THE UNIVERSITY OF CHICAGO

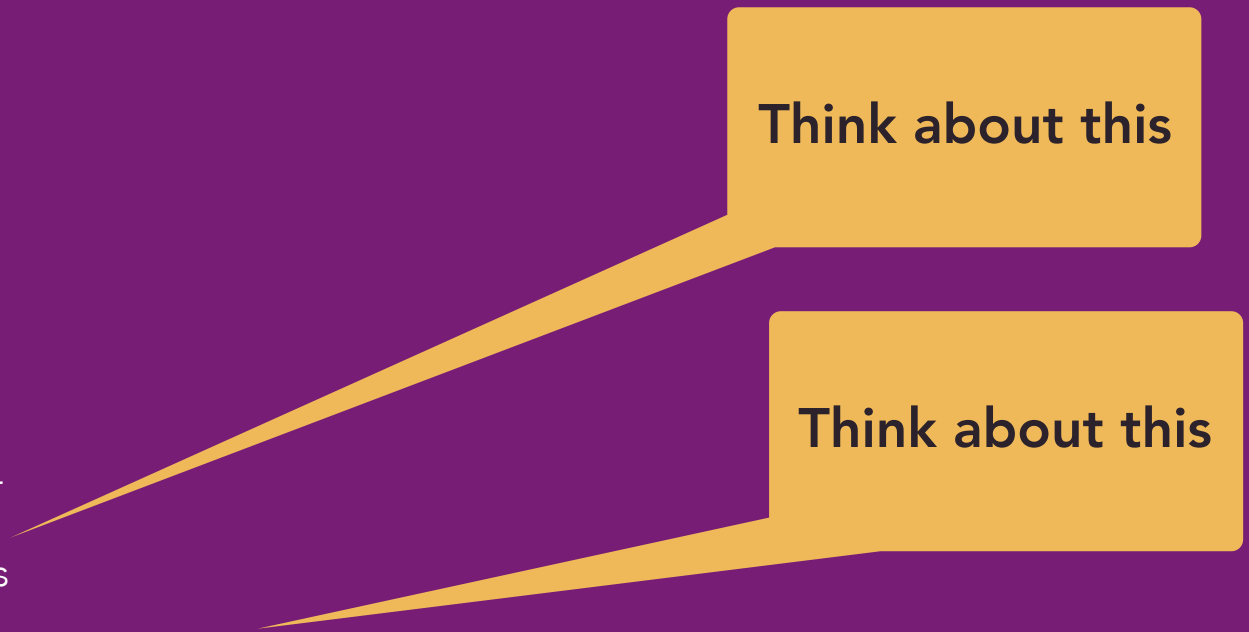
MPCS 51033 • AUTUMN 2019 • SESSION 5

BACKENDS FOR MOBILE APPLICATIONS

CLASS NEWS

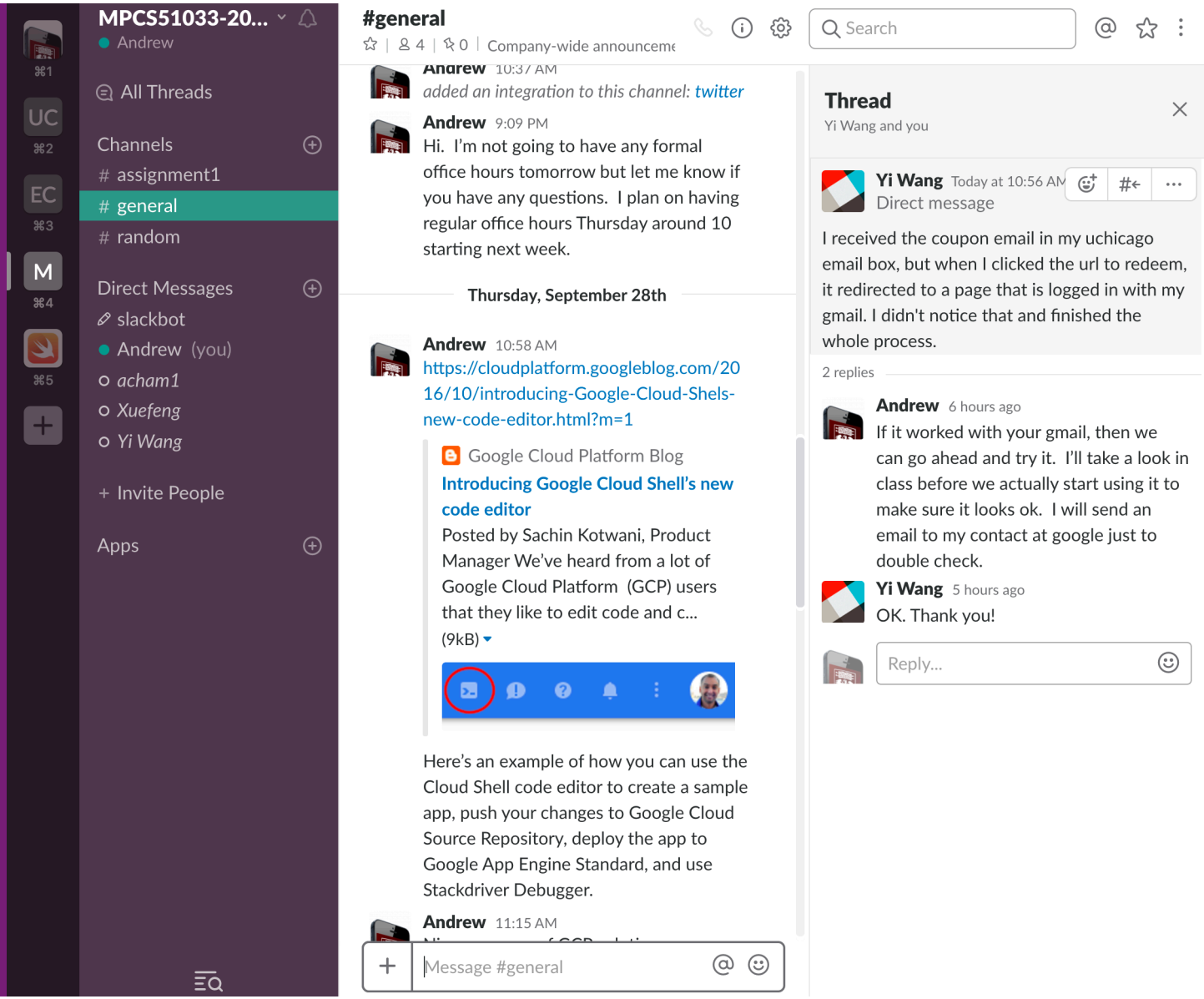
COURSE DESIGN

- Week 1 - Google Cloud Platform
- Week 2 - Google App Engine
- Week 3 - Google App Engine
- Week 4 - Firebase
- Week 5 - Firebase
- Week 6 - CloudKit
- Week 7 - CloudKit/Swift on Server
- Week 8 - Other Stuff/Case Studies
- Week 9 - Project Presentations (Last class meeting)



CLASS NEWS

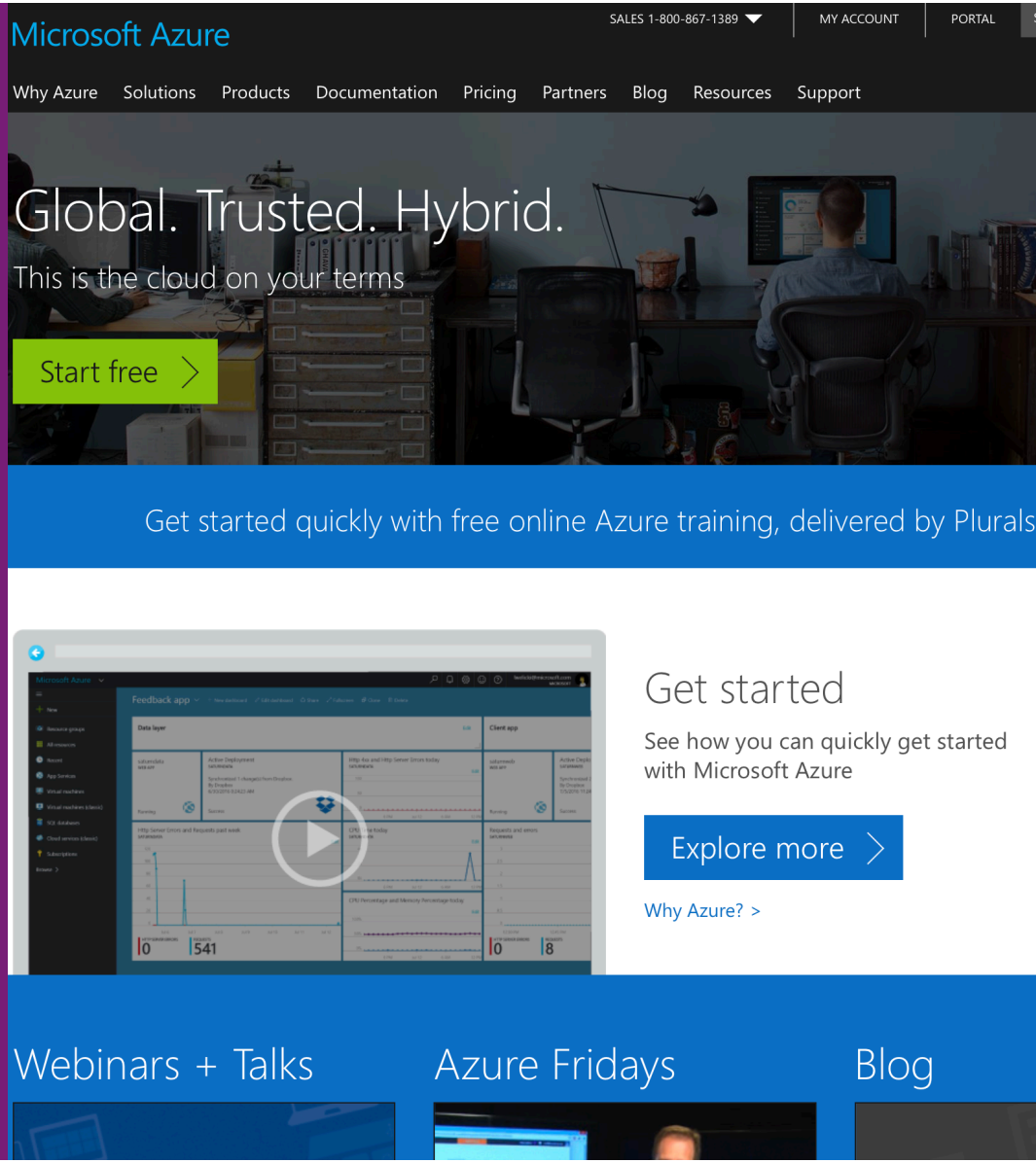
- One more big assignment (CloudKit)
- One small (swift on server)
- Case Study



CASE STUDIES

CASE STUDIES

- Teach us all something we did not discuss
  - Related to your final project?
- Present to class (or alternative)



CONTROL STUDIES

POTENTIAL TOPICS

- Up-and-computing technology
- Strategy/technique to deal with problem
- Microsoft Azure
- Compare costs of different services
- Industry survey
- Case study on company (SnapChat, Spotify, Evernote)

Using Microservices to Encode and Publish Videos at The New York Times

By FLAVIO RIBEIRO , FRANCISCO SOUZA , MAXWELL DA SILVA and THOMPSON MARZAGÃO

NOVEMBER 1, 2016 10:30 AM Comment

Email
Share
Tweet
Save
More

Video publishing at The Times is growing

For the past 10 years, the video publishing lifecycle at The New York Times has relied on vendors and in-house hardware solutions. With our growing investment in video journalism over the past couple of years, we've found ourselves producing more video content every month, along with supporting new initiatives such as 360-degree video and Virtual Reality. This growth has created the need to migrate to a video publishing platform that could adapt to, and keep up with, the fast pace that our newsroom demands and the continued evolution of our production process. Along with this, we needed a system that could continuously scale in both capacity and features while not compromising on either quality or reliability.

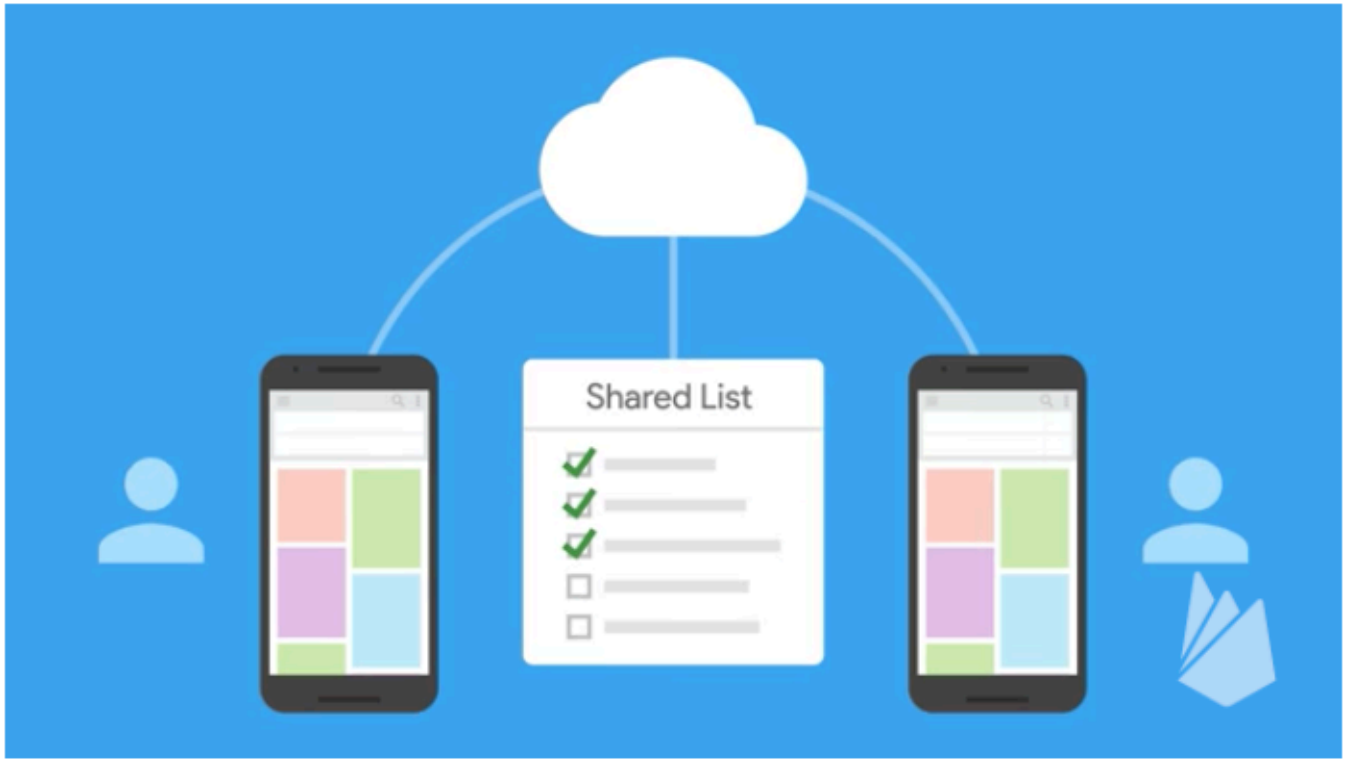
A solution

At the beginning of this year, we created a group inside our video engineering team to implement a new solution for the ingesting, encoding, publishing and the syndication of our growing library of video content. The main goal of the team was to implement a job processing pipeline that was vendor agnostic and cloud-based, along with being highly efficient, elastic, and, of course, reliable. Another goal was to make the system as easy to use as possible, removing any hurdles that might get in the way of our video producers publishing their work and distributing it to our platforms and third-party partners. To do that, we decided to leverage the

FIRESTORE

FIRESTORE

- https://youtu.be/QcsAb2RR52c?list=PLl-K7zZEsYLmOF_07IayrTntevxtbUxDL



FIRESTORE

- Strong parity with realtime database
- Better
  - Querying
  - Scaling

Flexibility	The Cloud Firestore data model supports flexible, hierarchical data structures. Store your data in documents, organized into collections. Documents can contain complex nested objects in addition to subcollections.
Expressive querying	In Cloud Firestore, you can use queries to retrieve individual, specific documents or to retrieve all the documents in a collection that match your query parameters. Your queries can include multiple, chained filters and combine filtering and sorting. They're also indexed by default, so query performance is proportional to the size of your result set, not your data set.
Realtime updates	Like Realtime Database, Cloud Firestore uses data synchronization to update data on any connected device. However, it's also designed to make simple, one-time fetch queries efficiently.
Offline support	Cloud Firestore caches data that your app is actively using, so the app can write, read, listen to, and query data even if the device is offline. When the device comes back online, Cloud Firestore synchronizes any local changes back to Cloud Firestore.
Designed to scale	Cloud Firestore brings you the best of Google Cloud Platform's powerful infrastructure: automatic multi-region data replication, strong consistency guarantees, atomic batch operations, and real transaction support. We've designed Cloud Firestore to handle the toughest database workloads from the world's biggest apps.

