Introducing
Realm Mobile Platform

Build Better Apps Faster!

About Realm

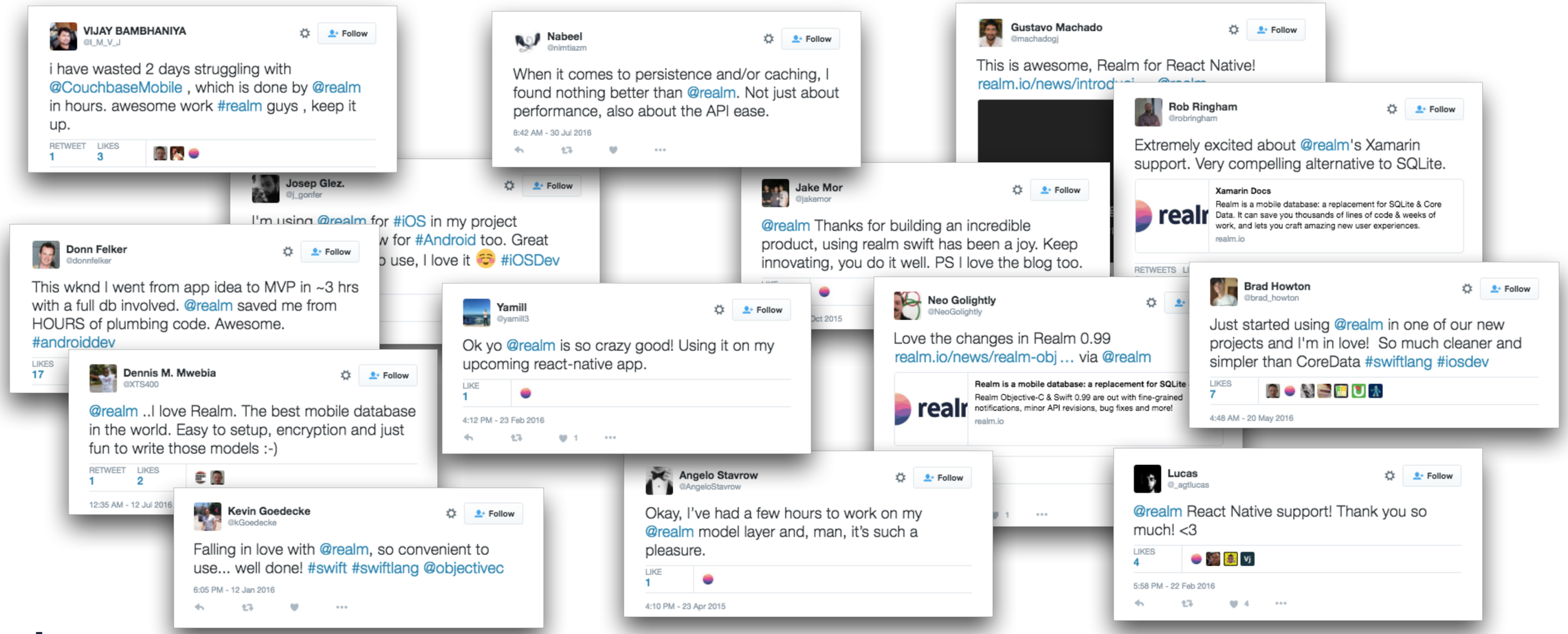
- Headquartered in San Francisco, with engineering office in Copenhagen
- ~50 employees
- Series B startup, $29M raised

Realm Mobile Database

- On-device cross-platform **object** database
- Launched July 2014
- Developed in the open, fully open source
- 16K+ GitHub stars, 100K+ active developers

realm

Developers Love Realm for Power, Simplicity, Ease of Use



realm

Apps of Every Type, 1.000.000.000+ installs

Realm Mobile Platform

Demo

Users Expect More of Apps Today

- **Offline first:** They need apps that work well offline and when network connections are intermittent
- **Reactive:** They expect apps that are highly responsive, with all changes immediately reflected in the UI
- **Real-time:** They expect every user and device to stay in sync with each other in real-time
- **Collaborative:** They want highly collaborative features and experiences within their apps
- **Seamless:** They expect to be be able to move seamlessly from device to device

