CloudKit

Just another iCloud API?

Michael Ochs

What is CloudKit?

What is CloudKit?

“The CloudKit framework provides interfaces for moving data between your app and your iCloud containers”

- CloudKit Framework Reference

What is CloudKit?

“CloudKit is not a replacement for your app’s existing data objects. Instead, CloudKit provides complementary services for managing the transfer of data to and from iCloud servers.”

- CloudKit Framework Reference

What is CloudKit?

- it is a transfer api to move data to and from the cloud
- it behaves like a remote database in many parts
- but: It is **not** your app's database

Structure

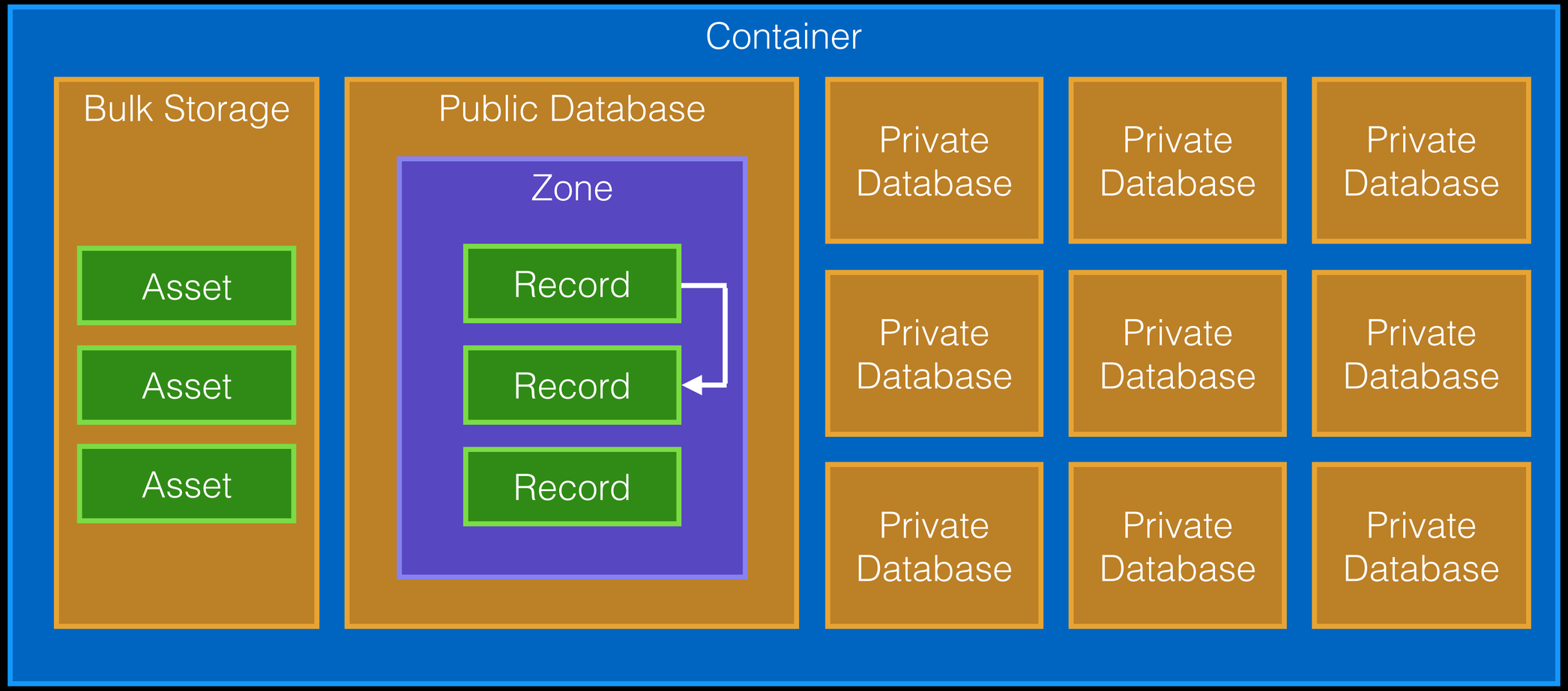
Structure

- local api
- remote database
- remote dashboard

- records
- references
- containers
- queries
- operations
- zones
- subscriptions