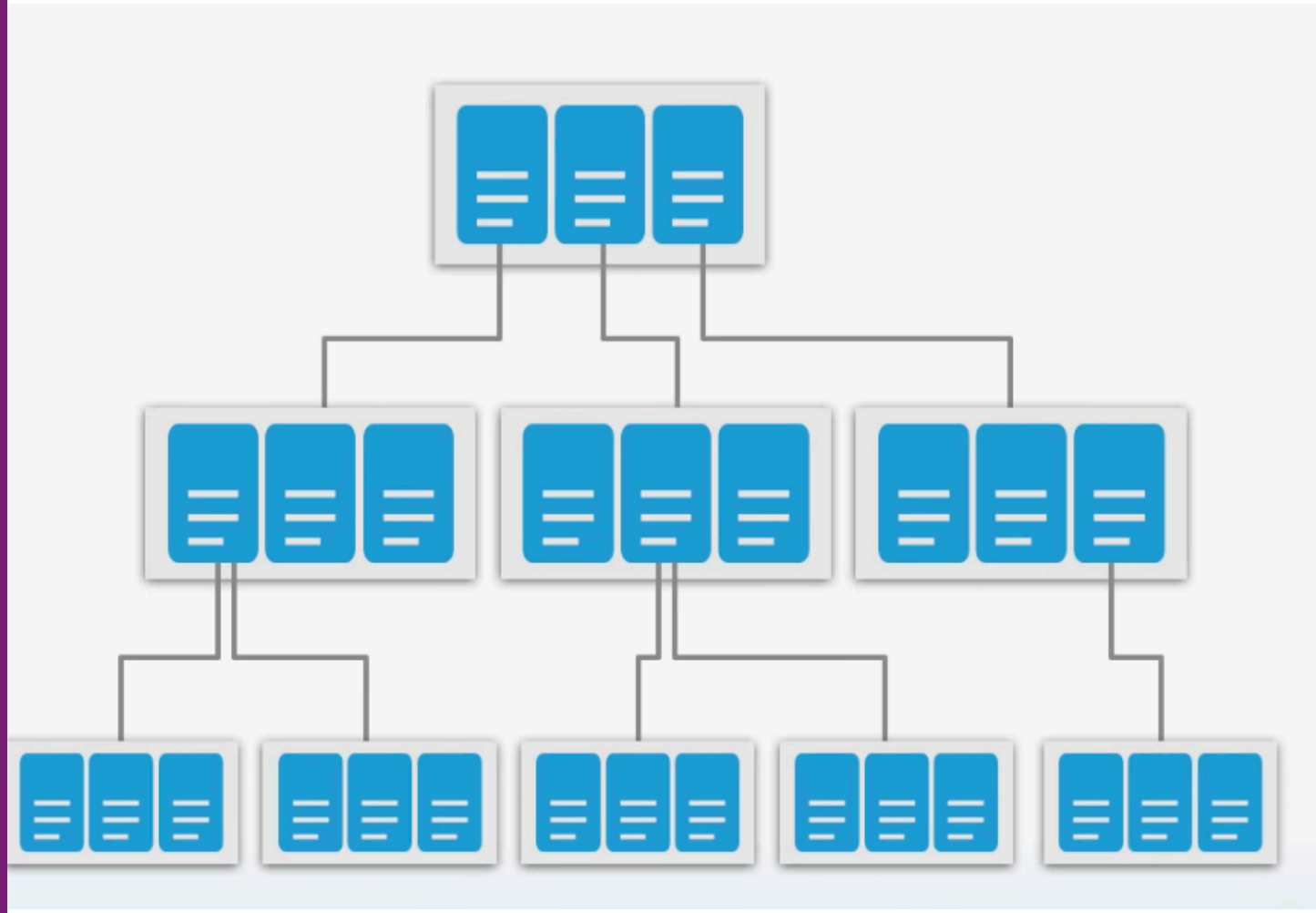
FIRESTORE

- Add to Podfile
- `pod install`

```
platform :ios, '9.0'

target 'FireChat' do
  # Comment the next line if you're not using Swift and don't want to use dynamic
      frameworks
  use_frameworks!

  # Pods for FireChat
  pod 'Firebase/Core'
  pod 'Firebase/Database'

  pod 'Firebase/Auth'
  pod 'GoogleSignIn'

  pod 'Firebase/Crash'

  pod 'Firebase/Storage'
  pod 'Firebase/Invites'
  pod 'Firebase/Messaging'
  pod 'Firebase/DynamicLinks'

  pod 'Firebase/RemoteConfig'
  pod 'Firebase/Firestore'

  pod 'SDWebImage'

end
```

FIRESTORE

- Data is stored as documents and collections



FIRESTORE

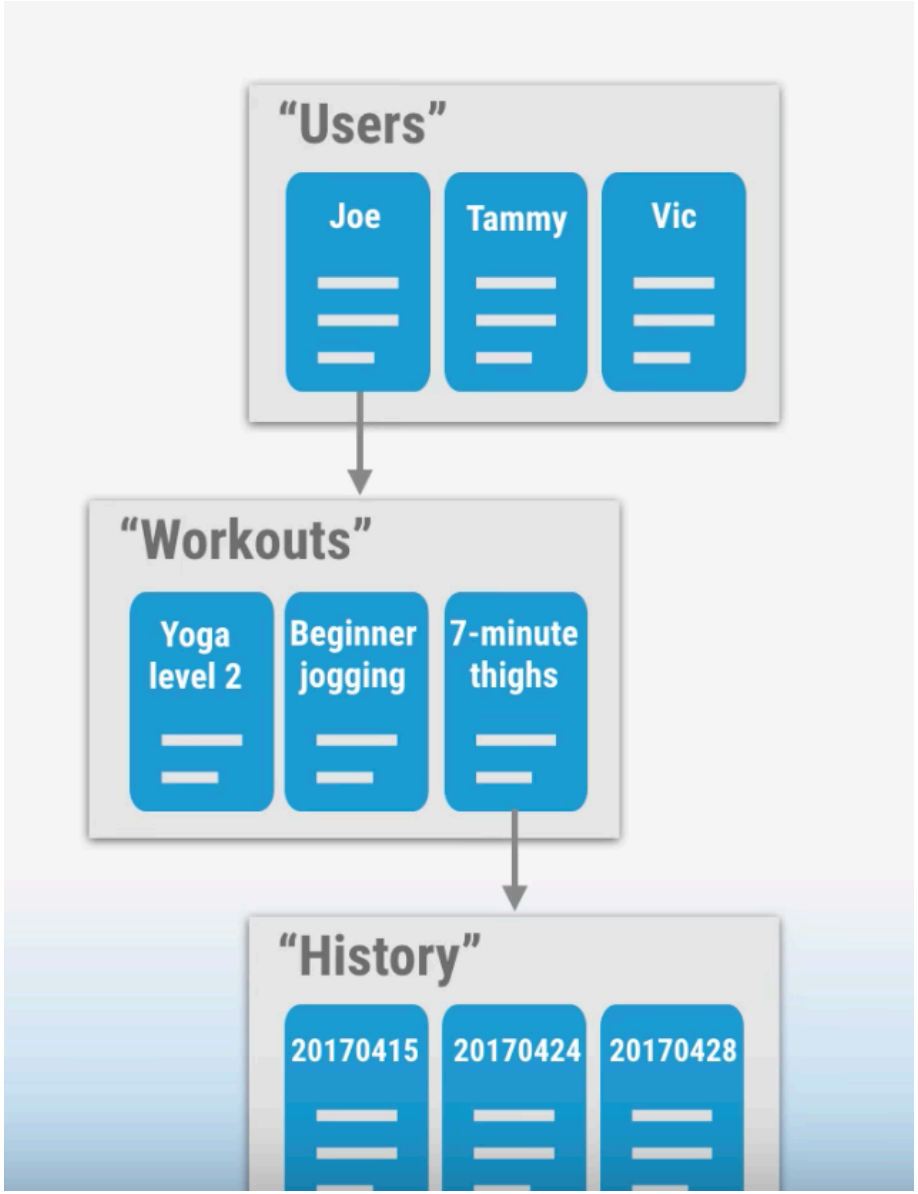
- Document is just key-value pairs

Document

```
bird_type: "swallow"
airspeed: 42.733
coconut_capacity: 0.62
isNative: false
icon: <binary data>
vector:
    {x: 36.4255,
     y: 25.1442,
     z: 18.8816}
distances_traveled:
    [42, 39, 12, 42]
```

FIRESTORE

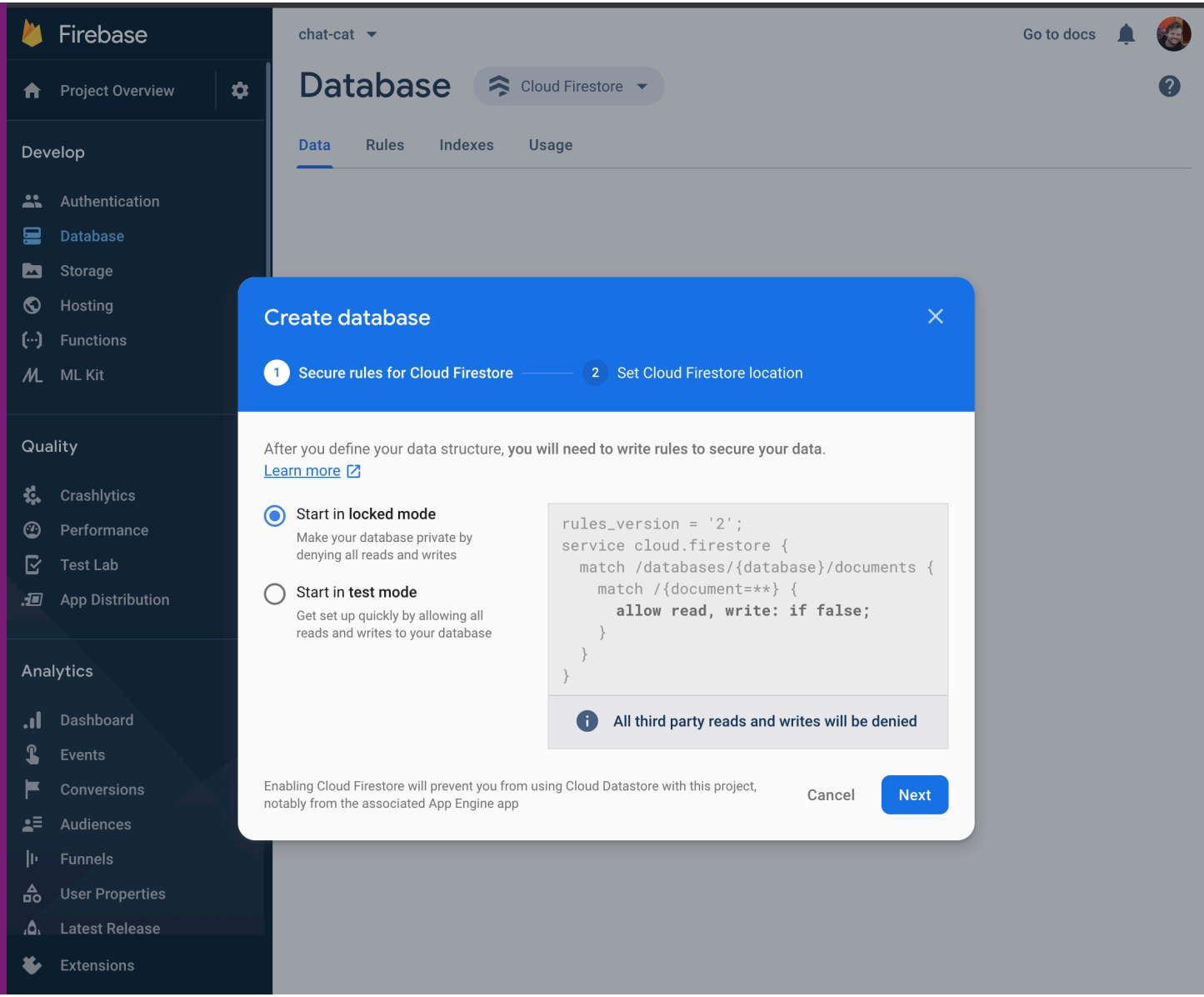
- Collections are groups of documents
- Collections can only contain documents
- Documents can point to other collections
- The Root of you application has to be a collection



USING FIRESTORE

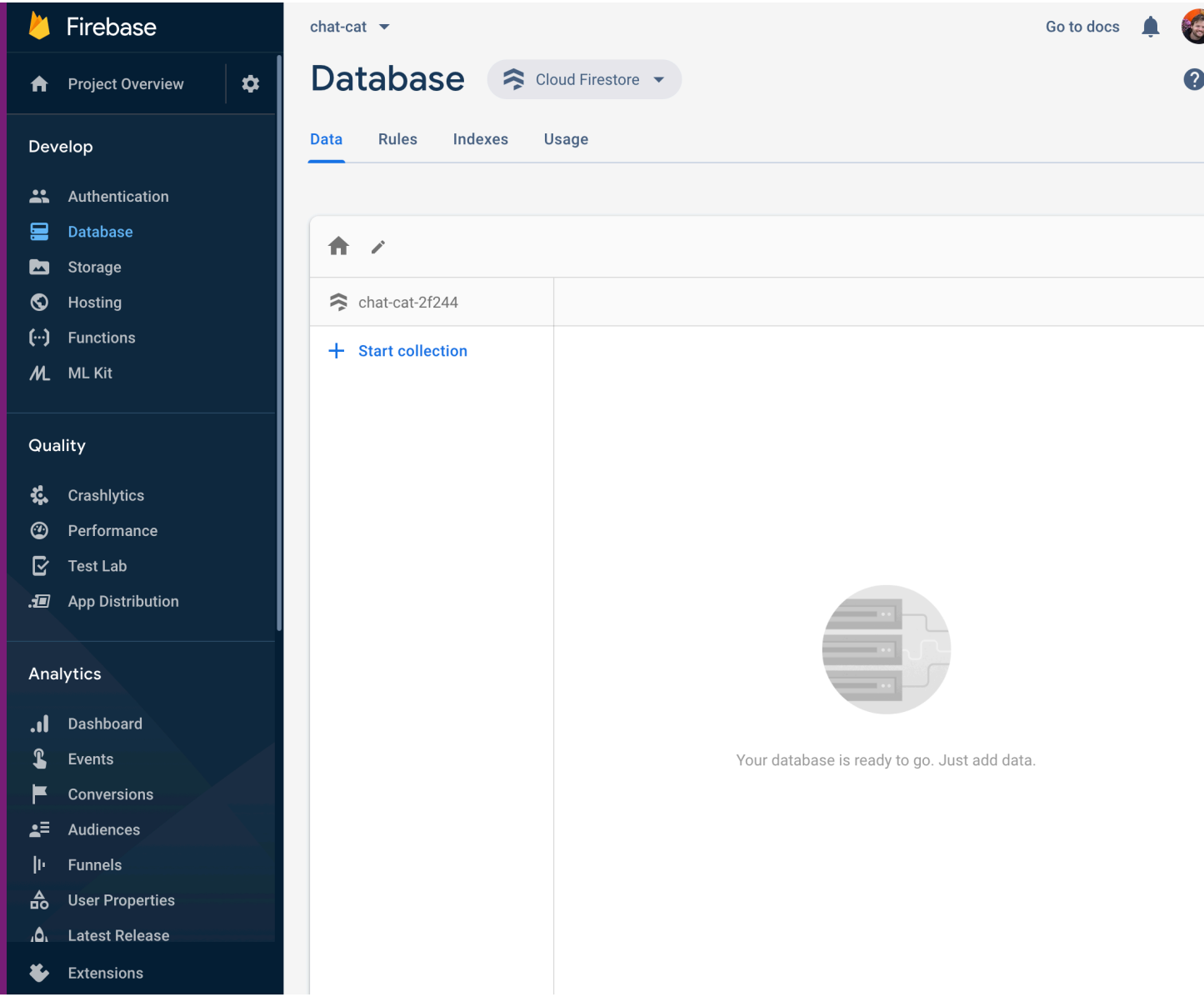
USING FIRESTORE

- Firestore can be used along side realtime database
- Security rules are private by default



USING FIRESTORE

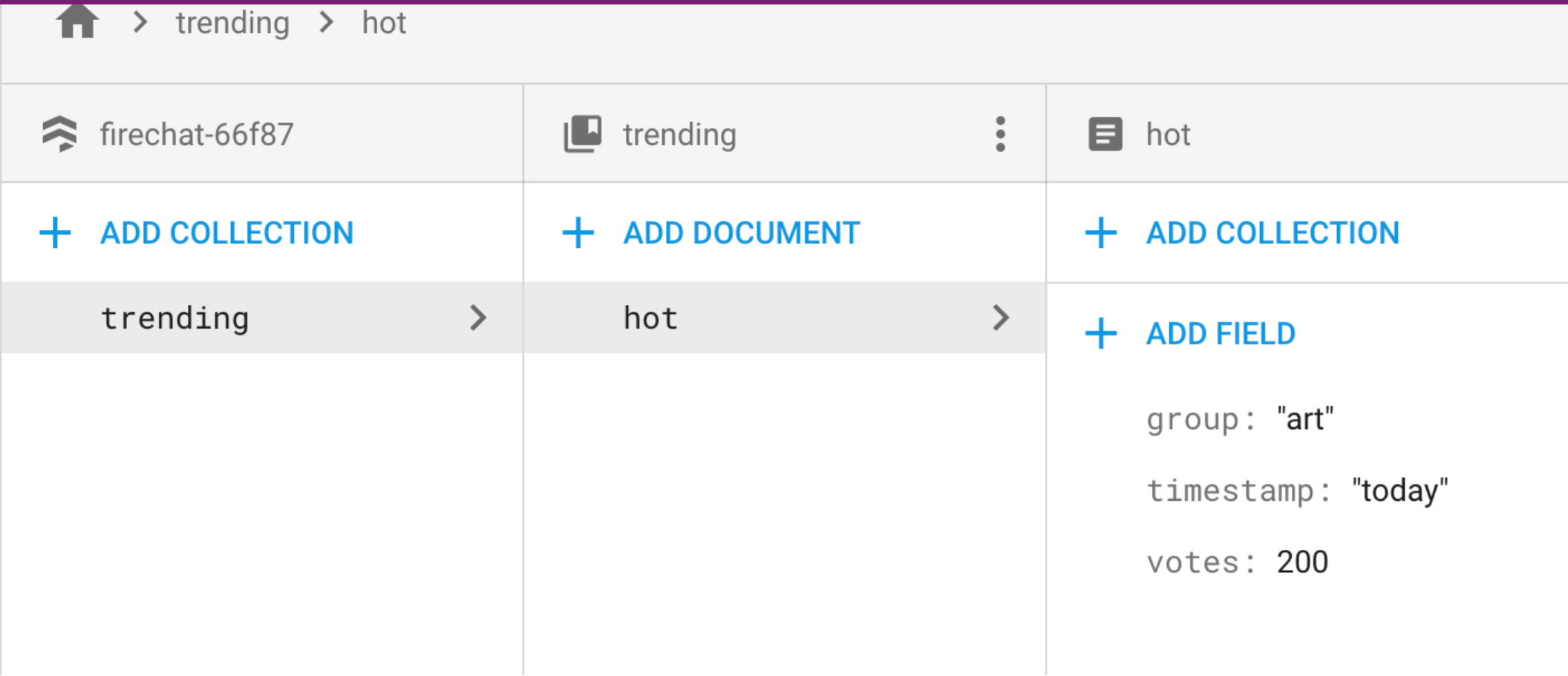
- Firestore can be used along side realtime database
- Security rules are private by default



USING FIRESTORE

```
// Firestore Ref
docRef = Firestore.firestore().document("trending/hot")

// Load some data up in Firestore
let trendingData: [String: Any] = ["group" : "art",
                                   "timestamp" : "today",
                                   "votes" : 200]

docRef.setData(trendingData) { (error) in
  if error != nil {
    print("Data!!")
  }
}
```

USING FIRESTORE



USING FIRESTORE

```
/// Get a one time snapshot
func oneTimeSnapshot() {
  docRef.getDocument { (docSnapshot, error) in
    guard let docSnapshot = docSnapshot, docSnapshot.exists else {return}
    let data = docSnapshot.data()
    self.trendingLabel.text = data["group"] as? String ?? ""
  }
}
```

USING FIRESTORE

```
/// Set up an observer
func observeDocument() {
  self.listenerRef = docRef.addSnapshotListener { (docSnapshot, error) in
    guard let docSnapshot = docSnapshot, docSnapshot.exists else {return}
    let data = docSnapshot.data()
    self.trendingLabel.text = data["group"] as? String ?? ""
  }
}
```

REMOVE IT

```
override func viewWillDisappear(_ animated: Bool) {
  super.viewWillDisappear(animated)
  self.listenerRef.remove()
}
```

USING FIRESTORE

App Engine

- Dashboard
- Services
- Versions
- Instances
- Task queues
- Security scans
- Firewall rules
- Quotas
- Blobstore
- Memcache
- Search
- Settings

Quotas

VIEW USAGE HISTORY

The quota details for this application are grouped by API and are listed below. If your application exceeds 50% of any particular quota halfway through the day, it may exceed the quota before the day is over. To learn more about how quotas work, read Understanding Quotas and Why is My App Over Quota?

