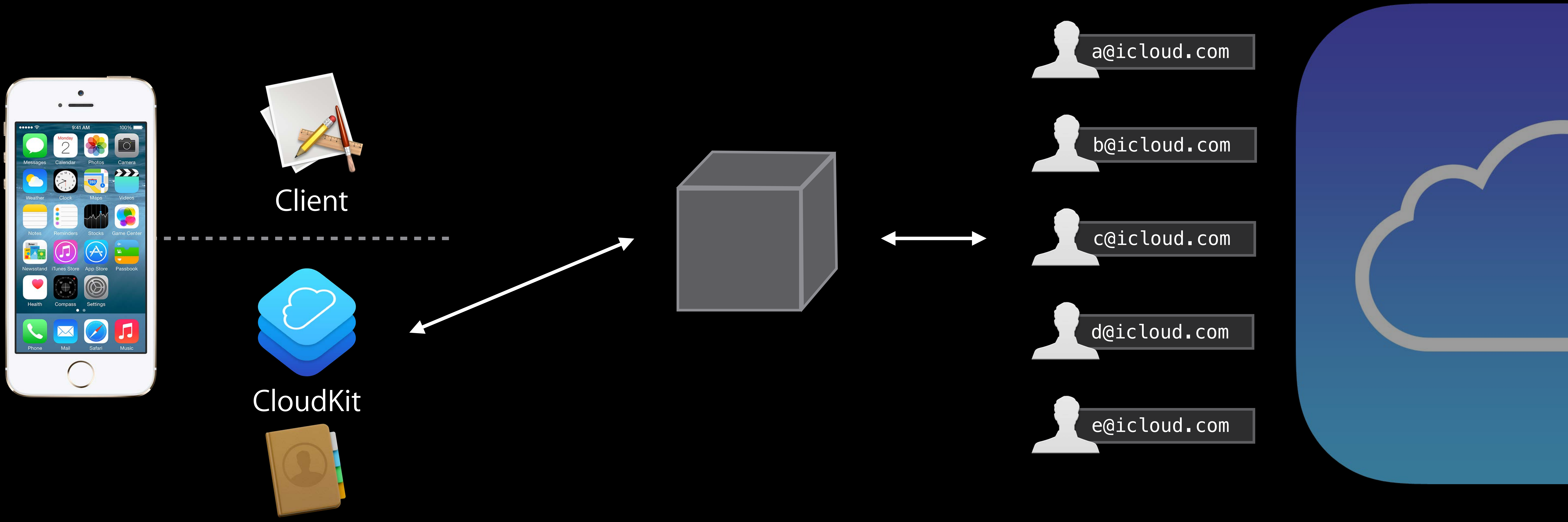
User Discovery

Entire address book



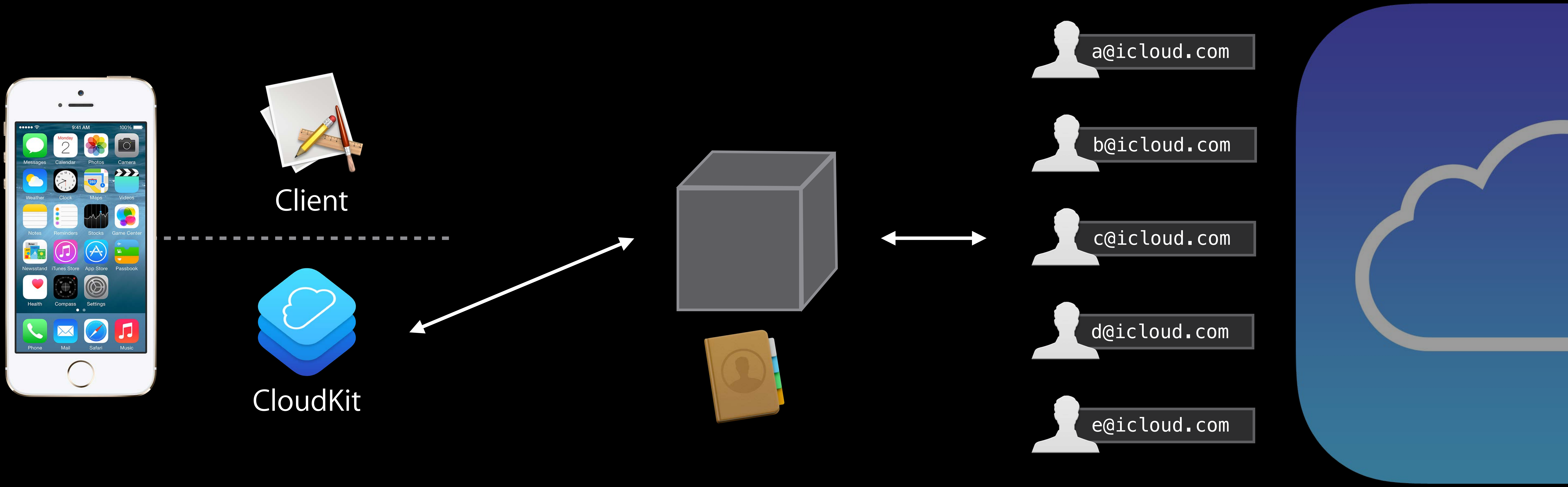
User Discovery

Entire address book



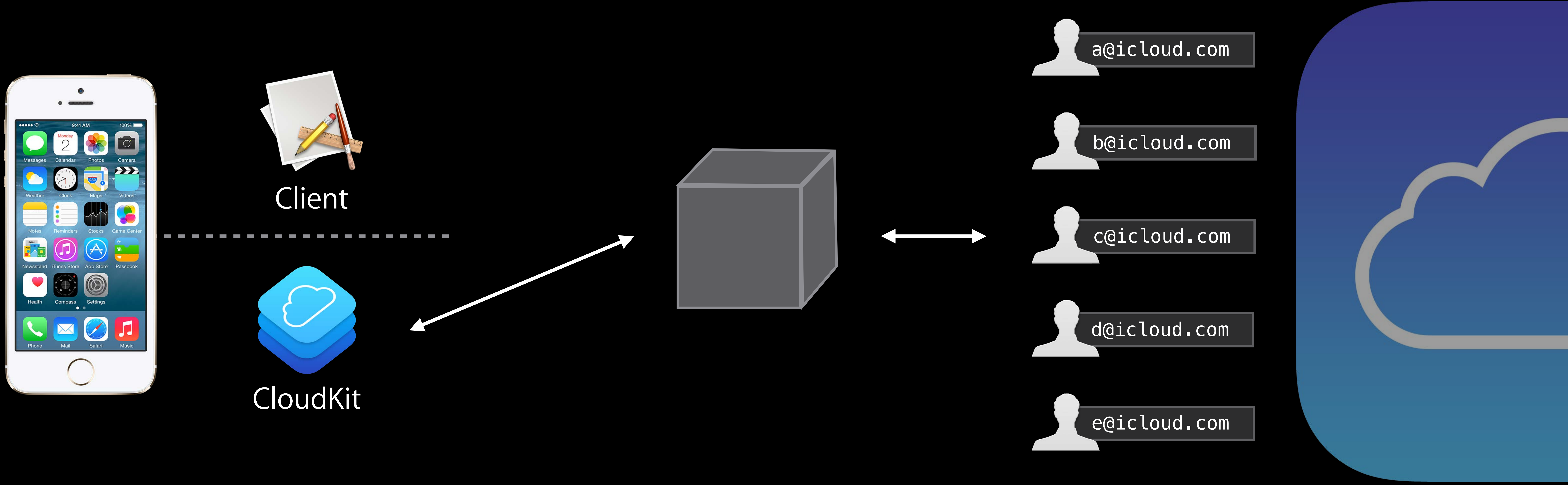
User Discovery

Entire address book



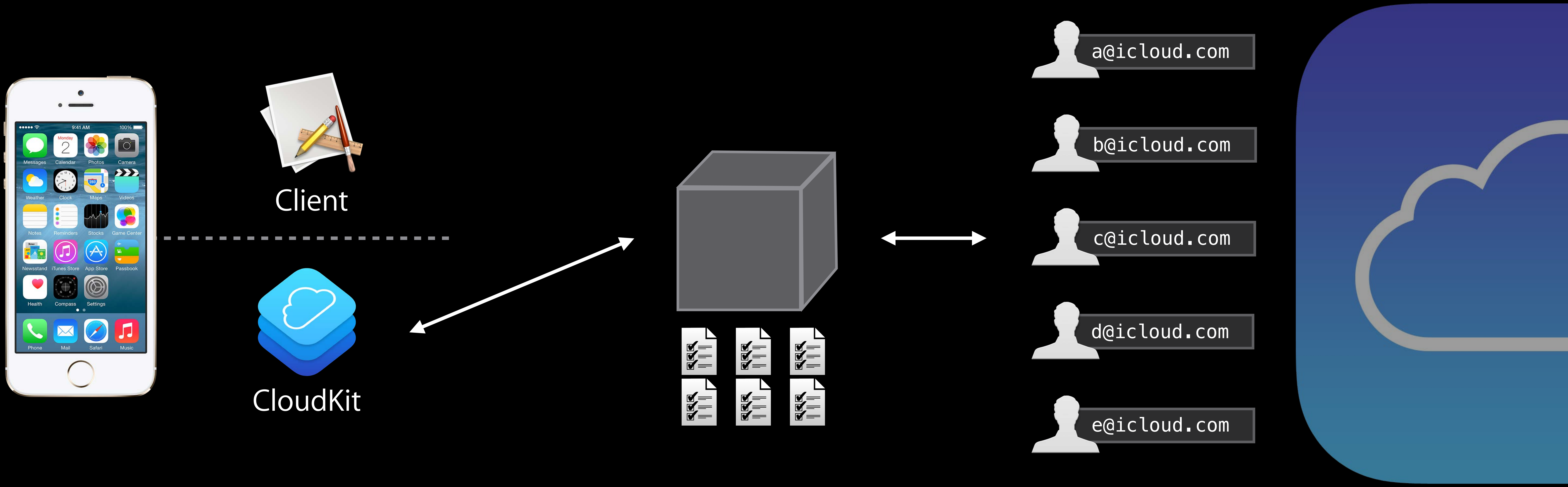
User Discovery

Entire address book



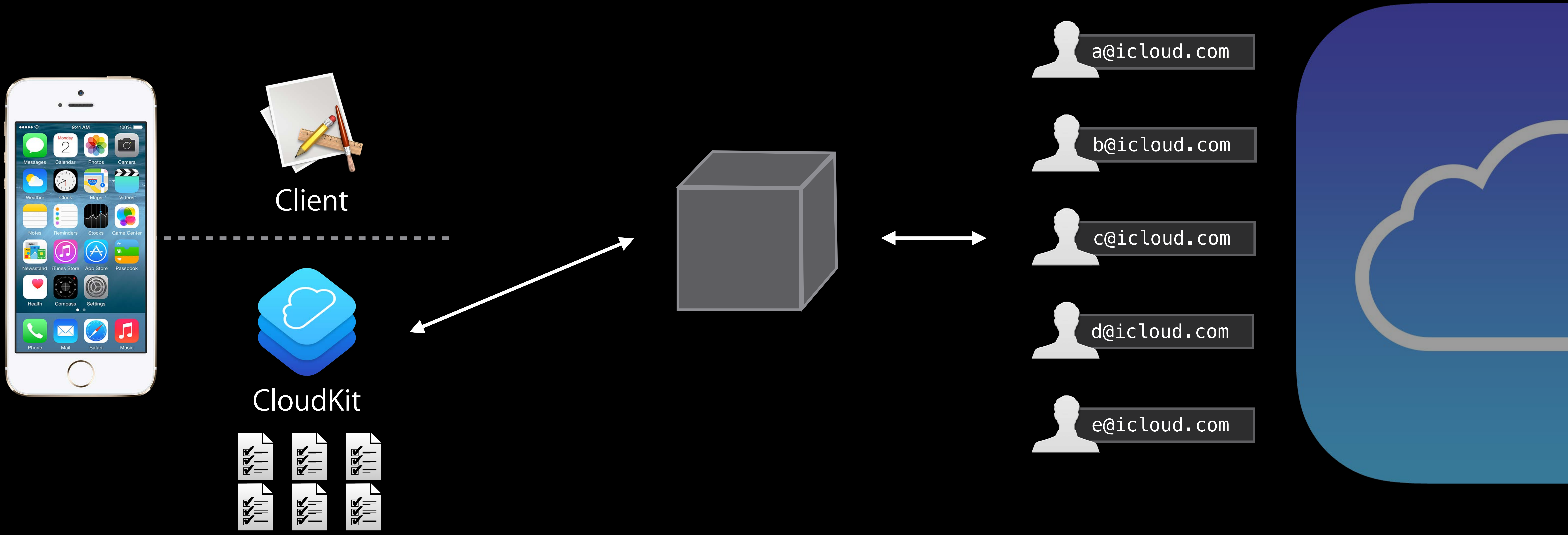
User Discovery

Entire address book



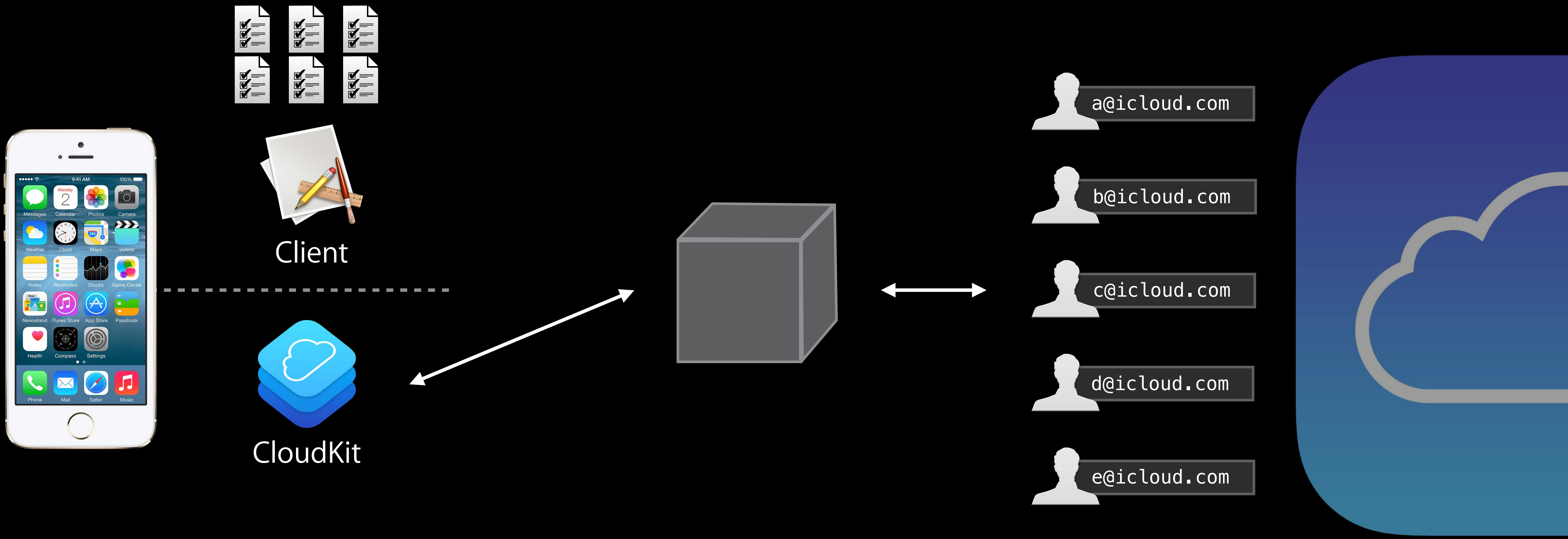
User Discovery

Entire address book



User Discovery

Entire address book



User Discovery

Input

- User RecordID
- Email address
- Entire address book

User Discovery

Input

- User RecordID
- Email address
- Entire address book

Output

User Discovery

Input

- User RecordID
- Email address
- Entire address book

Output

- User RecordID

User Discovery