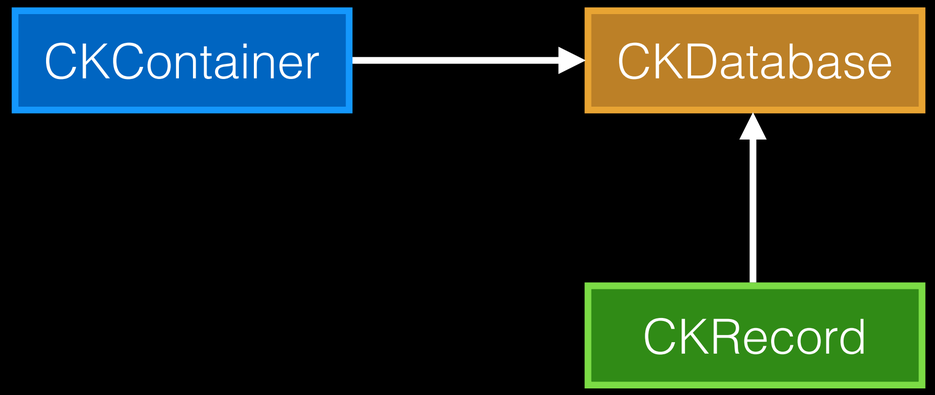
Structure

Data



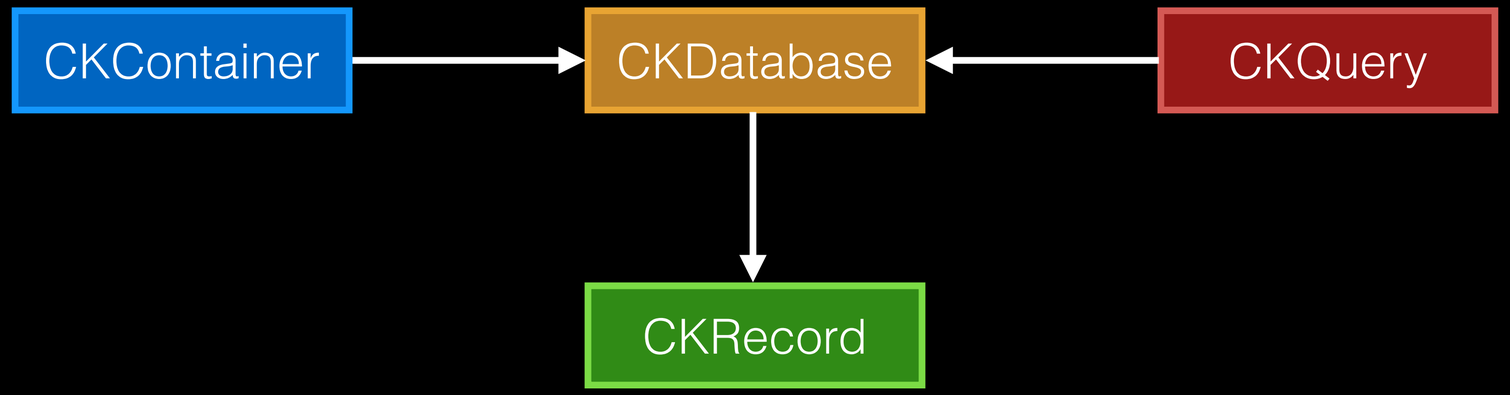
Structure

Convenience API



Structure

Convenience API

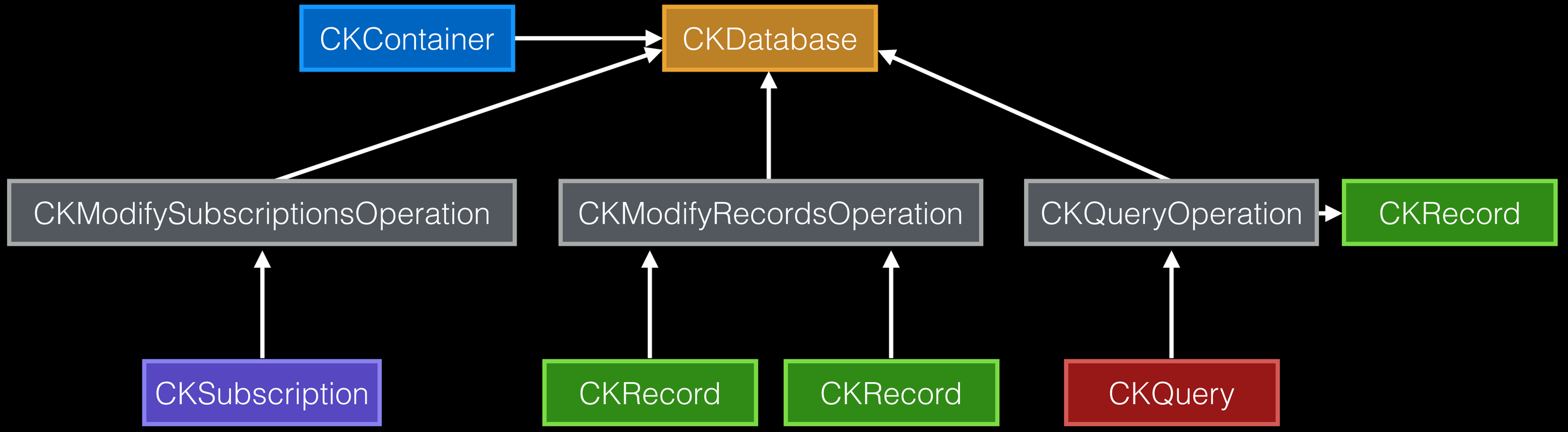


Structure

Convenience API

Structure

Not so convenient API



Structure

Inconvenient API

- CKModifySubscriptionsOperation
- CKModifyRecordsOperation
- CKModifyRecordZonesOperation
- CKFetchSubscriptionsOperation
- CKFetchRecordsOperation
- CKFetchRecordZonesOperation
- CKModifyBadgeOperation
- CKFetchRecordChangesOperation
- CKMarkNotificationsReadOperation
- CKQueryOperation
- CKFetchNotificationChangesOperation
- CKDiscoverAllContactsOperation
- CKDiscoverUserInfosOperation