Developer priorities

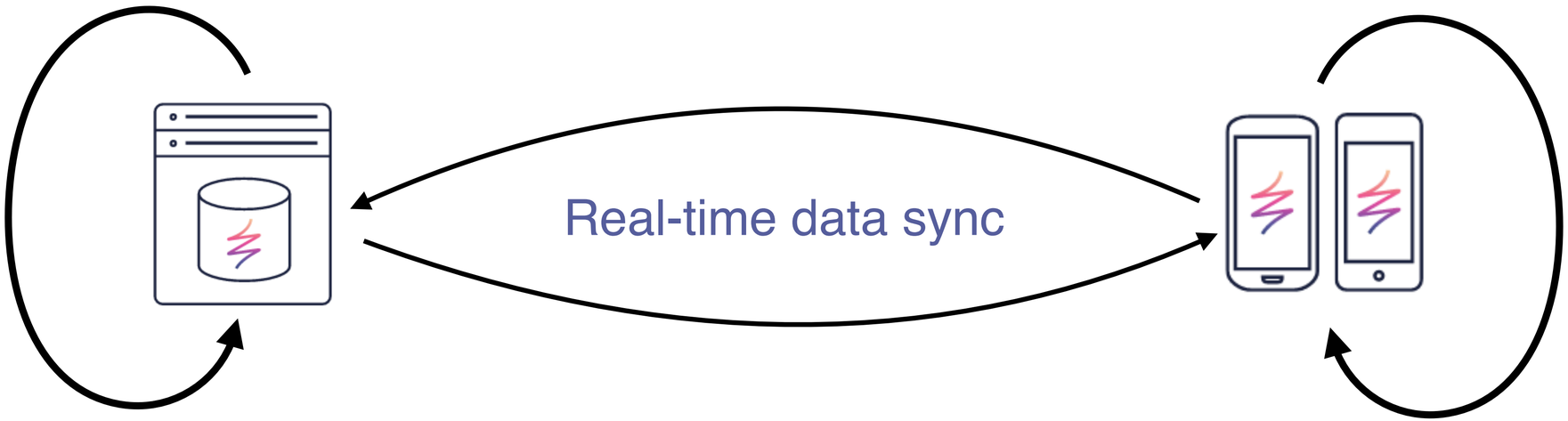
- **Time-to-market:** Your team need to be able to deliver quality apps quickly and on time
- **Ability:** With high user expectations, the team need the ability to easily add advanced features like data sync, messaging and collaboration.
- **Cross-platform:** To reach the full audience, you need to be able to deliver apps for both iOS and Android
- **Standardize:** To manage all the apps and reduce complexity, you want to standardize on a single platform

The Realm Mobile Platform

NEW

Realm Object Server

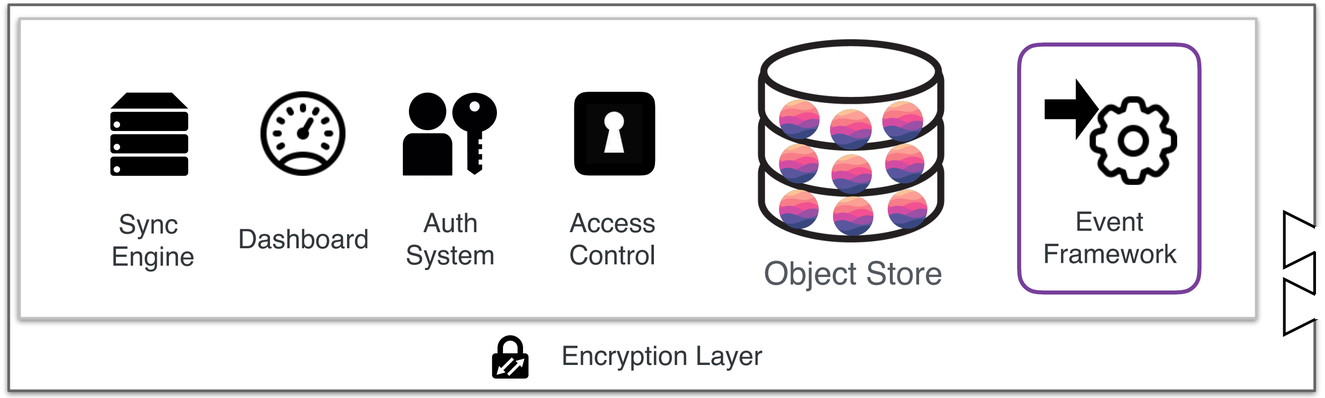
Realm Mobile

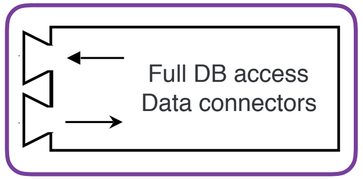


realm

What is it?