Storage

Resource	Usage today	Daily quota	Per-minute quota	Rate limit status
Cloud Storage Class B Operations	0 of 0.05	0%	--	Standard rate
Cloud Storage Class A Operations	0 of 0.02	0%	--	Standard rate
Cloud Storage Standard Storage	0.00098 of 5 GB	0%	--	Standard rate
Cloud Storage Network (Egress) - Americas and EMEA	0.000001 of 1 GB	0%	--	Standard rate
Firestore Read Operations	0.000023 of 0.05 Million Ops	0%	--	Standard rate
Firestore Small Operations	0.000003 of 0.05 Million Ops	0%	--	Standard rate
Firestore API Calls	81	--	--	Standard rate
Data Sent to Firestore API	0.000001 GB	--	--	Standard rate
Firestore Entity Fetch Ops	12	--	--	Standard rate

REALTIME DATABASE VS. FIRESTORE

REALTIME DATABASE VS. FIRESTORE

Realtime Database is Firebase's original database. It's an efficient, low-latency solution for mobile apps that require synced states across clients in realtime.

Cloud Firestore is Firebase's new flagship database for mobile app development. It improves on the successes of the Realtime Database with a new, more intuitive data model. Cloud Firestore also features richer, faster queries and scales better than the Realtime Database.

- Flagship...gives it all away

REALTIME DATABASE VS. FIRESTORE

Realtime Database	Cloud Firestore
Stores data as one large JSON tree. • Simple data is very easy to store. • Complex, hierarchical data is harder to organize at scale. Learn more about the Realtime Database data model.	**Stores data in documents organized in collections.** • Simple data is easy to store in documents, which are very similar to JSON. • Complex, hierarchical data is easier to organize at scale, using subcollections within documents. • Requires less denormalization and data flattening. Learn more about the Cloud Firestore data model.

REALTIME DATABASE VS. FIRESTORE

Realtime Database	Cloud Firestore
Deep queries with limited sorting and filtering functionality. • You can only sort *or* filter on a property, not sort *and* filter on a property, in a single query. • Queries are deep by default: They always return the entire subtree.	**Indexed queries with compound sorting and filtering.** • You can chain filters and combine filtering and sorting on a property in a single query. • Write shallow queries for subcollections: You can query subcollections within a document instead of an entire collection, or even an entire document. • Queries are indexed by default: Query performance is proportional to the size of your result set, not your data set.

- Major limitation of Realtime DB

REALTIME DATABASE VS. FIRESTORE

- Firestore
  - Better choice for complex data modeling
  - Could be more expensive
- Realtime
  - Limited scalablity
  - Tree can be difficult to manage
  - Data structure is simple

ML KIT

ML KIT

- Brings Google machine learning systems to firebase
- Expertise not required
  - From a blackbox to TensorFlow

Firebase > Docs > Guides

☆☆☆☆☆

ML Kit for Firebase

iOS

Use machine learning in your apps to solve real-world problems.

ML Kit is a mobile SDK that brings Google's machine learning expertise to Android and iOS apps in a powerful yet easy-to-use package. Whether you're new or experienced in machine learning, you can implement the functionality you need in just a few lines of code. There's no need to have deep knowledge of neural networks or model optimization to get started. On the other hand, if you are an experienced ML developer, ML Kit provides convenient APIs that help you use your custom TensorFlow Lite models in your mobile apps.

★ This is a beta release of ML Kit for Firebase. This API might be changed in backward-incompatible ways and is not subject to any SLA or deprecation policy.

Key capabilities

Production-ready for common use cases	ML Kit comes with a set of ready-to-use APIs for common mobile use cases: recognizing text, detecting faces, identifying landmarks, scanning barcodes, labeling images, and identifying the language of text. Simply pass in data to the ML Kit library and it gives you the information you need.

ML KIT

- Simplifies using the most common features of machine learning
  - What you would want for mobile applications

Firebase > Docs > Guides

ML Kit for Firebase

iOS

Use machine learning in your apps to solve real-world problems.

ML Kit is a mobile SDK that brings Google's machine learning expertise to Android and iOS apps in a powerful yet easy-to-use package. Whether you're new or experienced in machine learning, you can implement the functionality you need in just a few lines of code. There's no need to have deep knowledge of neural networks or model optimization to get started. On the other hand, if you are an experienced ML developer, ML Kit provides convenient APIs that help you use your custom TensorFlow Lite models in your mobile apps.

★ This is a beta release of ML Kit for Firebase. This API might be changed in backward-incompatible ways and is not subject to any SLA or deprecation policy.

Key capabilities

Production-ready for common use cases	ML Kit comes with a set of ready-to-use APIs for common mobile use cases: recognizing text, detecting faces, identifying landmarks, scanning barcodes, labeling images, and identifying the language of text. Simply pass in data to the ML Kit library and it gives you the information you need.

ML KIT

- Runs on the device or in the cloud
  - No network connection needed
  - Huge download

Firebase > Docs > Guides

ML Kit for Firebase

iOS

Use machine learning in your apps to solve real-world problems.

ML Kit is a mobile SDK that brings Google's machine learning expertise to Android and iOS apps in a powerful yet easy-to-use package. Whether you're new or experienced in machine learning, you can implement the functionality you need in just a few lines of code. There's no need to have deep knowledge of neural networks or model optimization to get started. On the other hand, if you are an experienced ML developer, ML Kit provides convenient APIs that help you use your custom TensorFlow Lite models in your mobile apps.

★ This is a beta release of ML Kit for Firebase. This API might be changed in backward-incompatible ways and is not subject to any SLA or deprecation policy.

Key capabilities

Production-ready for common use cases	ML Kit comes with a set of ready-to-use APIs for common mobile use cases: recognizing text, detecting faces, identifying landmarks, scanning barcodes, labeling images, and identifying the language of text. Simply pass in data to the ML Kit library and it gives you the information you need.

ML KIT

- Currently supported features
- Advanced features to train you own models

DOES NOT CLARIFY HOW THE RESULTS WOULD DIFFER

Feature	On-device	Cloud
Text recognition	✓	✓
Face detection	✓	
Barcode scanning	✓	
Image labeling	✓	✓
Object detection & tracking	✓	
Landmark recognition		✓
Language identification	✓	
Translation	✓	
Smart Reply	✓	
AutoML model inference	✓	
Custom model inference	✓	

ML KIT

- Currently supported features
- Advanced features to train you own models

Feature	On-device	Cloud
Text recognition	✓	✓
Face detection	✓	
Barcode scanning	✓	
Image labeling	✓	✓
Object detection & tracking	✓	
Landmark recognition		✓
Language identification	✓	
Translation	✓	
Smart Reply	✓	
AutoML model inference	✓	
Custom model inference	✓	

ML KIT

- Steps to implement MLKit in your application

1	Integrate the SDK	Quickly include the SDK using Gradle or CocoaPods.
2	Prepare input data	For example, if you're using a vision feature, capture an image from the camera and generate the necessary metadata such as image rotation, or prompt the user to select a photo from their gallery.
3	Apply the ML model to your data	By applying the ML model to your data, you generate insights such as the emotional state of detected faces or the objects and concepts that were recognized in the image, depending on the feature you used. Use these insights to power features in your app like photo embellishment, automatic metadata generation, or whatever else you can imagine.

IMAGE LABELS

ML KIT

IMAGE LABELS

- Label images using Google Vision API
  - Full
  - Subset

Image Labeling

iOS

With ML Kit's image labeling APIs, you can recognize entities in an image without having to provide any additional contextual metadata, using either an on-device API or a cloud-based API.

Image labeling gives you insight into the content of images. When you use the API, you get a list of the entities that were recognized: people, things, places, activities, and so on. Each label found comes with a score that indicates the confidence the ML model has in its relevance. With this in tasks such as automatic metadata generation and content moderation.

iOS Android

ML KIT

IMAGE LABELS

- Supports cloud and on device labelling
- iOS provides API for on device (you provide the models)

	On-device	Cloud
Pricing	Free	Free for first 1000 uses of this feature per month: see Pricing
Label coverage	400+ labels that cover the most commonly-found concepts in photos. See below.	10,000+ labels in many categories. See below. Also, try the Cloud Vision API demo to see what labels can be found for an image you provide.
Knowledge Graph entity ID support	✓	✓

ML KIT

IMAGE LABELS

- More labels and categorization in the cloud version
- Cloud version requires paid plan

Category	Example labels

People	`Crowd`
	`Selfie`
	`Smile`

Activities	`Dancing`
	`Eating`
	`Surfing`

Things	`Car`
	`Piano`
	`Receipt`

DEVICE

CLOUD

Category	Example labels	Category	Example labels

Arts & entertainment	`Sculpture`
	`Musical Instrument`
	`Dance`
		Astronomical objects	`Comet`
			`Galaxy`
			`Star`

Business & industrial	`Restaurant`
	`Factory`
	`Airline`
		Colors	`Red`
			`Green`
			`Blue`

Design	`Floral`
	`Pattern`
	`Wood Stain`
		Drink	`Coffee`
			`Tea`
			`Milk`

ML KIT

IMAGE LABELS

- Comparison example

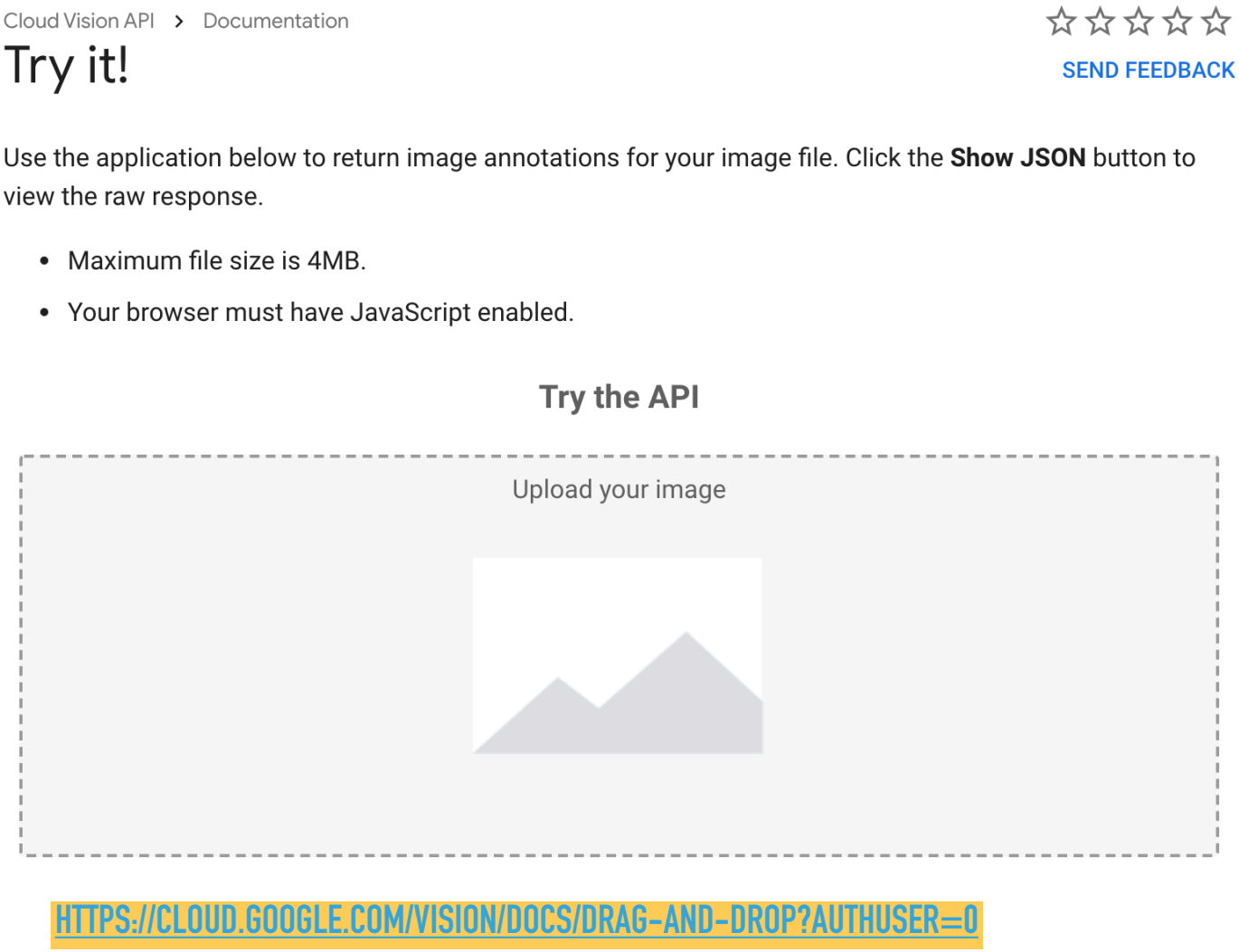
Photo: Clément Bucco-Lechat / Wikimedia Commons / CC BY-SA 3.0

On-device		Cloud	
Description	Stadium	**Description**	sport venue
Knowledge Graph entity ID	/m/019cfy	**Knowledge Graph entity ID**	/m/0bmgjqz
Confidence	0.9205354	**Confidence**	0.9860726
Description	Sports	**Description**	player
Knowledge Graph entity ID	/m/06ntj	**Knowledge Graph entity ID**	/m/02vzx9
Confidence	0.7531109	**Confidence**	0.9797604

ML KIT

IMAGE LABELS

- Cloud vision api demo site



ML KIT

IMAGE LABELS

- Add the pod

```
pod 'Firebase/MLVision'


# If using the on-device API:
pod 'Firebase/MLVisionLabelModel'
```

ML KIT

IMAGE LABELS

```
let kitten = UIImage(named:"Kitten")!

let image = VisionImage(image: kitten)
let labeler = Vision.vision().onDeviceImageLabeler()
labeler.process(image) { labels, error in
  guard error == nil, let labels = labels else { return }

  // Vision label succeeded
  for label in labels {
    let labelText = label.text
    if let entityId = label.entityID,
      let confidence = label.confidence {
      print("\(labelText) - \(entityId) - \(confidence)")
    }
  }
}
```

ML KIT

IMAGE LABELS

CLOUD

```
let image = VisionImage(image: kitten)
let labeler = Vision.vision().cloudImageLabeler()
labeler.process(image) { labels, error in
  guard error == nil, let labels = labels else { return }

  // Vision label succeeded
  for label in labels {
    let labelText = label.text
    if let entityId = label.entityID,
      let confidence = label.confidence {
      print("\(labelText) - \(entityId) - \(confidence)")
    }
  }
}
```

ML KIT

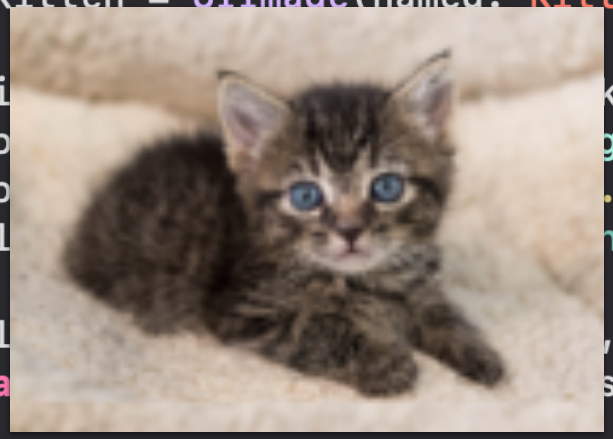
IMAGE LABELS

SET. A THRESHOLD FOR LABELS

```
let kitten = UIImage(named:"Kitten")!

let image = VisionImage(image: kitten)
let options = VisionOnDeviceImageLabelerOptions()
options.confidenceThreshold = 0.7
let labeler = Vision.vision().onDeviceImageLabeler(options: options)

labeler.process(image) { labels, error in
  guard error == nil, let labels = labels else { return }

  // Vision label succeeded
  for label in labels {
    let labelText = label.text
    if let entityId = label.entityID,
      let confidence = label.confidence {
      print("\(labelText) - \(entityId) - \(confidence)")
    }
  }
```

ML KIT

IMAGE LABELS



```
let kitten = UIImage(named:"Kitten")!

let i                                kitten)
let o                                geLabelerOptions()
optio                                .7
let l                                nDeviceImageLabeler(options: options)

labe                                 , error in
  gua                                s = labels else { return }

  // Vision label succeeded
  for label in labels {
    let labelText = label.text
    if let entityId = label.entityID,
      let confidence = label.confidence {
      print("\(labelText) - \(entityId) - \(confidence)")
    }
  }
```

EXAMPLE OUTPUT

```
Cat - /m/01yrx - 0.9914688
Pet - /m/068hy - 0.9326903
```

AUTOML VISION EDGE

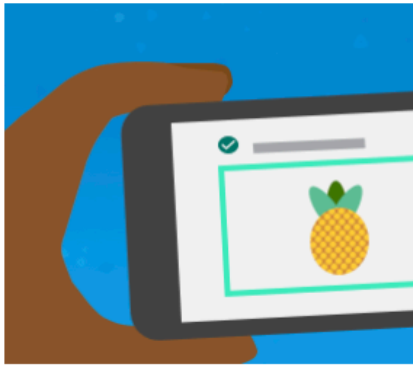
ML KIT

- Train your own models with vision edge

AutoML Vision Edge

iOS

Create custom image classification models from your own training data wi



ML Kit's base on-device image labeling API model is built for general-purpose use, and is trained to recognize around 400 categories that cover the most commonly-found concepts in photos. If you need a more specialized image labeling model, covering a narrower domain of concepts in more detail—for example, a model to distinguish between species of flowers or types of food—you can use AutoML Vision Edge to tra own images and use the model you trained instead.