Input

- User RecordID
- Email address
- Entire address book

Output

- User RecordID
- First and last names

User Discovery

Input

- User RecordID
- Email address
- Entire address book

Output

- User RecordID
- First and last names

Personally identifying information

User Discovery

Input

- User RecordID
- Email address
- Entire address book

Output

- User RecordID
- First and last names

Personally identifying information

- Requires opt-in

User Discovery

```
CKContainer *defaultContainer = [CKContainer defaultContainer];
```

User Discovery

```
CKContainer *defaultContainer = [CKContainer defaultContainer];

[defaultContainer discoverAllContactUserInfosWithCompletionHandler:
    ^(NSArray *userInfos, NSError *error) {
```

User Discovery

```
CKContainer *defaultContainer = [CKContainer defaultContainer];

[defaultContainer discoverAllContactUserInfosWithCompletionHandler:
    ^(NSArray *userInfos, NSError *error) {
    if (error) { ... } else {
       for (CKDiscoveredUserInfo *userInfo in userInfos) {
            NSLog(@"%@: %@ %@",
                  userInfo.userRecordID,
                  userInfo.firstName,
                  userInfo.lastName);
        }
    }
}];
```

CloudKit User Accounts

Identity

Metadata

Privacy

Discovery

When to Use CloudKit

When to Use CloudKit

iCloud Key Value Store

iCloud Drive

iCloud Core Data

CloudKit

When to Use CloudKit

iCloud Key Value Store

- Asynchronously kept up to date
- Data limit constraints
- Great for application preferences

iCloud Drive

iCloud Core Data

CloudKit

When to Use CloudKit