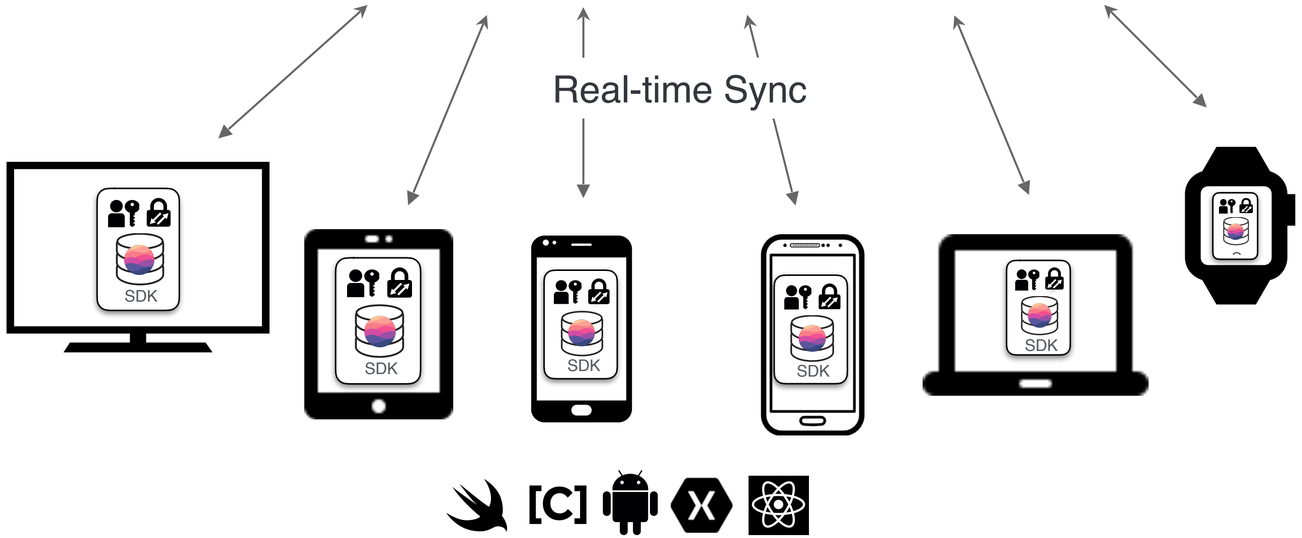
Realm Object Server NEW





Enterprise Feature

Realm Mobile Database



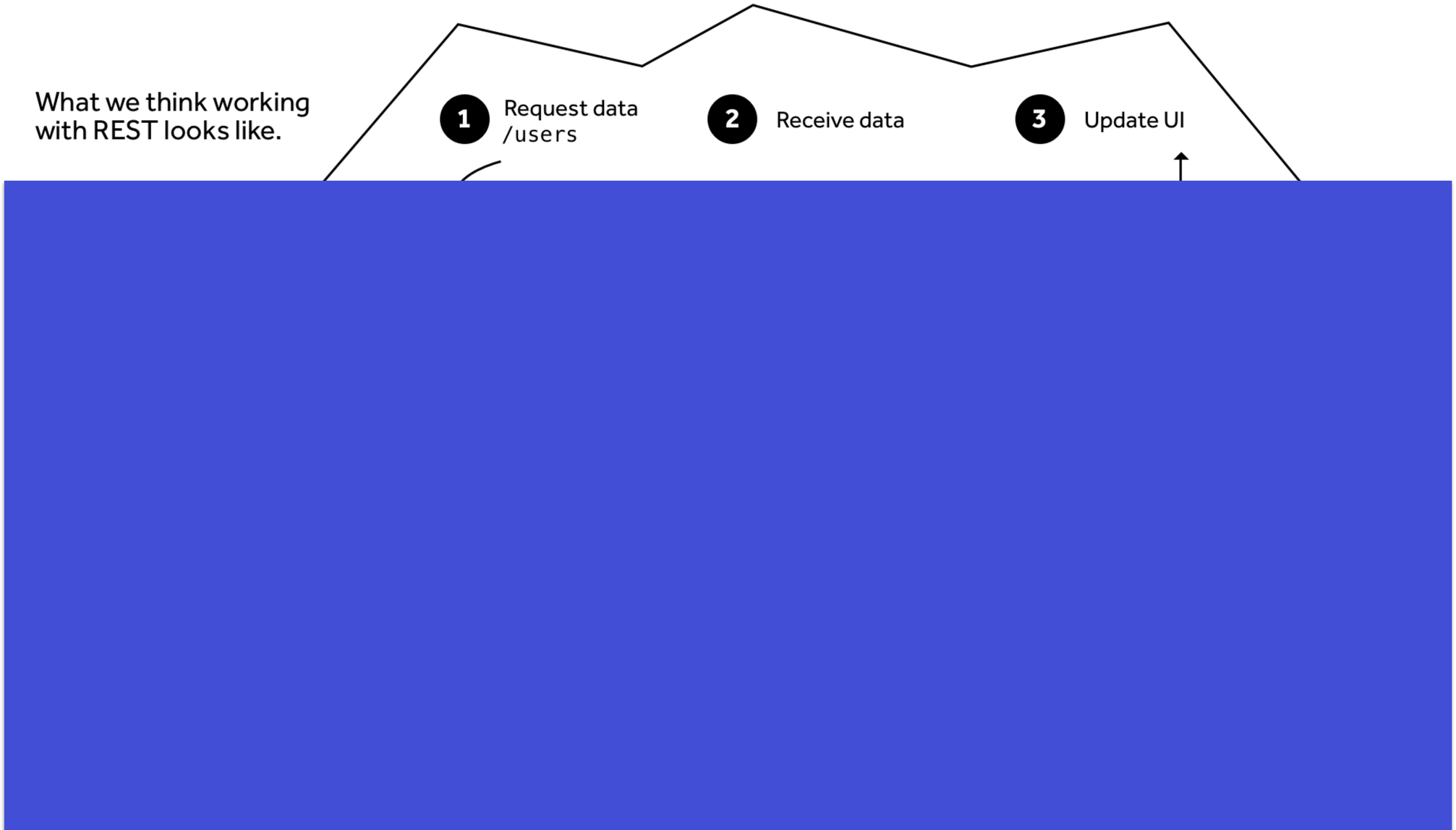
realm

Delivers Key Mobile Solutions

- Eliminates network challenges
- Simplifies data architecture
- Unlocks existing infrastructure

realm