Get started

ML KIT

- Identify and label your own images
- Upload them to Firebase (or Google Cloud Storage)
- Processes them to a model

```
my_training_data.zip
   |____accordion
   | |____001.jpg
   | |____002.jpg
   | |____003.jpg
   |____bass_guitar
   | |____hofner.gif
   | |____p-bass.png
   |____clavier
     |____well-tempered.jpg
     |____well-tempered (1).jp
     |____well-tempered (2).jp
```

ML KIT

ML Kit

APIs Custom AutoML New

Train high-quality custom machine learning models with minimum effort and machine learning expertise

Learn more

Your project has free usage quotas. You can create 1 dataset with a maximum of 1000 images and 3 free lifetime training hours. Upgrade to Blaze for larger datasets and more

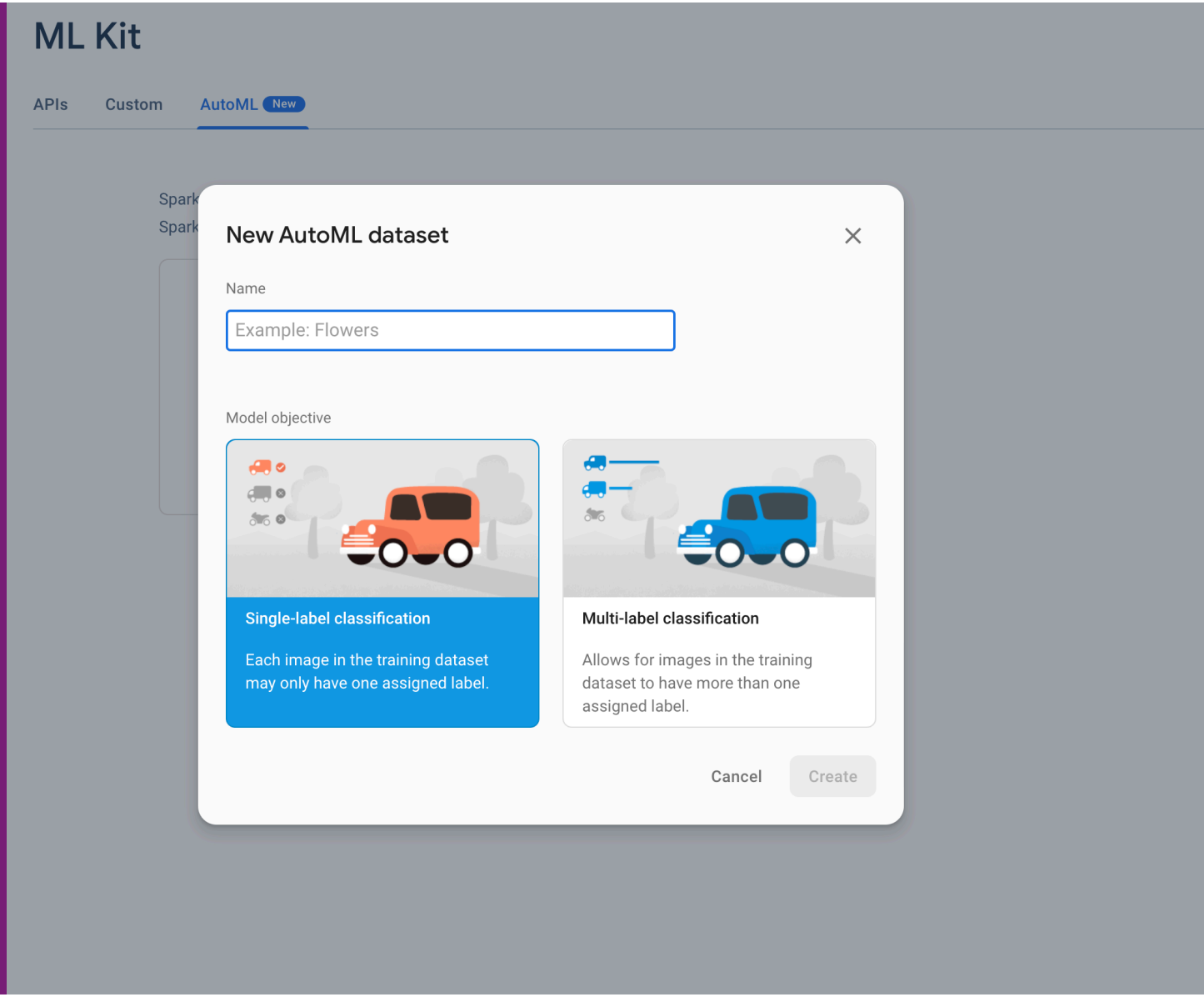
ML KIT

- Processes them to train a model
- 1 free model
  - 1000 images
  - 1 hours of processing

Typical training times	
Very small sets	1 hour
500 images	2 hours
1,000 images	3 hours
5,000 images	6 hours
10,000 images	7 hours
50,000 images	11 hours
100,000 images	13 hours
1,000,000 images	18 hours

ML KIT

- Enable your model in the console
- Real-time update



FIREBASE CLOUD FUNCTIONS

🔥FIREBASE CLOUD FUNCTIONS

- https://youtu.be/vr0Gfvp5v1A

HTTPS://YOUTU.BE/VR0GFVP5V1A

🔥FIREBASE CLOUD FUNCTIONS

- Integrates with Firebase in different ways
  - Realtime Database Triggers
  - Firebase Authentication Triggers
  - Firebase Analytics Triggers
  - Cloud Storage Triggers
  - Cloud Pub/Sub Triggers
  - HTTP Triggers

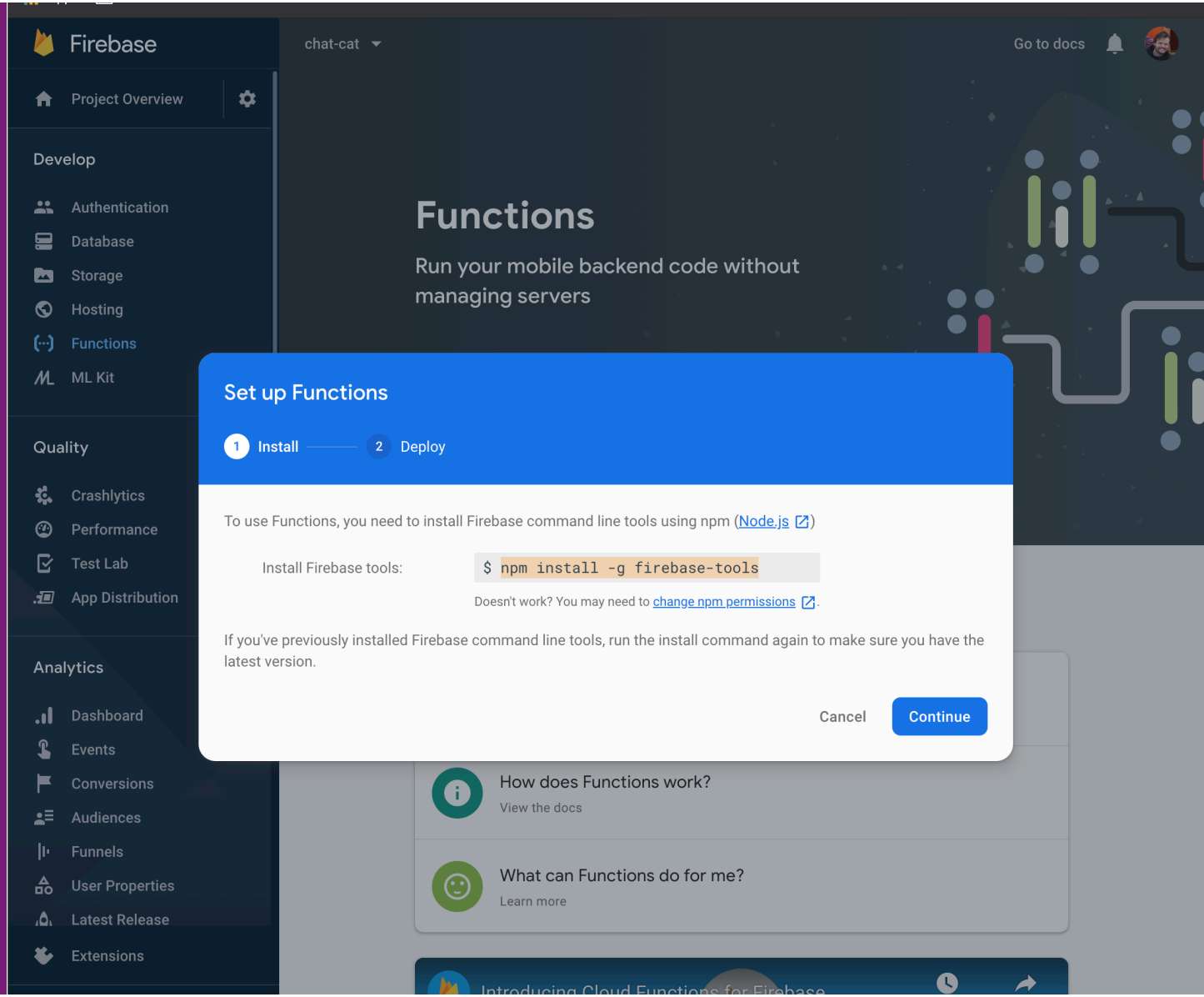
🔥FIREBASE
CLOUD FUNCTIONS

```
tabinkowski@Ts-MacBook-Pro ~
521 % brew install node
```

- Need to install node first

🔥FIREBASE CLOUD FUNCTIONS

- Use node locally to test and deploy
- Uses express framework



🔥FIREBASE
CLOUD FUNCTIONS

```
533 % firebase init functions
```

```
You're about to initialize a Firebase project in this directory:

  /Users/tabinkowski

Before we get started, keep in mind:

  * You are initializing your home directory as a Firebase project


=== Project Setup
```

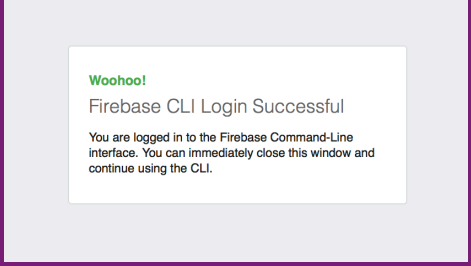
🔥FIREBASE
CLOUD FUNCTIONS

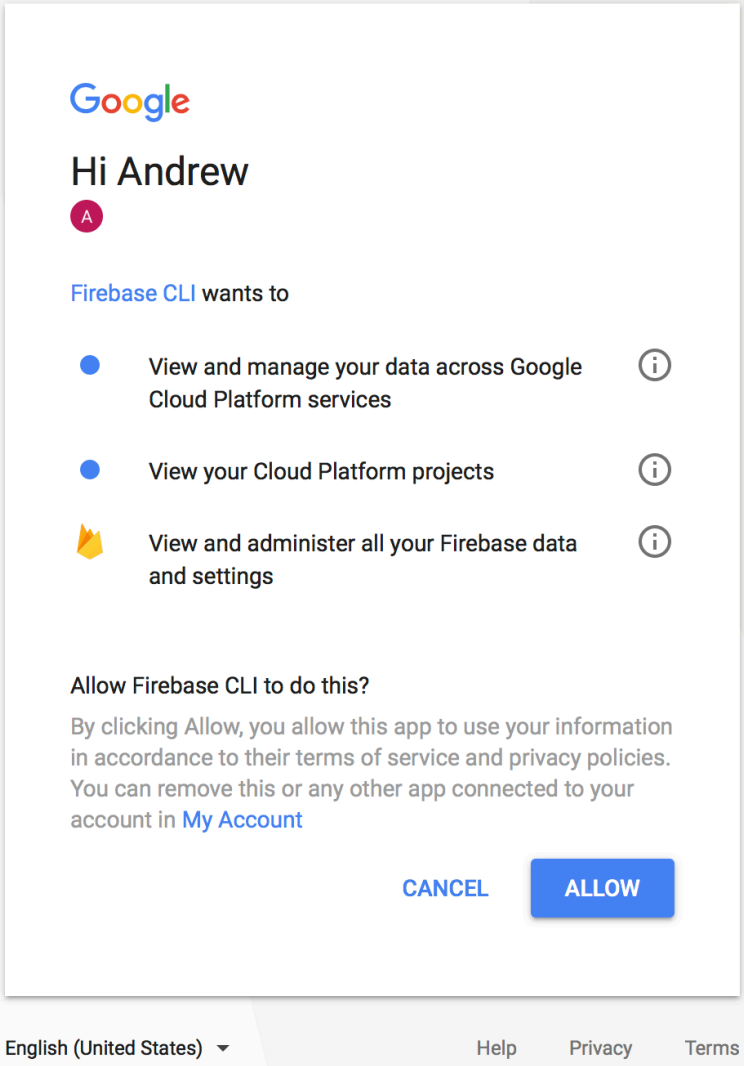
```
? Which Firebase CLI features do you want to setup for this folder? Press Space to
select features, then Enter to confirm your choices. Functions: Configure and deplo
y Cloud Functions

=== Project Setup

First, let's associate this project directory with a Firebase project.
You can create multiple project aliases by running firebase use --add,
but for now we'll just set up a default project.

? Select a default Firebase project for this directory: [create a new project]

=== Functions Setup

A functions directory will be created in your project with a Node.js
package pre-configured. Functions can be deployed with firebase deploy.

✔  Wrote functions/package.json
```

🔥FIREBASE
CLOUD FUNCTIONS

🔥FIREBASE CLOUD FUNCTIONS

- Give Firebase permissions

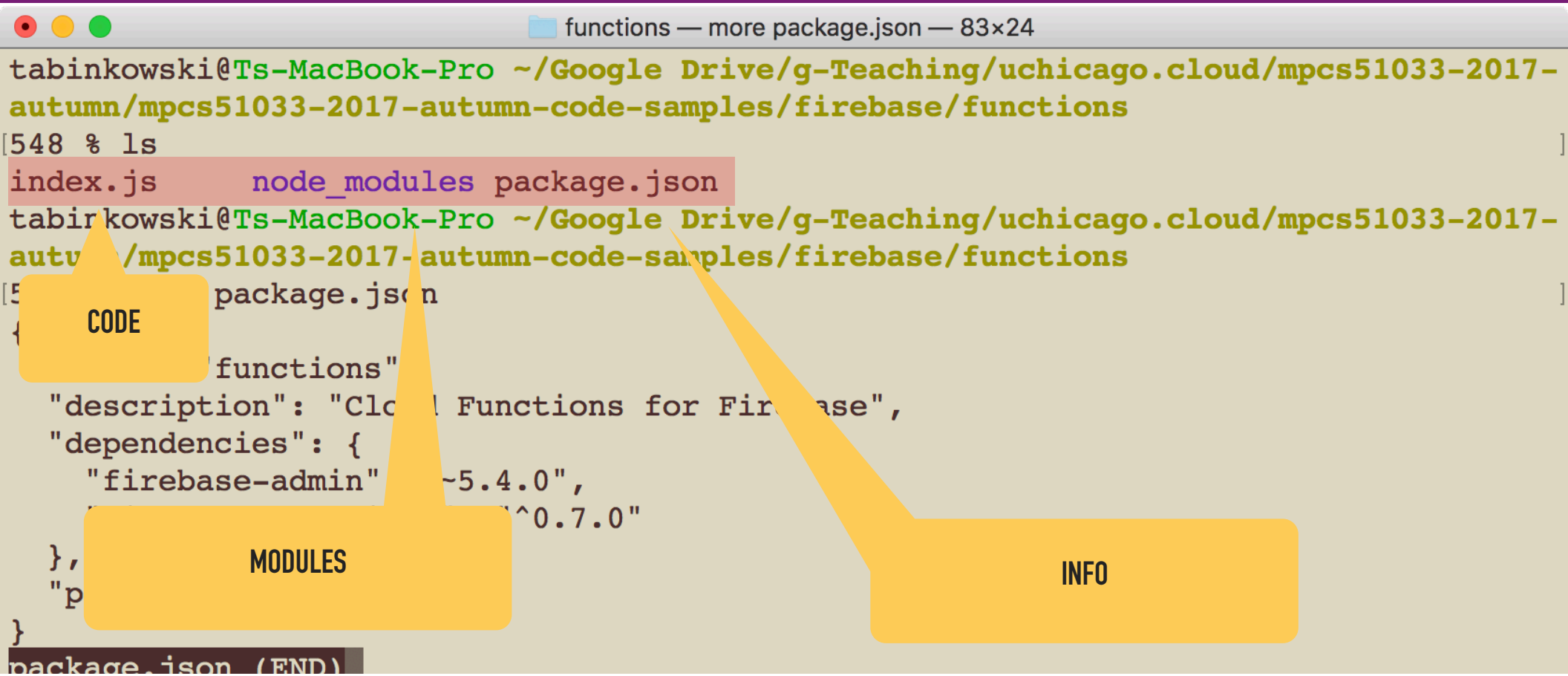




🔥FIREBASE
CLOUD FUNCTIONS

```
? Which Firebase CLI features do you want to setup for this folder? Press Space to
select features, then Enter to confirm your choices.
⚠  You have have not selected any features. Continuing will simply associate this f
older with a Firebase project. Press Ctrl + C if you want to start over.

=== Project Setup

First, let's associate this project directory with a Firebase project.
You can create multiple project aliases by running firebase use --add,
but for now we'll just set up a default project.

? Select a default Firebase project for this directory:
  [don't setup a default project]
  Firebase Demo Project (fir-demo-project)
  HuntHunt (hunthunt-77d3b)
❯ FireChat (firechat-66f87)
  ChattyCathy (chattycathy-56d4b)
  [create a new project]
```