Structure

CKModifySubscriptionsOperation

CKFetchSubscriptionsOperation

CKModifyBadgeOperation

CKMarkNotificationsReadOperation

CKFetchNotificationChangesOperation

CKModifyRecordsOperation

CKFetchRecordsOperation

CKFetchRecordChangesOperation

CKQueryOperation

InternalError
PartialFailure
NetworkUnavailable
NetworkFailure
BadContainer
ServiceUnavailable
RequestRateLimited
MissingEntitlement
NotAuthenticated
PermissionFailure
UnknownItem
InvalidArguments
ResultsTruncated
ServerRecordChanged
ServerRejectedRequest
AssetFileNotFound
AssetFileModified
IncompatibleVersion
ConstraintViolation
OperationCancelled
ChangeTokenExpired
BatchRequestFailed
ZoneBusy
BadDatabase
QuotaExceeded
ZoneNotFound

CKModifyRecordZonesOperation

CKFetchRecordZonesOperation

CKDiscoverAllContactsOperation

CKDiscoverUserInfosOperation

Structure

CKModifySubscriptionsOperation

CKModifyRecordsOperation

CKModifyRecordZonesOperation

CKFetchSubscriptionsOperation

CKFetchRecordsOperation

CKFetchRecordZonesOperation

CKModifyBadgeOperation

CKFetchRecordChangesOperation

Do not start with CloudKit in your productive application!