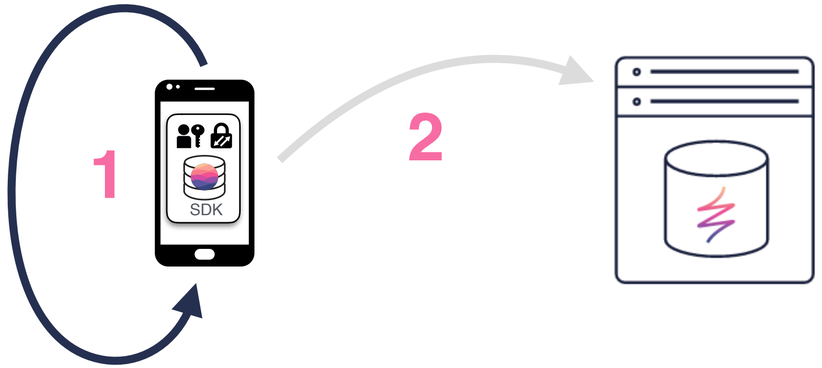
The Reality With REST



realm

Eliminate Network Challenges

- Offline-first
- Full database on the device
  - Queries run locally
  - Writes apply immediately
- Reactive architecture keeps UI up-to-date
- Automatic sync happens in the background
- Resilient to any network conditions