🔥FIREBASE
CLOUD FUNCTIONS



🔥FIREBASE
CLOUD FUNCTIONS

functions — more package.json — 83×24

```
tabinkowski@Ts-MacBook-Pro ~/Google Drive/g-Teaching/uchicago.cloud/mpcs51033-2017-autumn/mpcs51033-2017-autumn-code-samples/firebase/functions
[548 % ls
index.js        node_modules    package.json
tabinkowski@Ts-MacBook-Pro ~/Google Drive/g-Teaching/uchicago.cloud/mpcs51033-2017-autumn/mpcs51033-2017-autumn-code-samples/firebase/functions
[549 % more package.json
{
  "name": "functions",
  "description": "Cloud Functions for Firebase",
  "dependencies": {
    "firebase-admin": "~5.4.0",
    "firebase-functions": "^0.7.0"
  },
  "private": true
}
package.json (END)
```

🔥FIREBASE
CLOUD FUNCTIONS

```
const functions = require('firebase-functions');

// Create and Deploy Your First Cloud Functions
// https://firebase.google.com/docs/functions/write-firebase-functions

exports.helloWorld = functions.https.onRequest((request, response) => {
  response.send("Hello from Firebase!");
});
```

DEFAULT FUNCTION 😀

🔥FIREBASE CLOUD FUNCTIONS

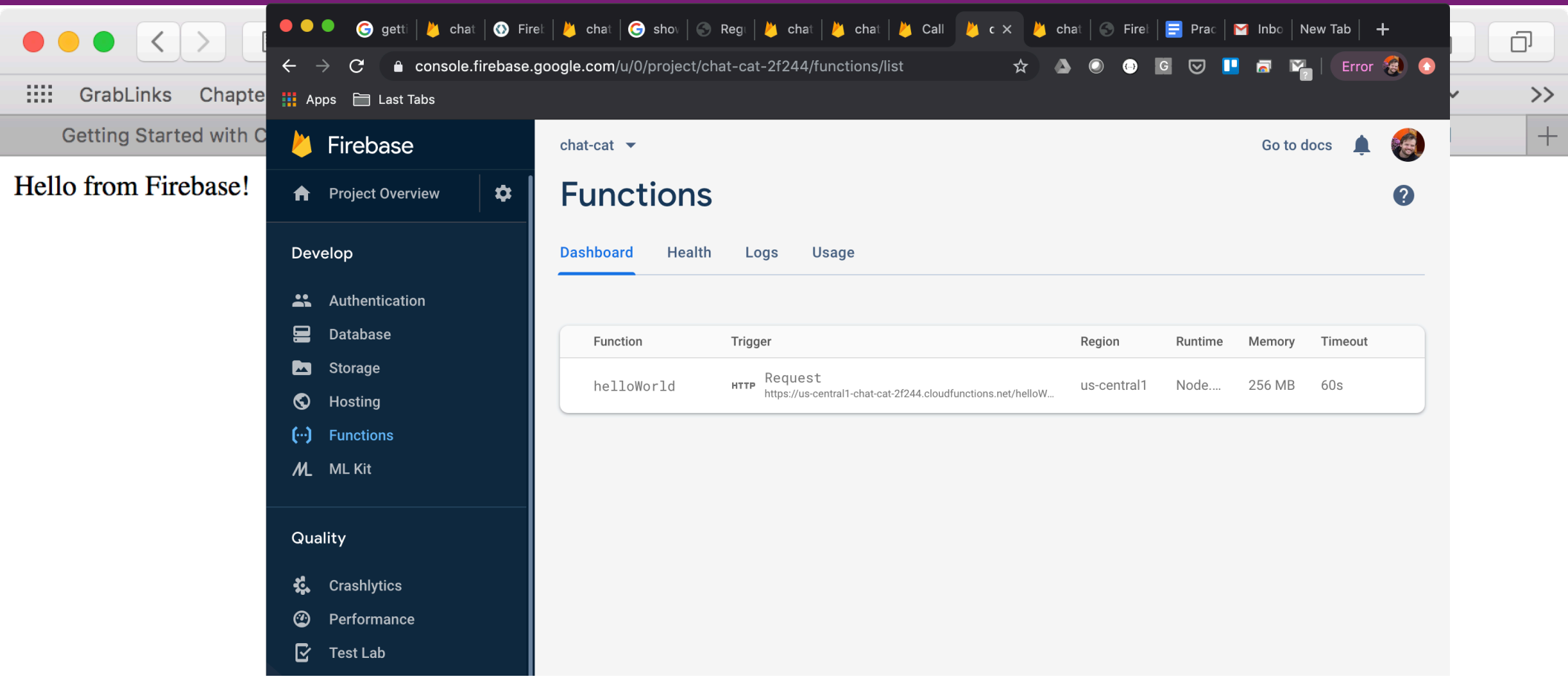
```
552 % emacs index.js
[tabinkowski@Ts-MacBook-Pro ~/Google Drive/g-Teaching/uchicago.cloud/mpcs51033-2017-
autumn/mpcs51033-2017-autumn-code-samples/firebase/functions
[553 % atom index.js                                                              ]
tabinkowski@Ts-MacBook-Pr                                  icago.cloud/mpcs51033-2017-
autumn/mpcs51033-2017-aut                                  tions
[554 % firebase deploy                                                           ]
1

=== Deploying to 'firechat-66f87'...

i  deploying functions
i  functions: ensuring necessary APIs are enabled...
i  runtimeconfig: ensuring necessary APIs are enabled...
✔  runtimeconfig: all necessary APIs are enabled
✔  functions: all necessary APIs are enabled
i  functions: preparing functions directory for uploading...
```

FIREBASE DEPLOY

🔥FIREBASE
CLOUD FUNCTIONS

```
=== Deploying to 'firechat-66f87'...

i  deploying functions
i  functions: ensuring necessary APIs are enabled...
i  runtimeconfig: ensuring necessary APIs are enabled...
✔  runtimeconfig: all necessary APIs are enabled
✔  functions: all necessary APIs are enabled
i  functions: preparing functions directory for uploading...
i  functions: packaged functions (1.14 KB) for uploading
✔  functions: functions folder uploaded successfully
i  starting release process (may take several minutes)...
i  functions: creating function helloWorld...
✔  functions[helloWorld]: Successful create operation.
✔  functions: all functions deployed successfully!

✔  Deploy complete!

Project Console: https://console.firebase.google.com/project/firechat-66f87/overview
Function URL (helloWorld): https://us-central1-firechat-66f87.cloudfunctions.net/helloWorld
tabinkowski@Ts-MacBook-Pro ~/Google Drive/g-Teaching/uchicago.cloud/mpcs51033-2017-autumn/mpcs51033-2017
-autumn-code-samples/firebase/functions
```

🔥FIREBASE
CLOUD FUNCTIONS



DATABASE TRIGGERS

DATABASE TRIGGERS

- Handle events in the database
- Make updates on functionality without changing client code

Database Triggers

With Cloud Functions, you can handle events in the Firebase Realtime Database with no need to update client code. Cloud Functions lets you run database operations with full administrative privileges, and ensures that each change to the database is processed individually. You can make Firebase Realtime Database changes via the `DataSnapshot` or via the Admin SDK.

In a typical lifecycle, a Firebase Realtime Database function does the following:

1. Waits for changes to a particular database location.
2. Triggers when an event occurs and performs its tasks (see What can I do with Cloud Functions? for examples of use cases).
3. Receives a data object that contains a snapshot of the data stored in the specified document.

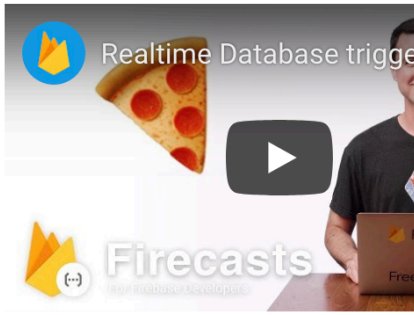
Trigger a database function

Create new functions for Realtime Database events with `functions.database`. To control when the function triggers, specify one of the event handlers, and specify the database path where it will listen for events.

DATABASE TRIGGERS

- 1. Waits for changes to a particular database location.
- 2. Triggers when an event occurs and performs its tasks (see What can I do with Cloud Functions?for examples of use cases).
- 3. Receives an event data object that contains two snapshots of the data stored at the specified path: one with the original data prior to the change, and one with the new data.

Database Triggers



With Cloud Functions, you can handle events in the Firebase Realtime Database with no need to update client code. Cloud Functions lets you run database operations with full administrative privileges, and ensures that each change to the database is processed individually. You can make Firebase Realtime Database changes via the `DataSnapshot` or via the Admin SDK.

In a typical lifecycle, a Firebase Realtime Database function does the following:

1. Waits for changes to a particular database location.
2. Triggers when an event occurs and performs its tasks (see What can I do with Cloud F for examples of use cases).
3. Receives a data object that contains a snapshot of the data stored in the specified doc

Trigger a database function

Create new functions for Realtime Database events with `functions.database`. To control function triggers, specify one of the event handlers, and specify the database path where it events.

Set the event handler

Functions let you handle database events at two levels of specificity; you can listen for spec only creation, update, or deletion events, or you can listen for any change of any kind to a pa

DATABASE TRIGGERS

`onWrite()`, which triggers when data is created, destroyed, or changed in the Realtime Database.

`onCreate()`, which triggers when new data is created in the Realtime Database.

`onUpdate()`, which triggers when data is updated in the Realtime Database.

`onDelete()`, which triggers when data is deleted from the Realtime Database.

- Event handlers for triggers

DATABASE TRIGGERS

- Triggers operate when a node path is met
- Try to use the deepest path match you can

```
# Trigger a call at /foo/bar
functions.database.ref('/foo/bar')


# Both paths will trigger the
# cloud function
/foo/bar
/foo/bar/baz/really/deep/path
```

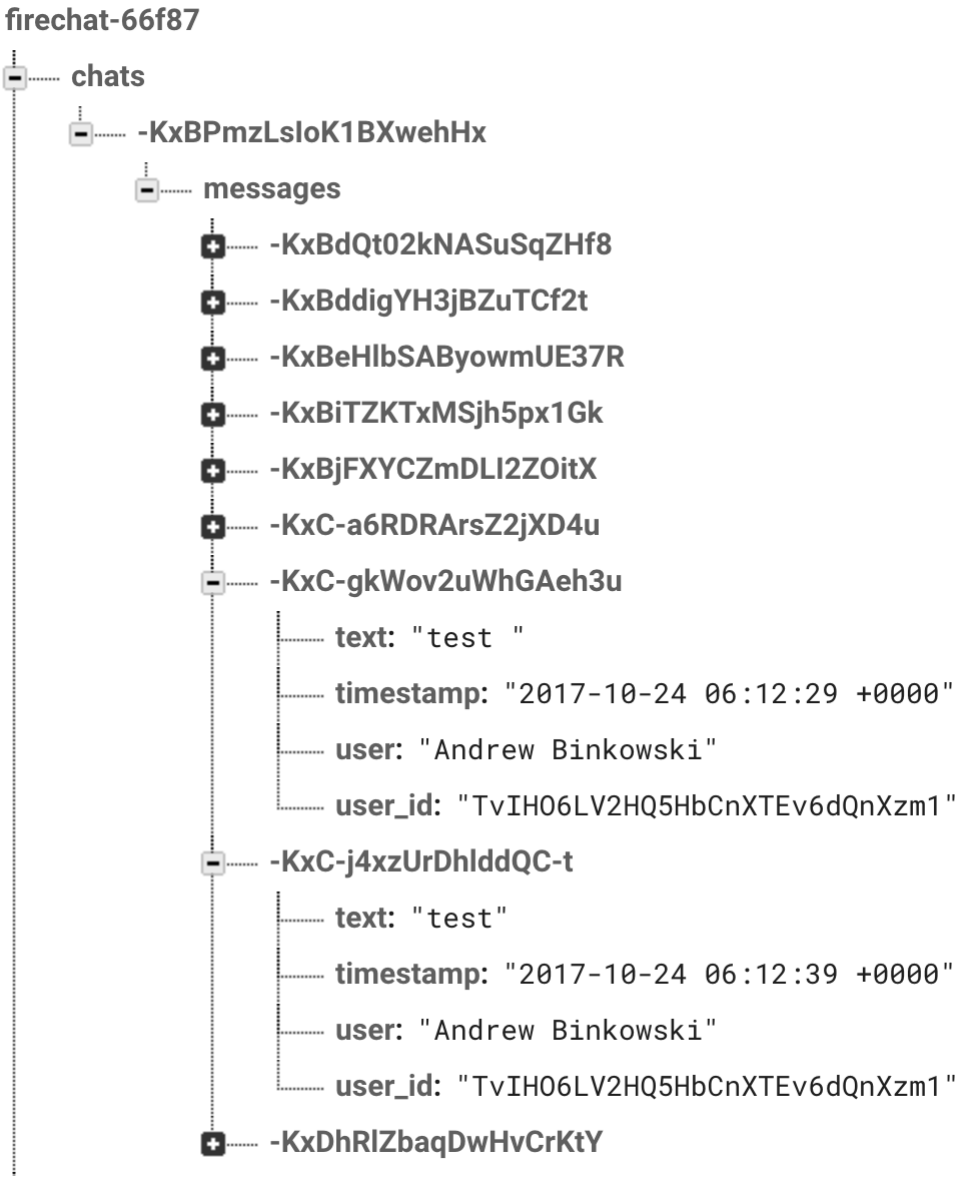
DATABASE TRIGGERS

- Use wildcards to match keys
- Path available at `event.params`

```
# Trigger a call at /foo/bar
functions.database.ref('/foo/{bar}/')


# Wildcards may be called multiple
# times depending on lovel
{
"foo": {
  "hello": "world",
  "firebase": "functions"
  }
}
```

DATABASE TRIGGERS

- Path:/chats/{chatId}/messages/{messageId}/text



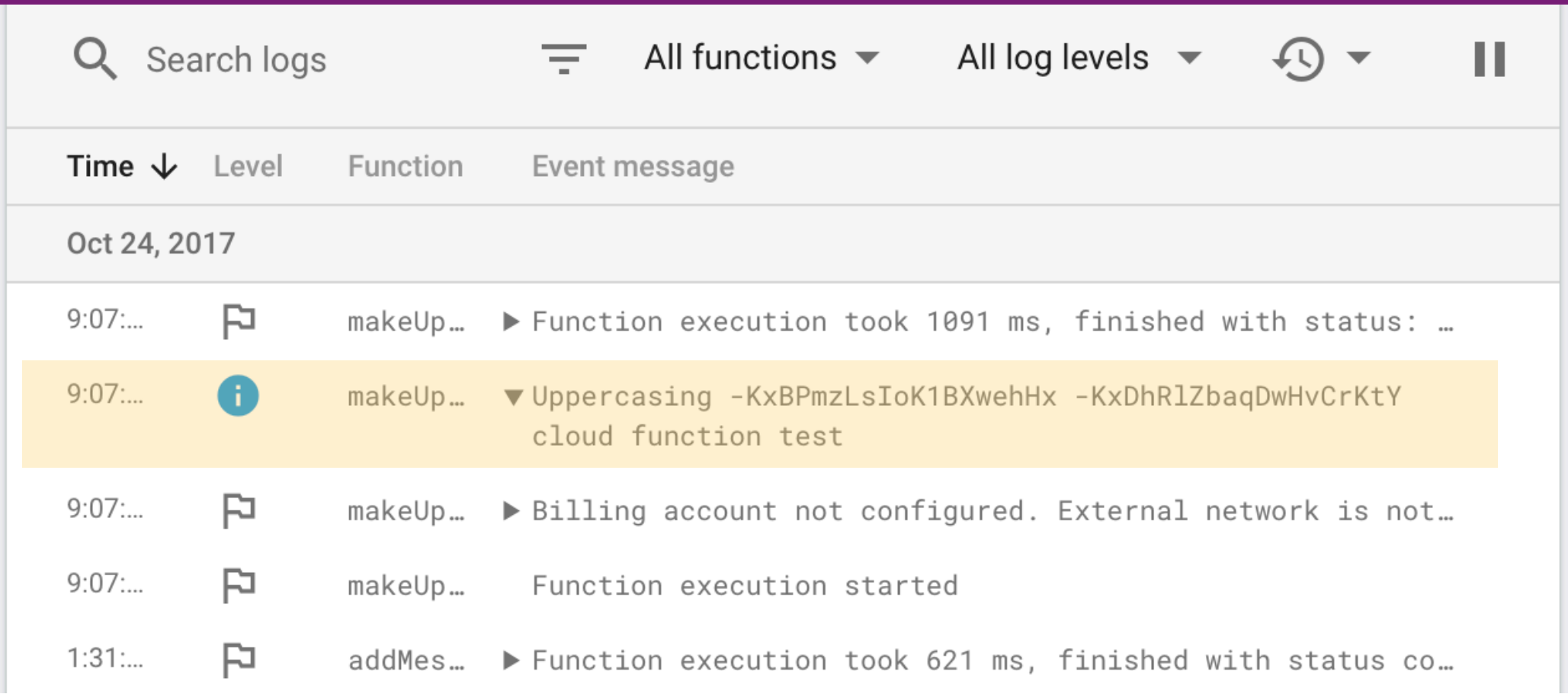
DATABASE TRIGGERS

```
exports.makeUppercase = functions.database.ref('/chats/{chatId}/messages/{messageId}/text')
    .onWrite(event => {

      // Grab the current value of what was written to the Realtime Database.
      const original = event.data.val();

      // Log out so we can debug
      console.log('Uppercasing', event.params.chatId, event.params.messageId, original);
      const uppercase = original.toUpperCase();

      // You must return a Promise when performing asynchronous tasks inside a
      // Functions such as  writing to the Firebase Realtime Database.
      // Setting an "uppercase" sibling returns a Promise
      const promise = event.data.ref.parent.child('uppercase').set(uppercase);
      return promise
    });
```