CKMarkNotificationsReadOperation

CKQueryOperation

CKFetchNotificationChangesOperation

CKDiscoverAllContactsOperation

CKDiscoverUserInfosOperation

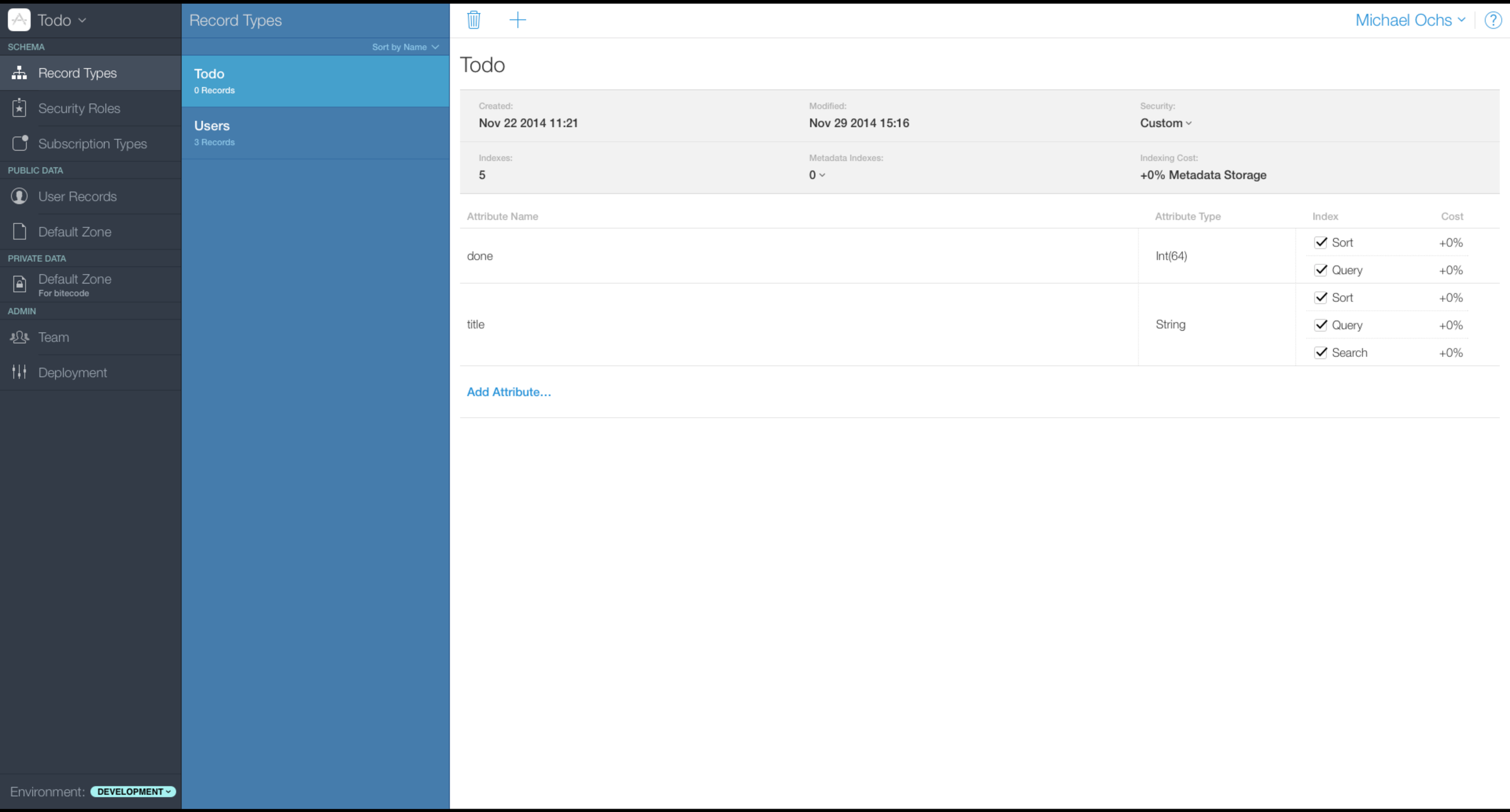
Structure

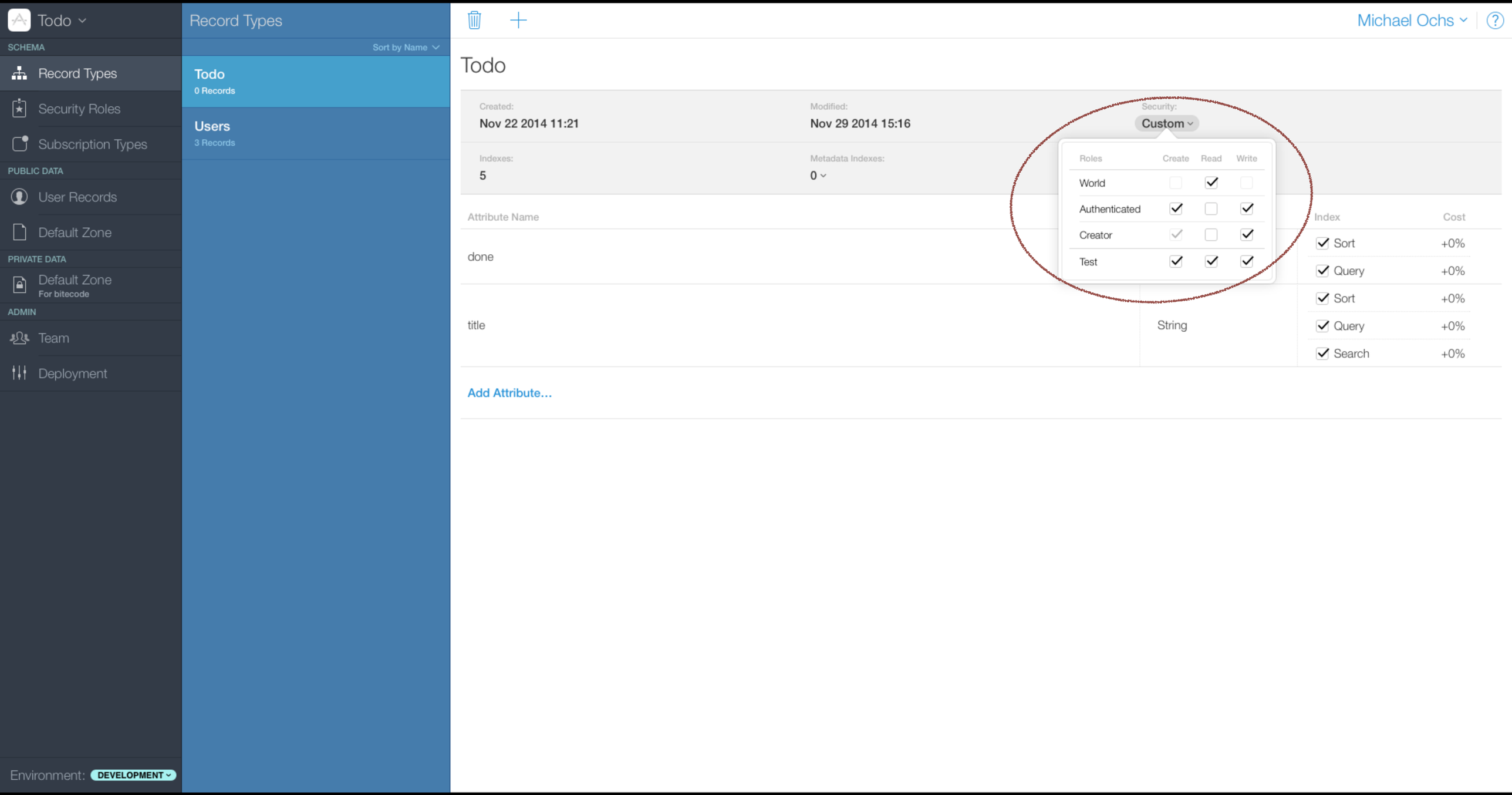
- This api has nothing to do with convenience
- …but this api is great
- It gives you a lot of responsibility
- …but also a lot of power and flexibility

Dashboard

Dashboard

- Web based administration
- View, create, edit, and remove records
- Edit, and remove record layouts
- Edit access groups / privileges


Todo
SCHEMA
Record Types
Security Roles
Subscription Types
PUBLIC DATA
User Records
Default Zone
PRIVATE DATA
Default Zone
For bitecode
ADMIN
Team
Deployment
Environment:
DEVELOPMENT
Record Types
Sort by Name
Todo
0 Records
Users
3 Records
Michael Ochs
Todo
Created:
Nov 22 2014 11:21
Modified:
Nov 29 2014 15:16
Security:
Custom
Indexes:
5
Metadata Indexes:
0
Indexing Cost:
+0% Metadata Storage
Attribute Name
Attribute Type
Index
Cost
done
Int(64)
Sort
+0%
Query
+0%
title
String
Sort
+0%
Query
+0%
Search
+0%
Add Attribute...