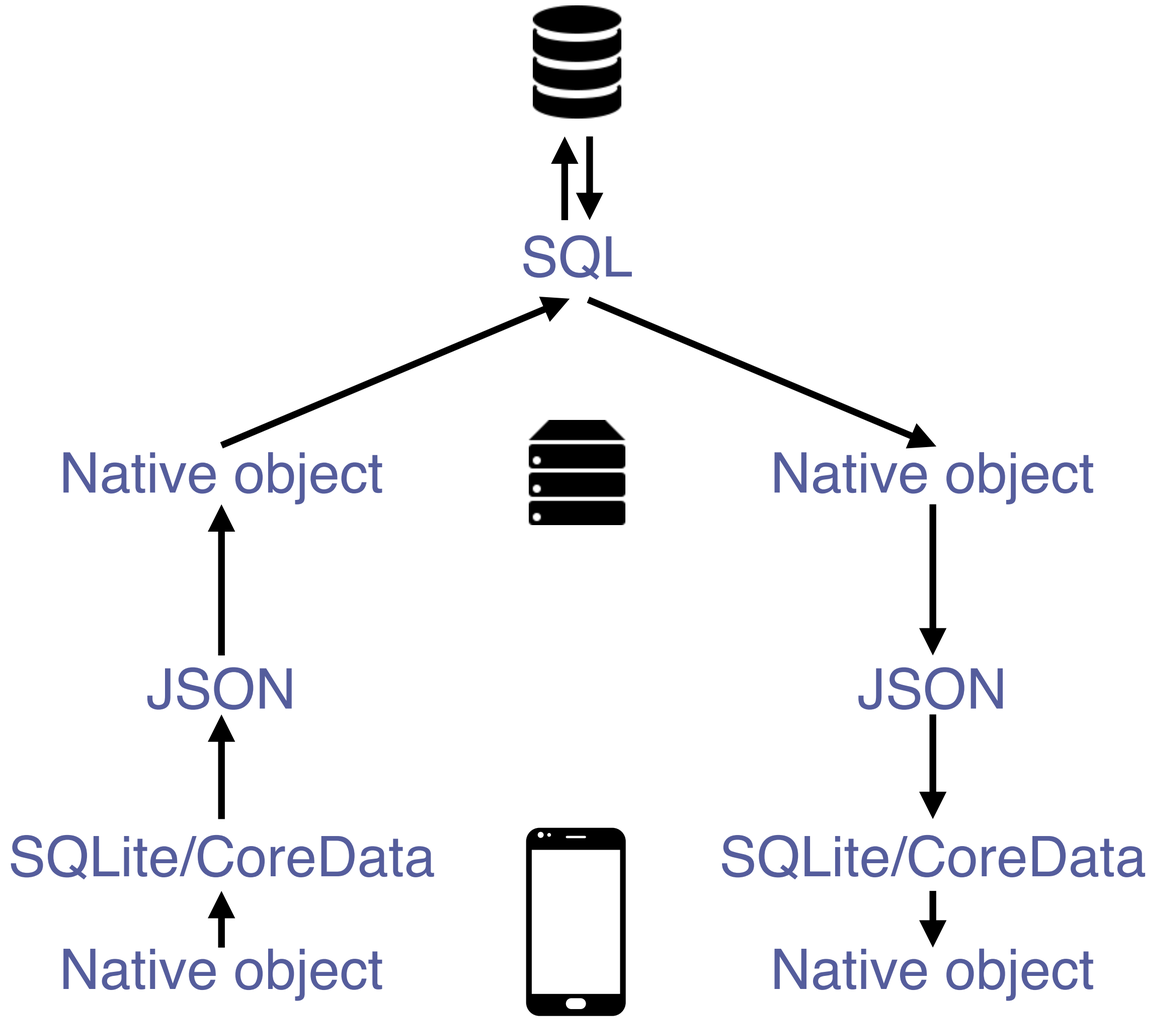
realm

Delivers Key Mobile Solutions

- Eliminates network challenges
- Simplifies data architecture
- Unlocks existing infrastructure

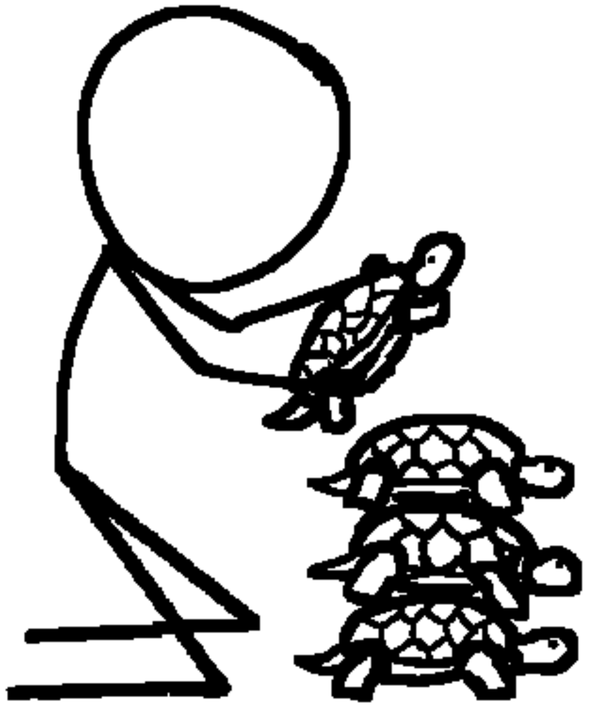
realm

Typical Application Data Flow



realm

Simplify Data Architecture



Objects all the way down

- **It's just objects**
  on the device and on the server
- **Objects are *live,***
  they update—automatically and immediately—in response to changes
- **Not an ORM,**
  the database *is* the data model
- **Easier for developers**
  to build, change, maintain
- **Reduce code complexity**
  via unified data format; cross-platform
- **Encrypted at rest & transit**

realm

Delivers Key Mobile Solutions

- Eliminates network challenges
- Simplifies data architecture
- Unlocks existing infrastructure