iCloud Key Value Store

iCloud Drive

- Simple API
- Full offline cache on OS X
- Unstructured
- Tied to the filesystem
- Great for document centric apps

iCloud Core Data

CloudKit

When to Use CloudKit

iCloud Key Value Store

iCloud Drive

iCloud Core Data

- Data replicated to all devices
- Data is single-user
- Great for keeping private, structured data in sync

CloudKit

When to Use CloudKit

iCloud Key Value Store

iCloud Drive

iCloud Core Data

CloudKit

- Public data
- Structured and bulk data
- Large data set
- Use iCloud accounts
- Client directed data transfer

Summary

Summary

Access to iCloud servers

Summary

Access to iCloud servers

Public and private data

Summary

Access to iCloud servers

Public and private data

Structured and bulk data

Summary

Access to iCloud servers

Public and private data

Structured and bulk data

Leverage iCloud accounts

Summary

Access to iCloud servers

Public and private data

Structured and bulk data

Leverage iCloud accounts

Apple is building on it

Summary

Access to iCloud servers

Public and private data

Structured and bulk data

Leverage iCloud accounts

Apple is building on it

We're excited to see what you're going to build on this

More Information

Dave DeLong
App Frameworks Evangelist
delong@apple.com

CloudKit Framework Reference
http://developer.apple.com

Apple Developer Forums
http://devforums.apple.com

Related Sessions

● Advanced CloudKit	Mission	Thursday 3:15PM

Labs

CloudKit Lab	Services Lab A	Tuesday 4:30PM
CloudKit Lab	Frameworks Lab B	Wednesday 12:45PM
CloudKit Lab	Frameworks Lab A	Friday 11:30AM

WWDC14