Todo
SCHEMA
Record Types
Security Roles
Subscription Types
PUBLIC DATA
User Records
Default Zone
PRIVATE DATA
Default Zone
For bitecode
ADMIN
Team
Deployment
Environment: DEVELOPMENT
Record Types
Sort by Name
Todo
0 Records
Users
3 Records
Michael Ochs
Todo
Created:
Nov 22 2014 11:21
Modified:
Nov 29 2014 15:16
Security:
Custom
Indexes:
5
Metadata Indexes:
0
Roles
Create
Read
Write
World
Authenticated
Creator
Test
Attribute Name
Index
Cost
done
Sort +0%
Query +0%
title
String
Sort +0%
Query +0%
Search +0%
Add Attribute...

Demo

CKDatabase

CKDatabase

- Public database
  - readable by everyone
  - writable by every iCloud user
- Private database
  - readable and writable by the current iCloud user

CKDatabase

Public Database

```
CKContainer *container = [CKContainer defaultContainer];
CKDatabase *database = [container publicCloudDatabase];
```

CKDatabase

Public Database

```
CKContainer *container = [CKContainer defaultContainer];
CKDatabase *database = [container publicCloudDatabase];
```

CKDatabase

Public Database

```
CKContainer *container = [CKContainer defaultContainer];
CKDatabase *database = [container publicCloudDatabase];
```

CKDatabase

Private Database

```
CKContainer *container = [CKContainer defaultContainer];
CKDatabase *database = [container privateCloudDatabase];
```

CKDatabase

Private Database

```
CKContainer *container = [CKContainer defaultContainer];
CKDatabase *database = [container privateCloudDatabase];
```

CKDatabase

```
CKRecordID *recordID = …;
[database deleteRecordWithID:recordID
           completionHandler:^(CKRecordID *recordID, NSError *error) {
    if (error) {
        dispatch_async(dispatch_get_main_queue(), ^{
            [self presentError:error
             completionHandler:^(BOOL didRecover){
                // TODO: handle error
            }];
        });
        return;
    }

    // TODO: handle success
}];
```

CKDatabase

```
CKRecordID *recordID = …;
[database deleteRecordWithID:recordID
           completionHandler:^(CKRecordID *recordID, NSError *error) {
    if (error) {
        dispatch_async(dispatch_get_main_queue(), ^{
            [self presentError:error
             completionHandler:^(BOOL didRecover){
                 // TODO: handle error
            }];
        });
        return;
    }

    // TODO: handle success
}];
```

CKDatabase

```
CKRecordID *recordID = …;
[database deleteRecordWithID:recordID
           completionHandler:^(CKRecordID *recordID, NSError *error) {
    if (error) {
        dispatch_async(dispatch_get_main_queue(), ^{
            [self presentError:error
             completionHandler:^(BOOL didRecover){
                 // TODO: handle error
            }];
        });
        return;
    }

    // TODO: handle success
}];
```

CKRecord

CKRecord

- Data object
- Dictionary like api
- On the fly model generation

CKRecord

Each record has a…

- …record type
- …record id
- …creation date / user record id
- …modification date / user record id

CKRecord

Class	Type
record type	NSString*
record id	CKRecordID*
creation date / user record id	NSDate* / CKRecordID*
modification date / user record id	NSDate* / CKRecordID*

CKRecord

CloudKit	CoreData
record type	entity name
record id	object id
creation date / user record id	n/a
modification date / user record id	n/a

CKRecord

```
CKRecord *record = [[CKRecord alloc] initWithRecordType:@"Todo"];
record[@"title"] = @"Get christmas presents";

[database saveRecord:record
   completionHandler:^(CKRecord *record, NSError *error) {
    if (error) {
        // TODO: handle error
        return;
    }
    // TODO: store record id to your local model
}];
```

CKRecord

```
CKRecord *record = [[CKRecord alloc] initWithRecordType:@"Todo"];
record[@"title"] = @"Get christmas presents";

[database saveRecord:record
   completionHandler:^(CKRecord *record, NSError *error) {
    if (error) {
        // TODO: handle error
        return;
    }
    // TODO: store record id to your local model
}];
```

CKRecord

```
CKRecord *record = [[CKRecord alloc] initWithRecordType:@"Todo"];
record[@"title"] = @"Get christmas presents";

[database saveRecord:record
   completionHandler:^(CKRecord *record, NSError *error) {
    if (error) {
        // TODO: handle error
        return;
    }
    // TODO: store record id to your local model
}];
```