realm

Legacy Systems Hold Back Mobile

SOAP
XML

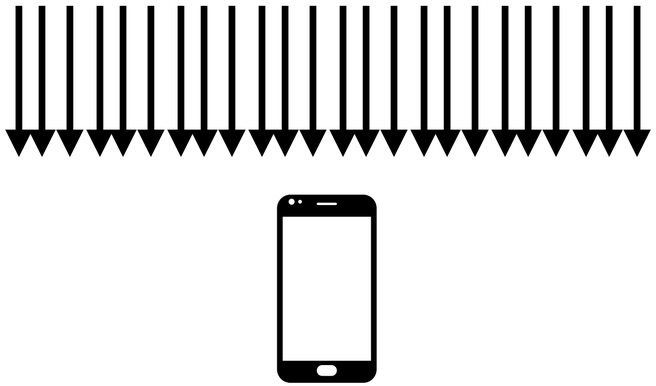
REST
JSON

/facilities

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <department>manufacturing</department>
  <facilities>west-1A</facilities>
</root>
```

/departments

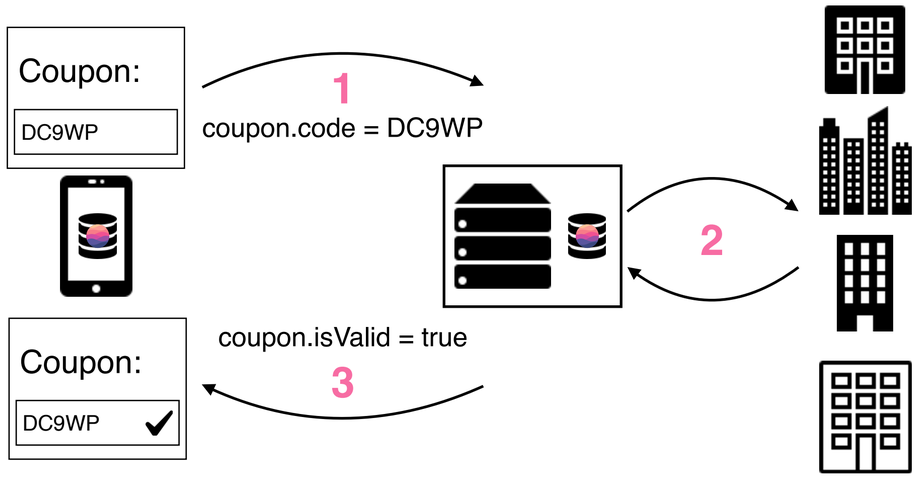
```
{"facility": "west-1A",
   "departments": [
     {"type": "manufacturing"},
     {"type": "operations"}]
}
```



realm

Unlock Existing Infrastructure

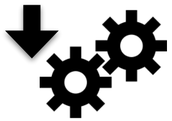
- Bridge legacy APIs
  - Shift complexity to backend
- Node.js SDK
  - Identical object interface
  - Full database capabilities
  - Apply changes to push data
- Event framework
  - Listen and respond to changes from clients
  - Pass along data to other systems or databases
- Supports custom authentication



realm

Hard features made easy

Offline First Experiences



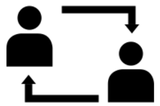
Event Processing

Real-time 2-Way Data Sync

API Mobilization

Presence

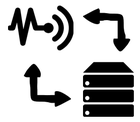
Messaging / Chat



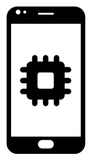
Realtime Collaboration

Pub/Sub & Push Notifications

In-App & Cross-App Search



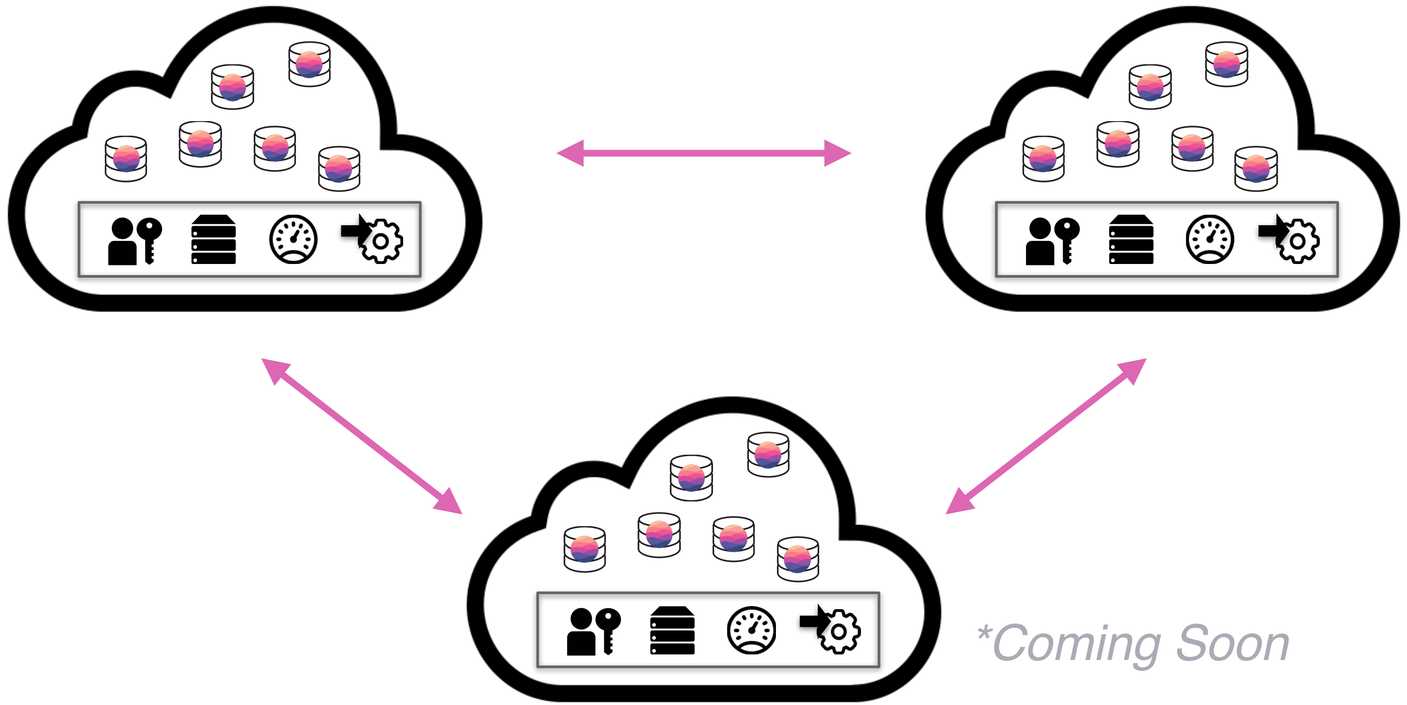
IoT Sensor Sync



Endpoint Computing

realm

Scale with server-side support for **high-availability clustering***



*Coming Soon

realm

The code centric view

Realm is an **object** database

Create native objects that map **directly** to the database.