DATABASE TRIGGERS

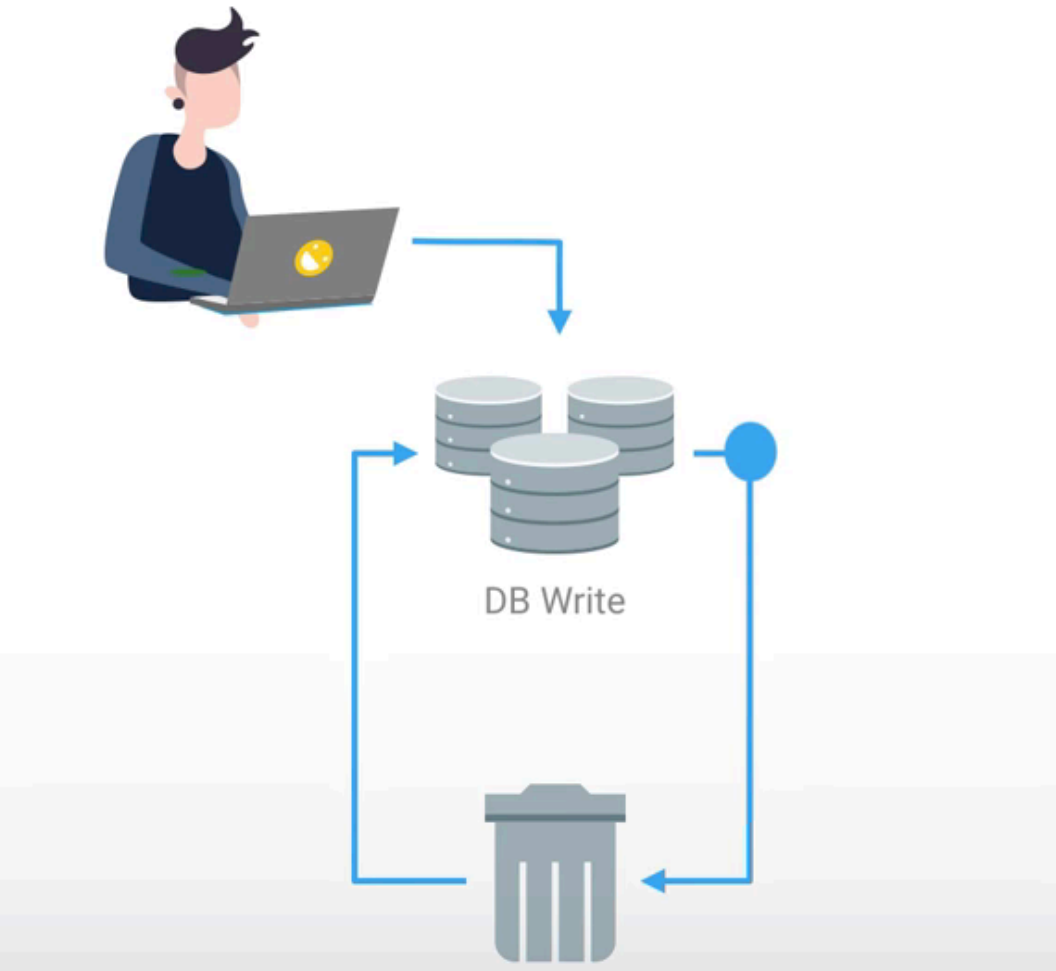
- Current structure of our data in the realtime database
- Path:/chats/{chatId}/messages/{messageId}/text

DATABASE TRIGGERS



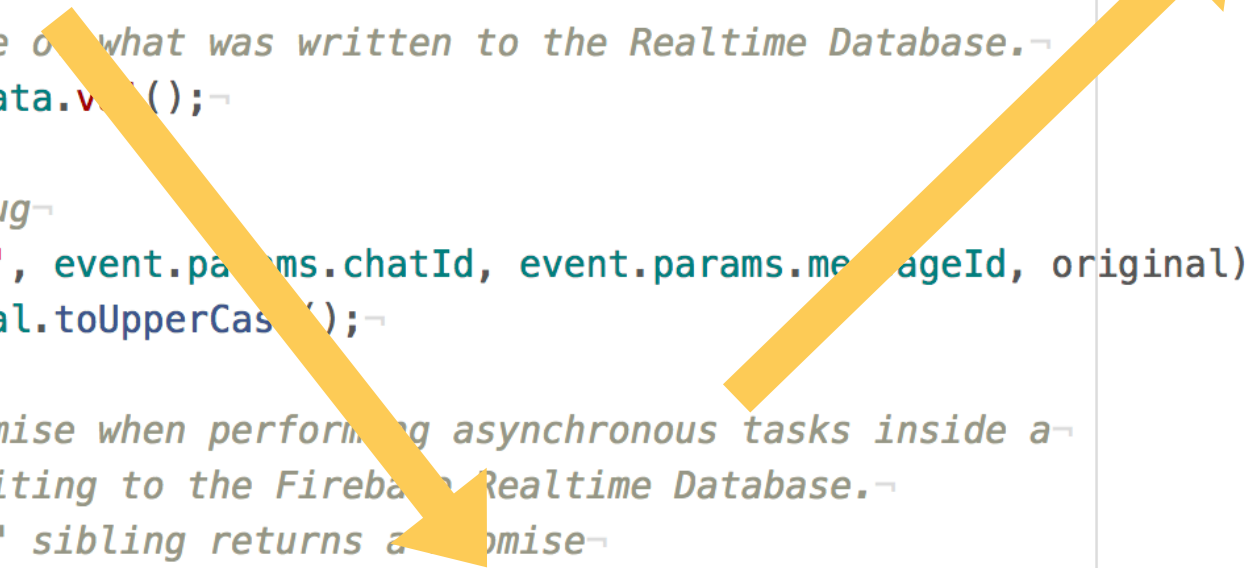
DATABASE TRIGGERS

- Careful that changes do not trigger infinite loop



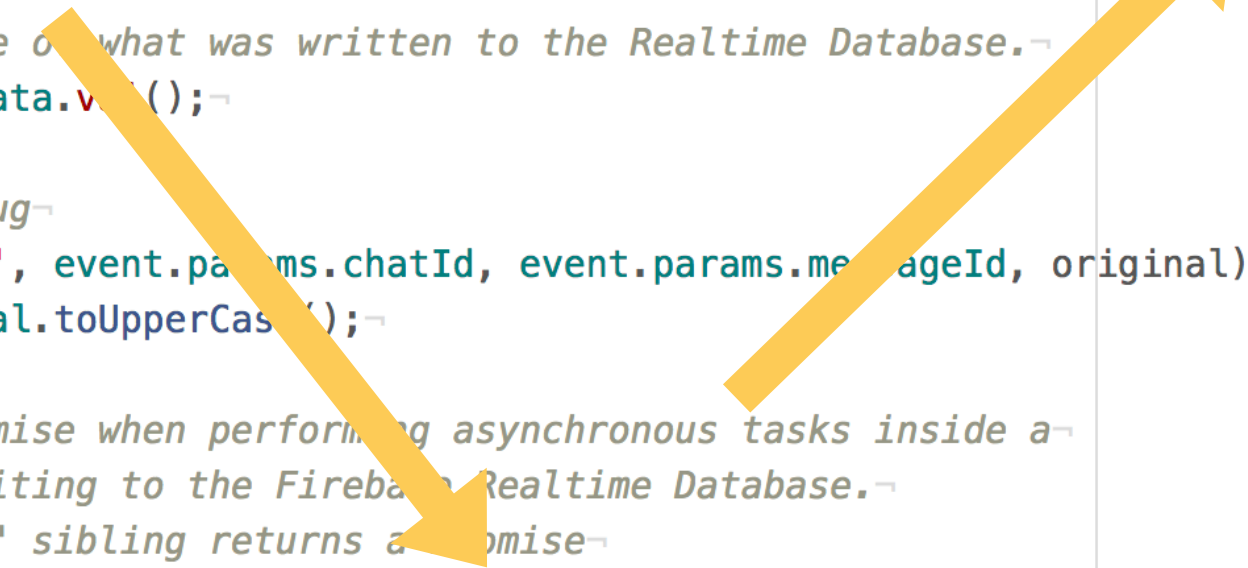
DATABASE TRIGGERS

```
exports.makeUppercase = functions.database.ref('/chats/{chatId}/messages/{messageId}/text')
    .onWrite(event => {

      // Grab the current value of what was written to the Realtime Database.
      const original = event.data.val();

      // Log out so we can debug
      console.log('Uppercasing', event.params.chatId, event.params.messageId, original);
      const uppercase = original.toUpperCase();

      // You must return a Promise when performing asynchronous tasks inside a
      // Functions such as  writing to the Firebase Realtime Database.
      // Setting an "uppercase" sibling returns a Promise
      const promise = event.data.ref.parent.child('uppercase').set(uppercase);
      return promise
    });
```



DATABASE TRIGGERS

```
exports.makeUppercase = functions.database.ref('/chats/{chatId}/messages/{messageId}/text')
    .onWrite(event => {

      // Grab the current value of what was written to the Realtime Database.
      const original = event.data.val();

      // Log out so we can debug
      console.log('Uppercasing', event.params.chatId, event.params.messageId, original);
      const uppercase = original.toUpperCase();

      // You must return a Promise when performing asynchronous tasks inside a
      // Functions such as  writing to the Firebase Realtime Database.
      // Setting an "uppercase" sibling returns a Promise
      const promise = event.data.ref.parent.child('uppercase').set(uppercase);
      return promise
    });
```



DATABASE TRIGGERS

Functions

Dashboard | Health | Logs | Usage

Function	Trigger	Region	Runtime	Memory	Timeout
date	HTTP Request https://us-central1-chat-cat-2f244.cloudfunctions.net/date	us-central1	Node.j...	256 MB	60s
helloWorld	HTTP Request https://us-central1-chat-cat-2f244.cloudfunctions.net/helloWorld	us-central1	Node.j...	256 MB	60s
makeUppercase	ref.create messages/{pushId}/text	us-central1	Node.j...	256 MB	60s
sentiment	ref.create messages/{pushId}/text	us-central1	Node.j...	256 MB	60s

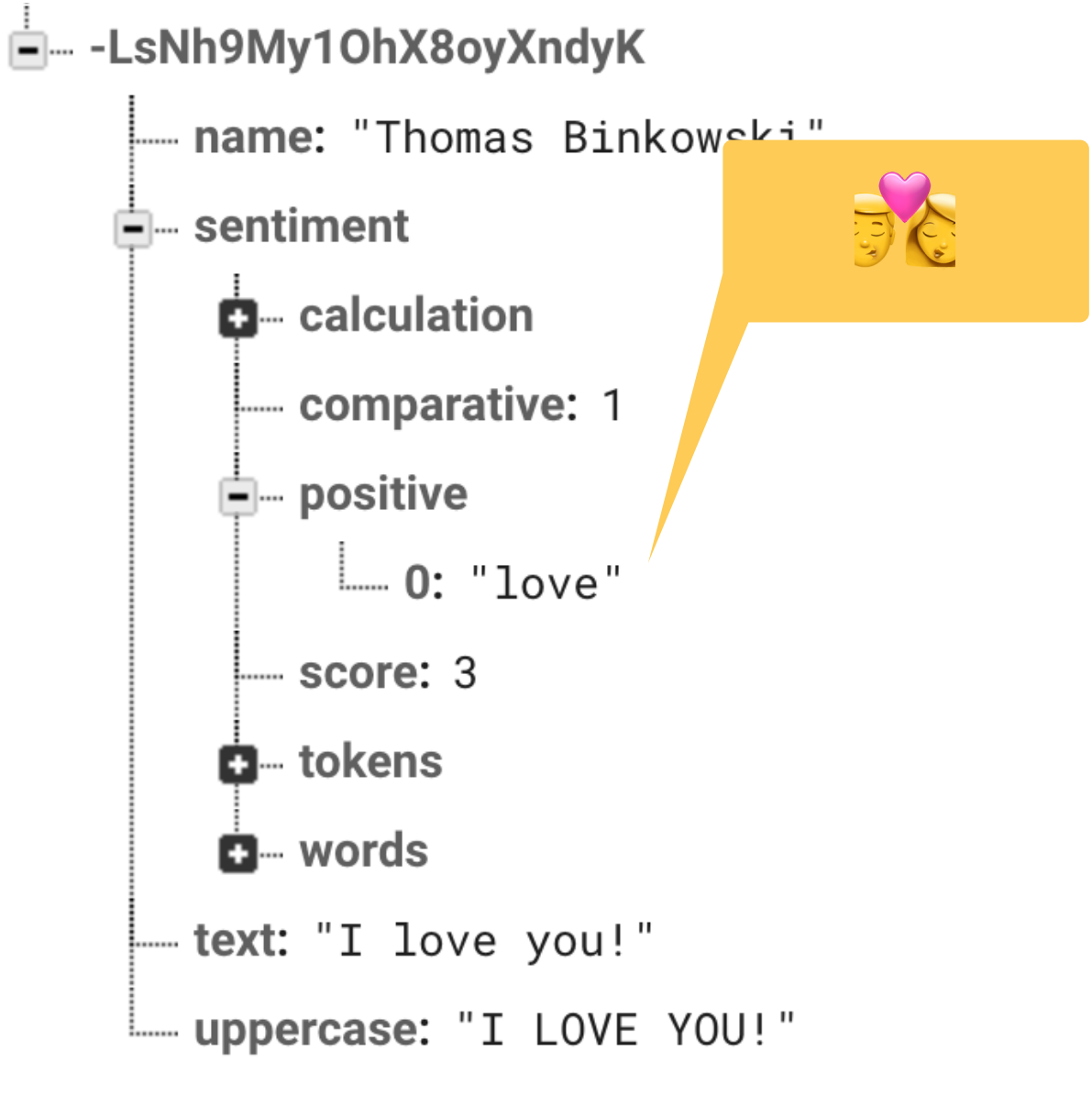
DATABASE TRIGGERS

SENTIMENT ANALYSIS

```
// Listens for new messages added to /messages/:pushId/original and creates
// uppercase version of the message to /messages/:pushId/uppercase
exports.sentiment = functions.database.ref('/messages/{pushId}/text')
.onCreate((snapshot, context) => {
  // Grab the current value of what was written to the Realtime Database.
  const original = snapshot.val();
  console.log('Uppercasing', context.params.pushId, original);

  var sentiment = new Sentiment();
  var result = sentiment.analyze(original);
  return snapshot.ref.parent.child('sentiment').set(result);
});
```

DATABASE TRIGGERS

- Updates the database with new fields



HTTP TRIGGERS

HTTP TRIGGERS

- HTTP Trigger
- Creates an API for your data

GET REQUES RETURN JSON DATA

```
exports.simplehttp = functions.https.onRequest((req, res) => {
  // Forbidding PUT requests.
  if (req.method === 'PUT') {
    return res.status(403).send('Forbidden!');
  }


  var ref = admin.database().ref('messages')

  admin.database().ref().once('value', (data) => {
      console.log(JSON.stringify(data.val()));
      res.status(200).send(JSON.stringify(data.val()));
  });
```