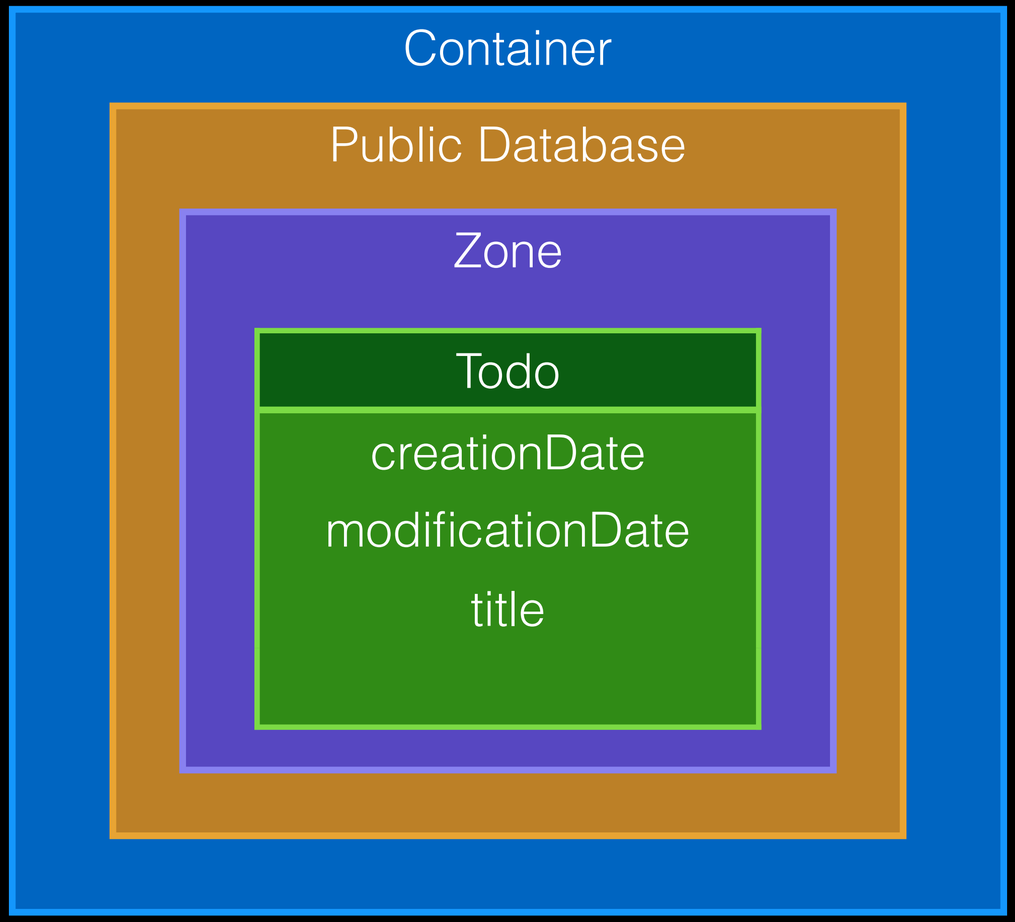
CKRecord

```
CKRecord *record = [[CKRecord alloc] initWithRecordType:@"Todo"];
record[@"title"] = @"Get christmas presents";

[database saveRecord:record
   completionHandler:^(CKRecord *record, NSError *error) {
    if (error) {
        // TODO: handle error
        return;
    }
    // TODO: store record id to your local model
}];
```

CKRecord

```
CKRecord *record = [[CKRecord alloc] initWithRecordType:@"Todo"];
record[@"title"] = @"Get christmas presents";

[database saveRecord:record
   completionHandler:^(CKRecord *record, NSError *error) {
    if (error) {
        // TODO: handle error
        return;
    }
    // TODO: store record id to your local model
}];
```

CKRecord


Container
Public Database
Zone
Todo
creationDate
modificationDate
title

CKRecord

```
CKRecordID *recordID = …; // get record id from your model
[database fetchRecordWithID:recordID
          completionHandler:^(CKRecord *record, NSError *error) {
    if (error) {
        // TODO: handle error
        return;
    }
    record[@"done"] = @YES;

    [database saveRecord:record
       completionHandler:^(CKRecord *record, NSError *error) {
       // TODO: check & handle error
    }];
}];
```

CKRecord

```
CKRecordID *recordID = …; // get record id from your model
[database fetchRecordWithID:recordID
          completionHandler:^(CKRecord *record, NSError *error) {
    if (error) {
        // TODO: handle error
        return;
    }
    record[@"done"] = @YES;

    [database saveRecord:record
       completionHandler:^(CKRecord *record, NSError *error) {
       // TODO: check & handle error
    }];
}];
```