```
// Define your models like regular Swift classes
class Dog: Object {
    dynamic var name = ""
    dynamic var age = 0
}
class Person: Object {
    dynamic var name = ""
    dynamic var picture: NSData? = nil // optionals supported
    let dogs = List<Dog>()
}
```

realm

Perform **complex** queries across your object graph

```
// Query using a predicate string
var tanDogs = realm.objects(Dog.self).filter("color = 'tan' AND name BEGINSWITH 'B'")

// Query using an NSPredicate
let predicate = NSPredicate(format: "color = %@ AND name BEGINSWITH %@", "tan", "B")
tanDogs = realm.objects(Dog.self).filter(predicate)

// Sort tan dogs by name
let sortedDogs = realm.objects(Dog.self).filter("color = 'tan'").sorted("name")
```

realm

All changes occur in transactions offering **ACID** compliance

```
let cheeseBook = Book()
cheeseBook.title = "Gouda recipes"
cheeseBook.id = 1

// Add object in write transaction
try! realm.write {
    realm.add(cheeseBook)
}
```

```
// Creating a book with the same primary
// key as a previously saved book
let cheeseBook = Book()
cheeseBook.title = "Cheddar recipes"
cheeseBook.id = 1

// Updating book with id = 1
try! realm.write {
    realm.add(cheeseBook, update: true)
}
```

realm

But Realm objects have a **special** capability…

Realm objects **live-update** in response to changes

```
// Query Realm for all dogs
let puppies = realm.objects(Dog.self).filter("age < 2")
puppies.count // => 0

let myDog = Dog()
myDog.name = "Rex"
myDog.age = 1

try! realm.write {
    realm.add(myDog)
}

puppies.count // => 1
```

realm

Realm is a **live** **object** database

~~Realm is an **object** database~~

realm

Listen for live object changes

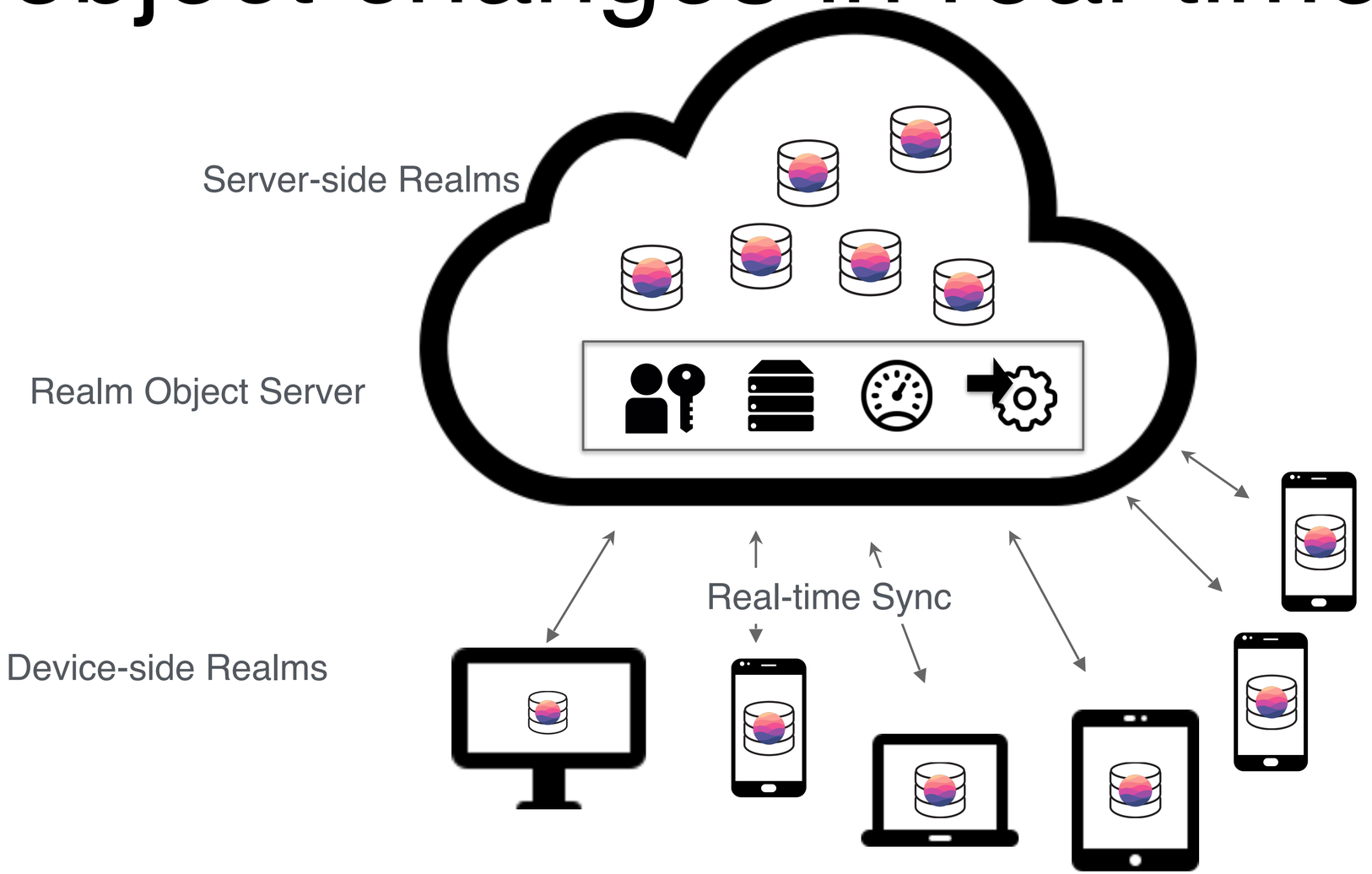
to simplify your architecture