HTTP TRIGGERS

- HTTP Trigger
- Creates an API for your data

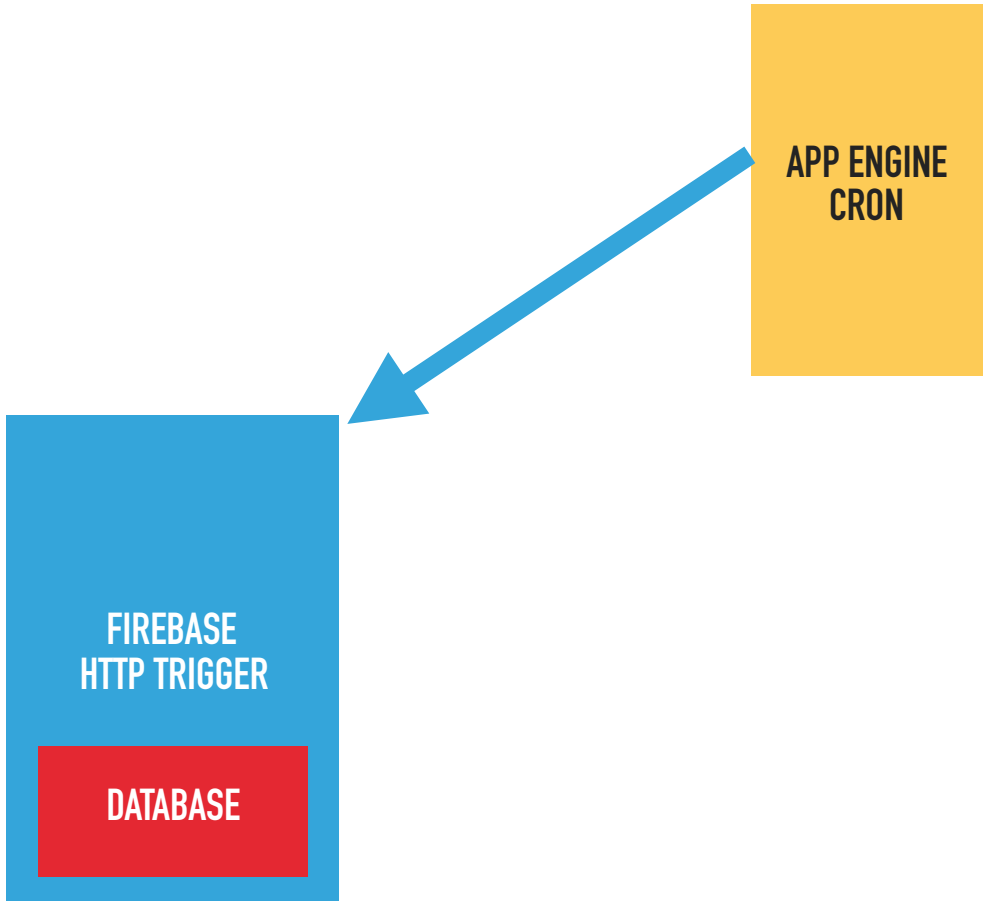
DATABASE TRIGGERS

- Authentication trigger
- Welcome email via gmail API

```
// Adds a message that welcomes new users
exports.addWelcomeMessages = functions.auth.user().onDelete(event => {
  const user = event.data; // The Firebase user.
  const email = user.email; // The email of the user.
  const displayName = user.displayName; // The display name of the user.
  return sendWelcomeEmail(email, displayName);
});

// Sends a welcome email to the given user.
function sendWelcomeEmail(email, displayName) {
  const mailOptions = {
    from: `${APP_NAME} <noreply@firebase.com>`,
    to: email
  };

  // The user subscribed to the newsletter.
  mailOptions.subject = `Welcome to ${APP_NAME}!`;
  mailOptions.text = `Hey ${displayName || ''}! Welcome to ${APP_NAME}.`;
  return mailTransport.sendMail(mailOptions).then(() => {
    console.log('New welcome email sent to:', email);
  });
}
```

DATABASE TRIGGERS

- Regularly schedule updates
- App Engine Cron pushes to HTTP trigger on Firebase



HTTP TRIGGERS

- Count the number of messages and update the database

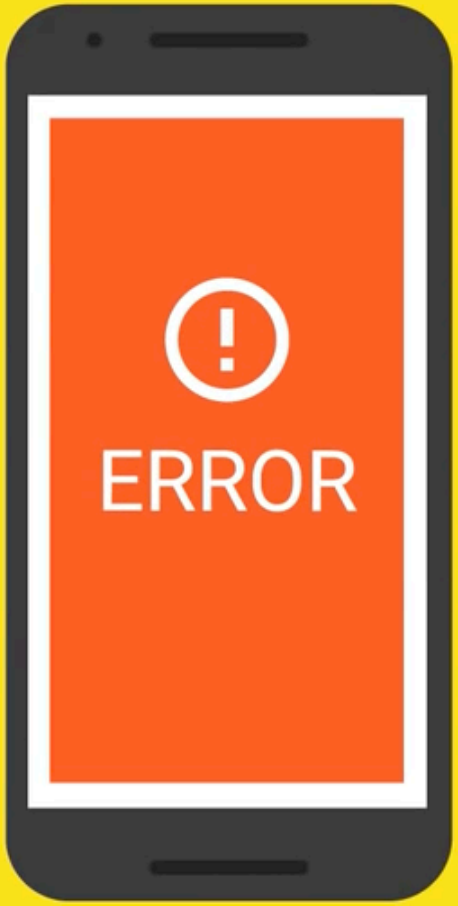
```
exports.cron = functions.https.onRequest((req
  // Forbidding PUT requests.
  if (req.method === 'PUT') {
    return res.status(403).send('Forbidden!')
  }

  var ref = admin.database().ref('messages')
  admin.database().ref().once('value', (data)
    console.log('Children', data.numChildren)
    admin.database().ref('messages').child('t
    res.status(200).send('updated from cron')
  });

});
```

BREAK TIME 

FIREBASE CRASHLYTICS
CRASH REPORTING

FIREBASE CRASH REPORTING

- https://www.youtube.com/watch?list=PLI-K7zZEsYLmOF_07IayrTntevxtbUxDL&v=B7mILVAkcfU#

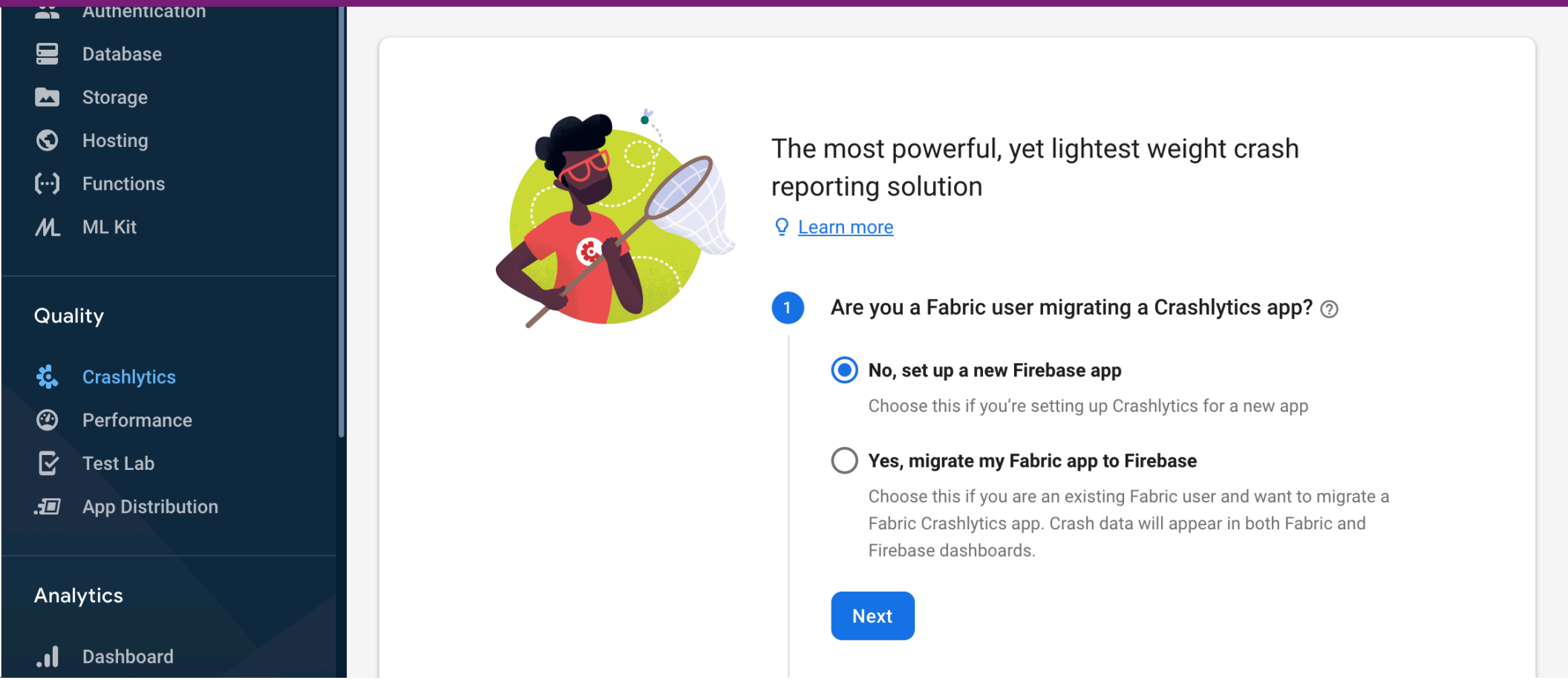


FIREBASE CRASH REPORTING

- Crashlytics acquisition
- Part of Firebase Core
- Free

Monitor fatal and non-fatal errors	Monitor fatal errors in iOS and fatal and non-fatal errors in Android. Reports are triaged by the severity of impact on users.
Collect the data you need to diagnose problems	Each report contains a full stack trace as well as device characteristics, performance data, and user circumstances when the error took place. Similar reports are automatically grouped into issues to make it easier to identify related bugs.
Email alerts	Enable email alerts to receive frequent updates when new crashes are uncovered or regressions are detected.
Integrate with Analytics	Errors captured are set as **`app_exception`** events in Analytics, allowing you to filter audiences based on who sees errors. In addition to grouping errors into similar stack traces, Crash Reporting also integrates with Analytics to provide you with the list of events that preceded a crash. This information helps to simplify your debugging process.

FIREBASE CRASH REPORTING



FIREBASE CRASH REPORTING

```
# platform :ios, '9.0'

target 'MyProject' do
# Comment the next line if you're not using Swift and don't want to use
use_frameworks!

# Pods for PodTest
pod 'Fabric', '~> 1.10.2'
pod 'Crashlytics', '~> 3.14.0'

# (Recommended) Pod for Google Analytics
pod 'Firebase/Analytics'
end
```

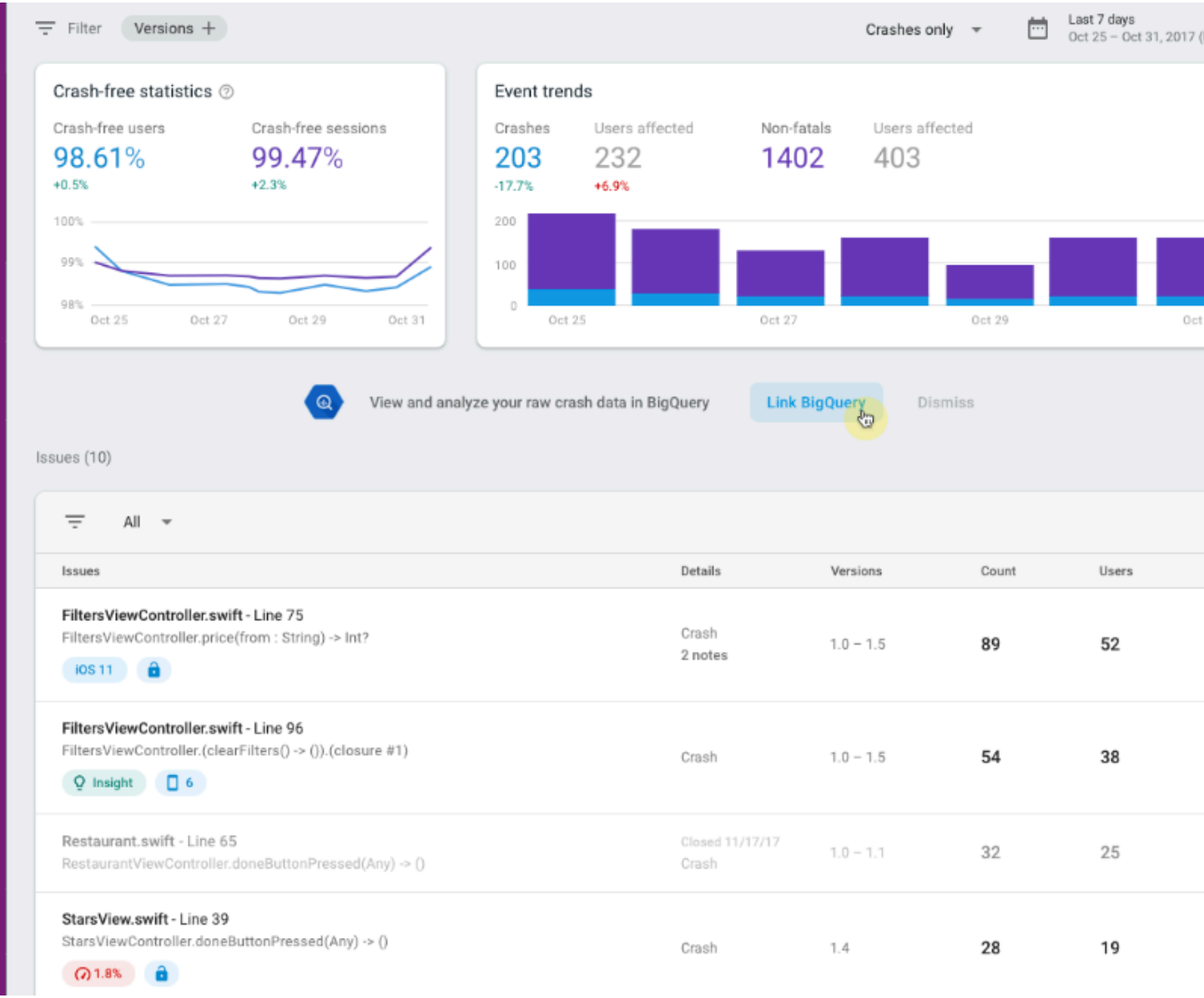
FIREBASE CRASH REPORTING

```
@IBAction func didPressCrash(_ sender: AnyObject) {
  FIRCrashMessage("Forced Crash")
  fatalError()
}
```

- Add additional information to crash messages
- Should be anonymous

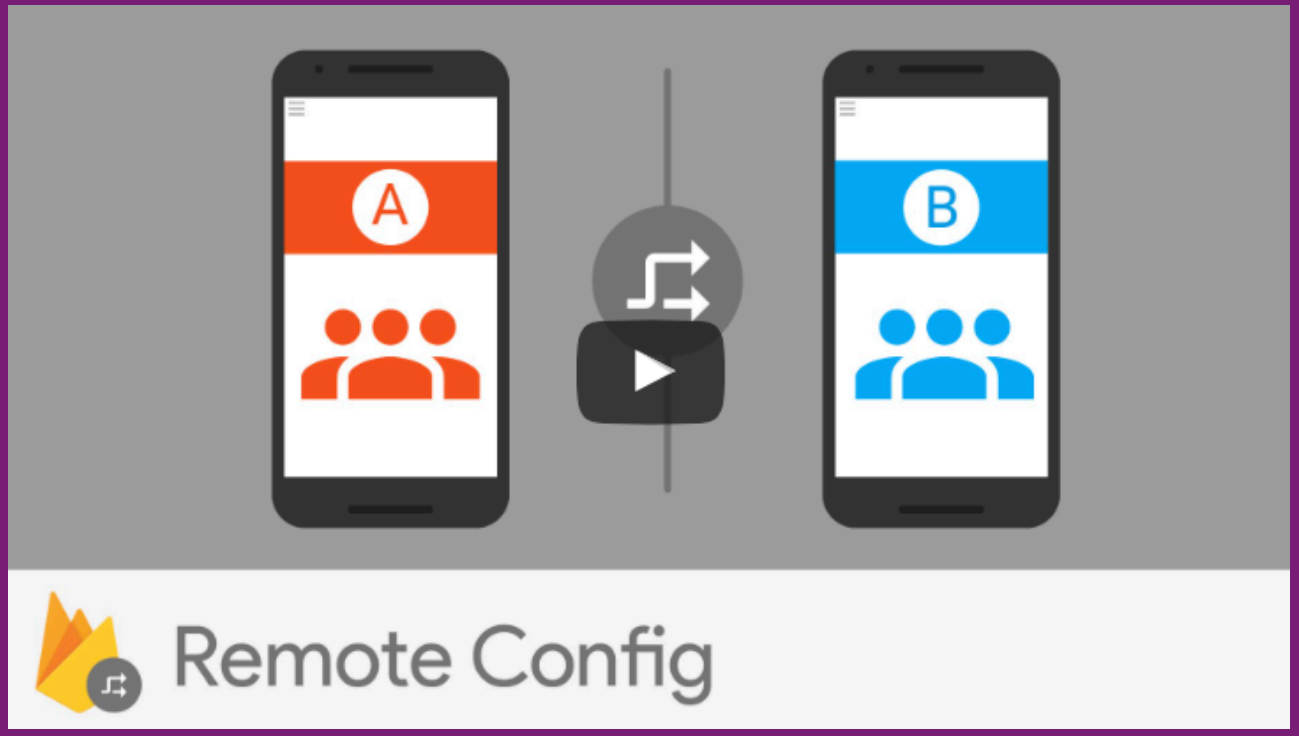
FIREBASE CRASH REPORTING

- Logs crash data
- Deobsfucation (human readable)