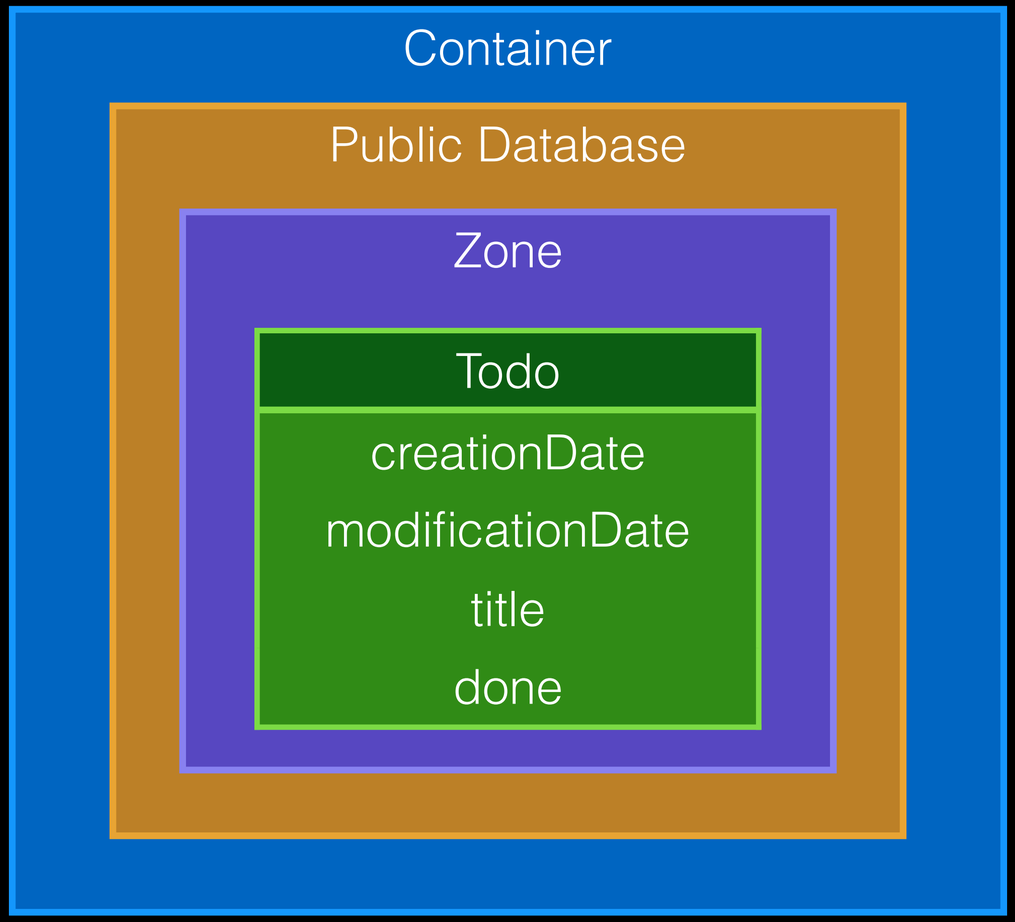
CKRecord

```
CKRecordID *recordID = …; // get record id from your model
[database fetchRecordWithID:recordID
          completionHandler:^(CKRecord *record, NSError *error) {
    if (error) {
        // TODO: handle error
        return;
    }
    record[@"done"] = @YES;

    [database saveRecord:record
       completionHandler:^(CKRecord *record, NSError *error) {
       // TODO: check & handle error
    }];
}];
```

CKRecord

```
CKRecordID *recordID = …; // get record id from your model
[database fetchRecordWithID:recordID
          completionHandler:^(CKRecord *record, NSError *error) {
    if (error) {
        // TODO: handle error
        return;
    }
    record[@"done"] = @YES;

    [database saveRecord:record
       completionHandler:^(CKRecord *record, NSError *error) {
       // TODO: check & handle error
    }];
}];
```

CKRecord


Container
Public Database
Zone
Todo
creationDate
modificationDate
title
done

CKSubscription

CKSubscription

- Subscribe to push notifications
- Bound to a record type & predicate
- on create / on update / on delete
- silent / badge / alert / sound

CKSubscription

- Configure push notifications
- Register for push notifications
- Subscribe to cloud kit

CKSubscription

```
- (BOOL)application:(UIApplication *)application
  didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    [application registerForRemoteNotifications];
    return YES;
}

- (void)application:(UIApplication *)application
  didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken {
    // trigger subscription
}
```

CKSubscription

```
- (BOOL)application:(UIApplication *)application
  didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    [application registerForRemoteNotifications];
    return YES;
}

- (void)application:(UIApplication *)application
  didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken {
    // trigger subscription
}
```

CKSubscription

```
- (BOOL)application:(UIApplication *)application
  didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    [application registerForRemoteNotifications];
    return YES;
}

- (void)application:(UIApplication *)application
  didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken {
    // trigger subscription
}
```

CKSubscription

```
NSPredicate *predicate = [NSPredicate predicateWithValue:YES];

CKSubscriptionOptions options = (
                        CKSubscriptionOptionsFiresOnRecordUpdate |
                        CKSubscriptionOptionsFiresOnRecordDeletion |
                        CKSubscriptionOptionsFiresOnRecordCreation);

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

CKSubscription

```
NSPredicate *predicate = [NSPredicate predicateWithValue:YES];

CKSubscriptionOptions options = (
                        CKSubscriptionOptionsFiresOnRecordUpdate |
                        CKSubscriptionOptionsFiresOnRecordDeletion |
                        CKSubscriptionOptionsFiresOnRecordCreation);

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

CKSubscription

```
NSPredicate *predicate = [NSPredicate predicateWithValue:YES];

CKSubscriptionOptions options = (
                        CKSubscriptionOptionsFiresOnRecordUpdate |
                        CKSubscriptionOptionsFiresOnRecordDeletion |
                        CKSubscriptionOptionsFiresOnRecordCreation);

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

CKSubscription

```
NSPredicate *predicate = [NSPredicate predicateWithValue:YES];