```
let realm = try! Realm()
notificationToken = realm.objects(Dog.self).filter("age > 5").addNotificationBlock { changes in
    switch changes {
    case .initial:
        tableView.reloadData()
        break
    case .update(_, let deletions, let insertions, let modifications):
        // Update table view to animate deletions, insertions and modifications
        break
    case .error(let error):
        // handle error
        break
    }
}
```

realm

Now imagine if these live-updates came from **remote** changes?

Realm Object Server **synchronizes** object changes in real-time.



realm

Push object changes server-side instantly via Node.js interface

```
// Open a user's Realm server-side in Javascript
let realm = new Realm({
    schema: [{
        name: 'Dog',
        properties: {
            name: {type: 'string'},
            age: {type: 'int'}
        },
        syncUrl: syncUrl,
        syncUserToken: syncToken
    }]
});

// Add a Dog object to instantly push change to client
realm.write(() => {
    realm.create('Dog', {name: 'Rex', age: 1});
}
```

realm

Authenticating a user

```
let serverURL = NSURL(string: "http://my.realmServer.com:9080")!
let usernameCredential = Credential.usernamePassword(username: “username",
              password: "password", actions: [.createAccount])

SyncUser.authenticate(with: credential, server: serverURL) { user, error in
                        if let user = user {
                            // can now open a synchronized Realm with this user
                        } else if let error = error {
                            // handle error
                        }
}
```

realm

Open a Realm instance and get ready for synchronization

```
// Create the configuration
// user and serverURL as before
let config = Realm.Configuration(syncConfiguration: (user, serverURL))

// Open the remote Realm
let realm = try! Realm(configuration: config)
// Any changes made to this Realm will be synced across all devices!
```

realm

Only Additive schema changes

Standalone Realms support automatic migrations

Sync'ed Realms only support additive changes

- Adding class
- Adding property
- Removing property
- Adding and removing search index

realm

Conflicts will occur

Deletes always win: If one side deletes an object it will always stay deleted

Last update wins: If two sides has updated the same property, the value will end up as the last updated.

Inserts in lists are ordered by time: If two items are inserted at the same position, the item that was inserted first will end up before the other item.

Trigger server-side functions based on data changes across Realms

```
var change_notification_callback = function(realm_name, realm) {
    // Get the label scan object to process OCR
    var scan = realm.objects("LabelScan").filtered("status = $0", kUploadingStatus)[0];

    // Update label scan status to push change to client
    realm.write(function() {
        scan.status = kProcessingStatus;
    });

    // Send image to Google Vision API
    var res = request('POST',
                      "https://vision.googleapis.com/v1/images:annotate?key=###",
                      {json: {"requests": [{"image": {"content": scan.imageBase64String},
                        "features": [{"type": "TEXT_DETECTION","maxResults": 10}]}]}
    });
    var foundText = textFromJSON(JSON.parse(res.getBody('utf8')));

    // Save the found text to push change to client
    realm.write(function() {
        scan.status = kResultsReadyStatus;
        scan.result = foundText;
    });
}
```

realm

Just to Sum Up

Realm Mobile Database: local object database
Open Source Software (Apache 2 license)

Realm Mobile Platform: synchronize the database with server and multiple devices

Realm Dashboard: shows you statistics, execute triggers, …

realm

Questions?

Try @ realm.io

Build Better Apps Faster!