CKSubscriptionOptions options = (
                        CKSubscriptionOptionsFiresOnRecordUpdate |
                        CKSubscriptionOptionsFiresOnRecordDeletion |
                        CKSubscriptionOptionsFiresOnRecordCreation);

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

CKSubscription

```
CKNotificationInfo *notificationInfo = [CKNotificationInfo new];
notificationInfo.shouldSendContentAvailable = YES;

subscription.notificationInfo = notificationInfo;

[database saveSubscription:subscription
         completionHandler:^(CKSubscription *subscription,
                             NSError *error) {
    if (error) {
        // TODO: handle error
        return;
    }
    // TODO: store subscription id
}];
```

CKSubscription

```
CKNotificationInfo *notificationInfo = [CKNotificationInfo new];
notificationInfo.shouldSendContentAvailable = YES;

subscription.notificationInfo = notificationInfo;

[database saveSubscription:subscription
         completionHandler:^(CKSubscription *subscription,
                             NSError *error) {
    if (error) {
        // TODO: handle error
        return;
    }
    // TODO: store subscription id
}];
```

CKSubscription

```
CKNotificationInfo *notificationInfo = [CKNotificationInfo new];
notificationInfo.shouldSendContentAvailable = YES;

subscription.notificationInfo = notificationInfo;

[database saveSubscription:subscription
         completionHandler:^(CKSubscription *subscription,
                             NSError *error) {
    if (error) {
        // TODO: handle error
        return;
    }
    // TODO: store subscription id
}];
```

CKSubscription

```
CKNotificationInfo *notificationInfo = [CKNotificationInfo new];
notificationInfo.shouldSendContentAvailable = YES;

subscription.notificationInfo = notificationInfo;

[database saveSubscription:subscription
         completionHandler:^(CKSubscription *subscription,
                             NSError *error) {
    if (error) {
        // TODO: handle error
        return;
    }
    // TODO: store subscription id
}];
```

CKSubscription

```
CKNotificationInfo *notificationInfo = [CKNotificationInfo new];
notificationInfo.shouldSendContentAvailable = YES;

subscription.notificationInfo = notificationInfo;

[database saveSubscription:subscription
         completionHandler:^(CKSubscription *subscription,
                             NSError *error) {
    if (error) {
        // TODO: handle error
        return;
    }
    // TODO: store subscription id
}];
```

CKSubscription

```
CKNotificationInfo *notificationInfo = [CKNotificationInfo new];
notificationInfo.shouldSendContentAvailable = YES;

subscription.notificationInfo = notificationInfo;

[database saveSubscription:subscription
         completionHandler:^(CKSubscription *subscription,
                             NSError *error) {
    if (error) {
        // TODO: handle error
        return;
    }
    // TODO: store subscription id
}];
```

CKSubscription

```
- (void)application:(UIApplication *)application
  didReceiveRemoteNotification:(NSDictionary *)userInfo
        fetchCompletionHandler:
                (void(^)(UIBackgroundFetchResult))completionHandler {

    // TODO: fetch updates and handle them
    completionHandler(UIBackgroundFetchResultNewData);

}
```

CKSubscription

```
- (void)application:(UIApplication *)application
  didReceiveRemoteNotification:(NSDictionary *)userInfo
        fetchCompletionHandler:
                (void(^)(UIBackgroundFetchResult))completionHandler {

    // TODO: fetch updates and handle them
    completionHandler(UIBackgroundFetchResultNewData);

}
```

CKSubscription

```
- (void)application:(UIApplication *)application
  didReceiveRemoteNotification:(NSDictionary *)userInfo
        fetchCompletionHandler:
                (void(^)(UIBackgroundFetchResult))completionHandler {

    // TODO: fetch updates and handle them
    completionHandler(UIBackgroundFetchResultNewData);

}
```

CKSubscription

- mark notifications as read
- fetch missed notifications
- handle badges on all devices

// TODO: handle error

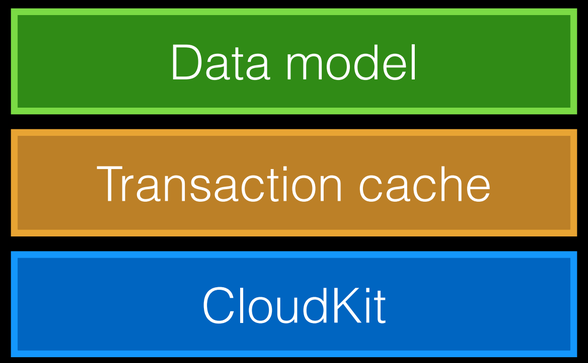
// TODO: handle error

- You **need** to handle them
- Otherwise your data models will become inconsistent
- They might occur often

// TODO: handle error

- Handle errors in a central place if possible
- HRSCustomErrorHandling might help you

// TODO: handle error



// TODO: handle error

Problems

Problems

- convenient API can not handle complexity of CloudKit
- lack of documentation
- strange behavior
- iOS simulator is not working
- privileges handling is lacking features

Next steps

Next steps

- experiment with the convenient api
- check if CloudKit is the right iCloud api for your task
- move to the operation based api
- get your models together

Feedback / Questions

@_mochs

ios-coding.com

Thank you