FIREBASE REMOTE CONFIG

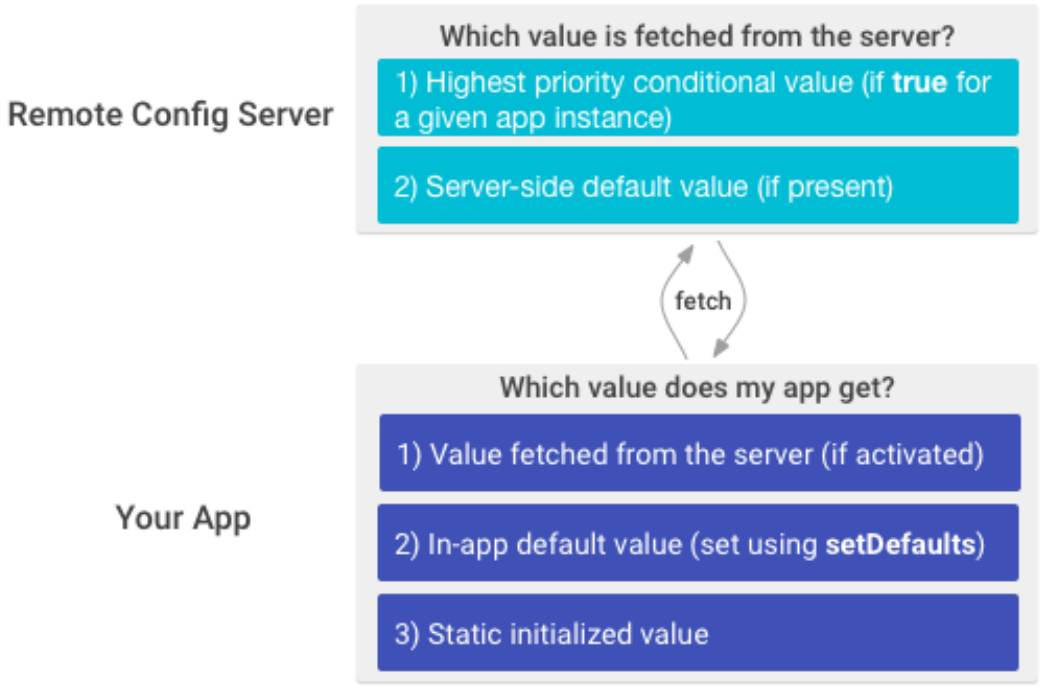
REMOTE CONFIG



- https://youtu.be/_CXXVFPO6f0?list=PLI-K7zZEsYLmOF_07IayrTntevxtbUxDL

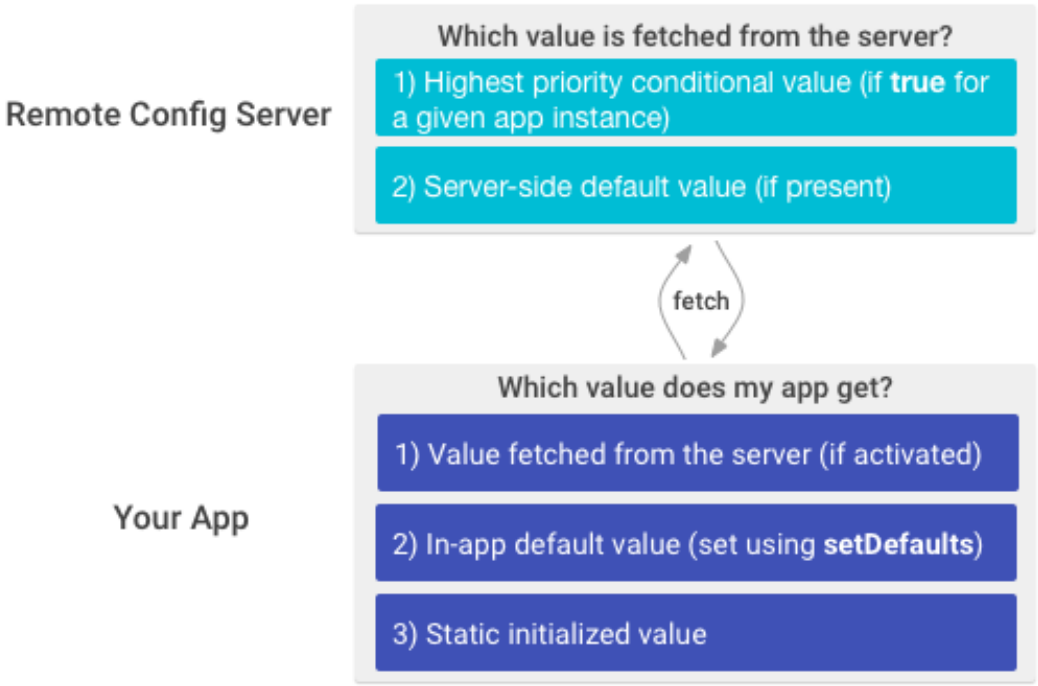
REMOTE CONFIG

- Firebase Remote Config is a cloud service that lets you change the behavior and appearance of your app without requiring users to download an app update



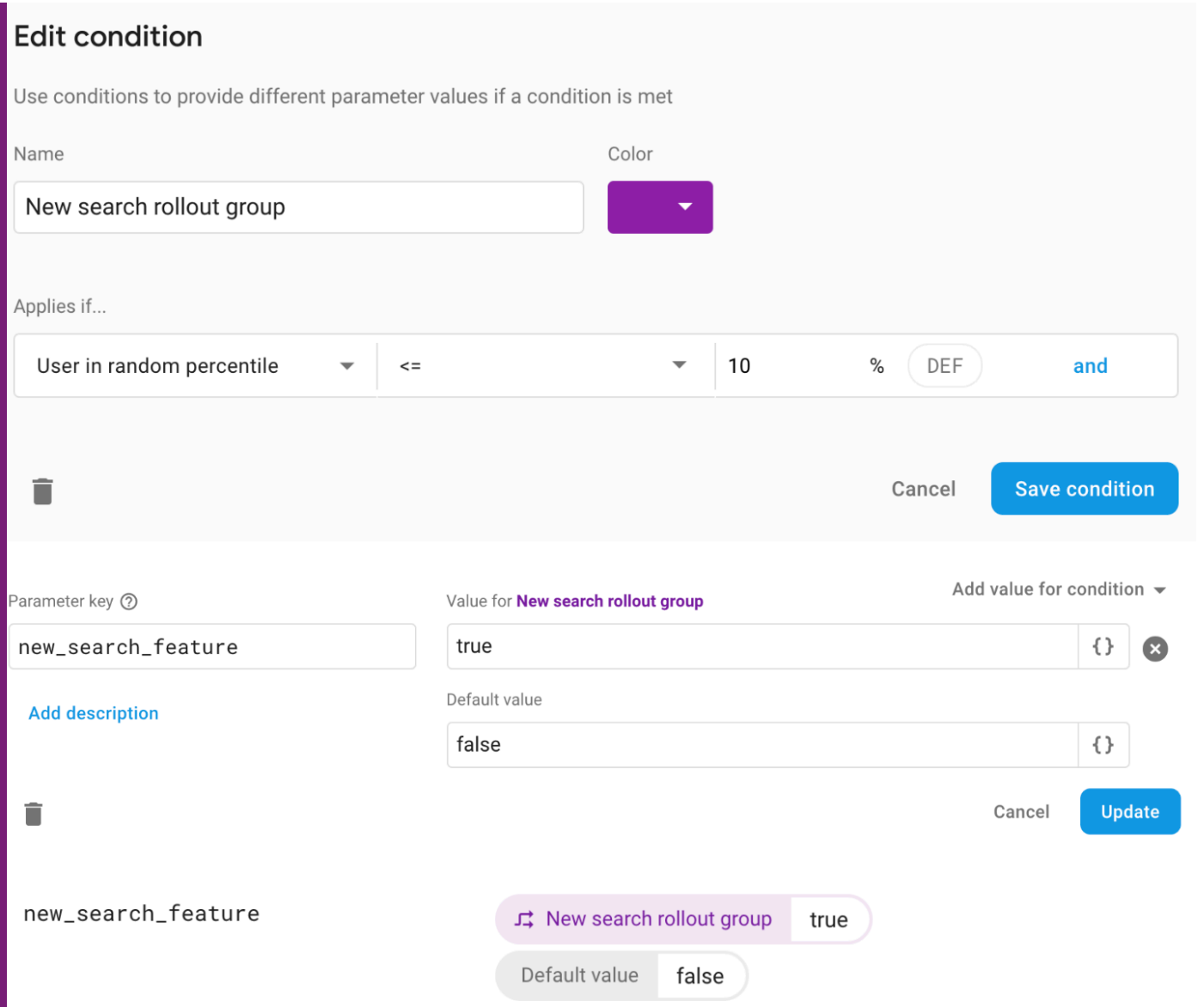
REMOTE CONFIG

- Create values that can be changed in the console to make them appear in your app
- There are plenty of ways to do this...this is just less work

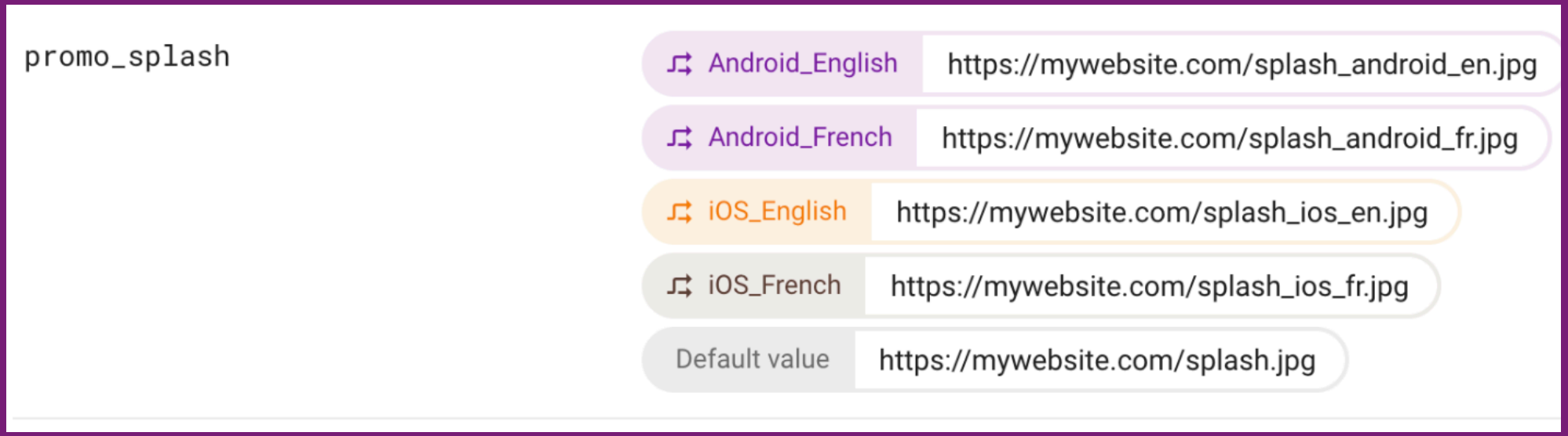


REMOTE CONFIG

- Use cases
  - Staged rollout



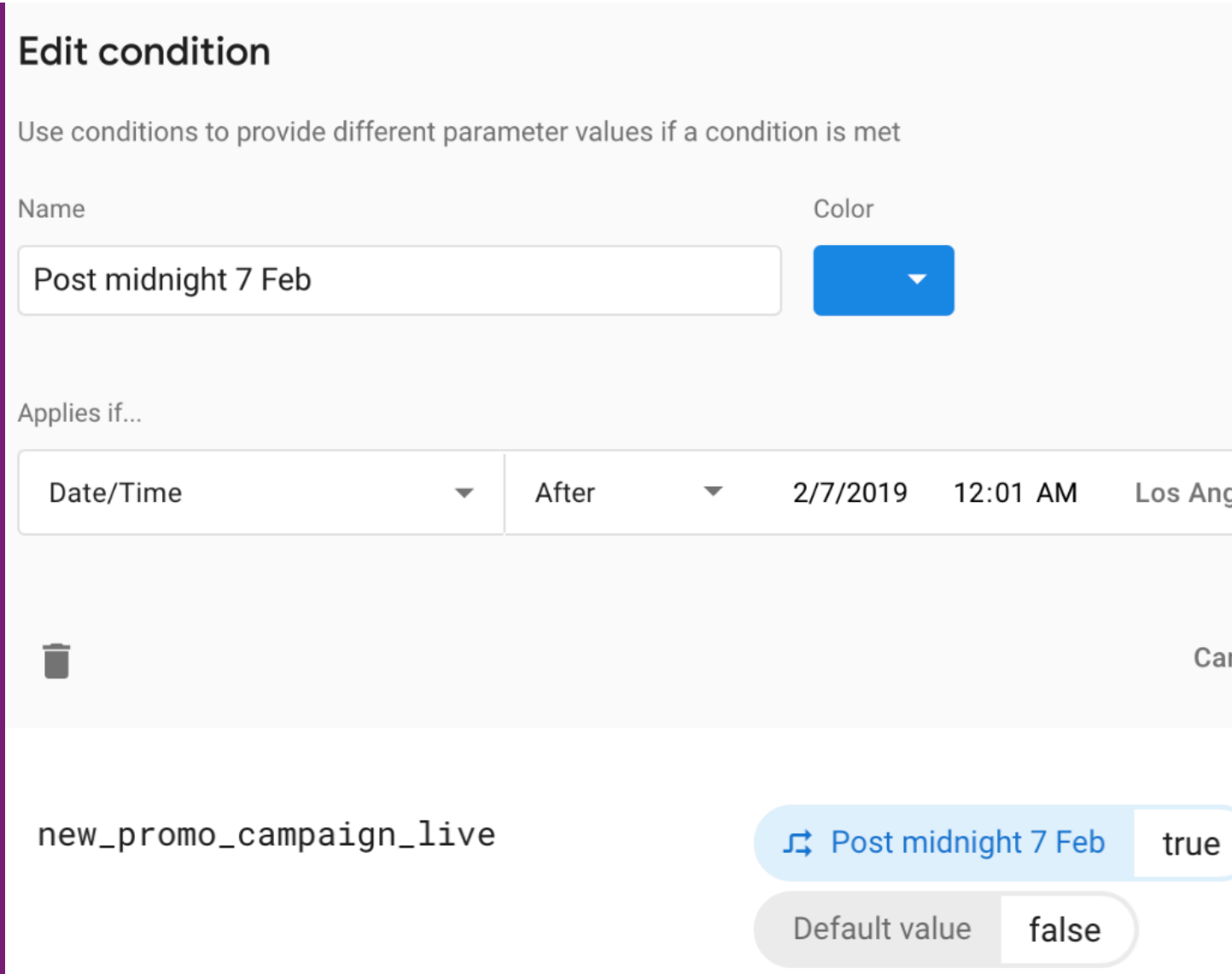
REMOTE CONFIG



- Use cases
  - Platform/language specific assets

REMOTE CONFIG

- Use cases
  - Start and stop promotions, ads, etc.



REMOTE CONFIG

- Note the following policies
  - Don't use Remote Config to make app updates that should require a user's authorization. This could cause your app to be perceived as untrustworthy.
  - Don't store confidential data in R[redacted]ys or parameter values. It is possible to [redacted]s or values stored in the Remote Config settings for your project
  - Don't attempt to circumvent the requirements of your app's target platform using Remote Config.

REMOTE CONFIG

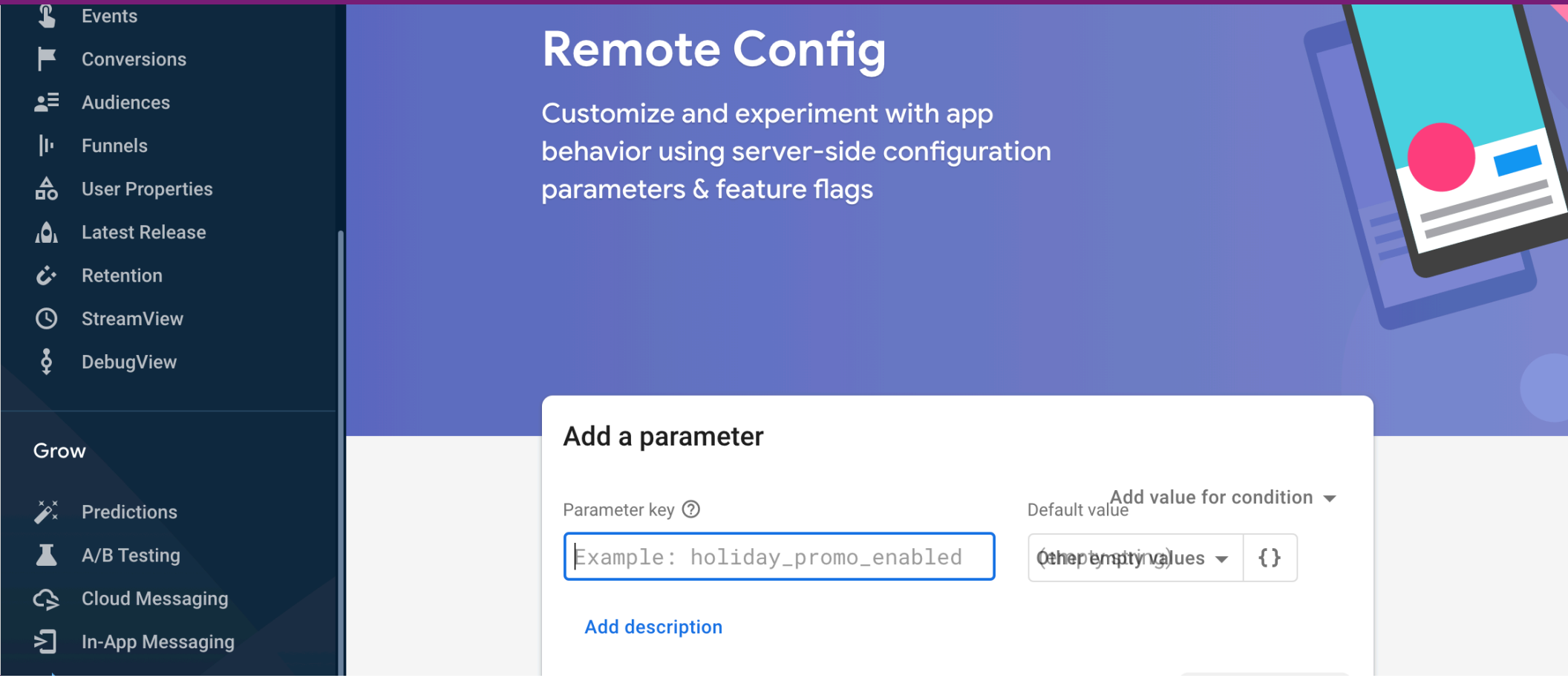
- Note the following policies
  - Don't use Remo[illegible]uire a user's authorization. T[illegible]rustworthy.
  - Don't store con[illegible]or parameter value[illegible]r values stored in the Re[illegible]
  - Don't attempt t[illegible]get platform using Remote Config.

IF FIRST RUN
AND IN CUPERTINO,
DON'T RUN MY SECRET ANALYTICS CODE :)

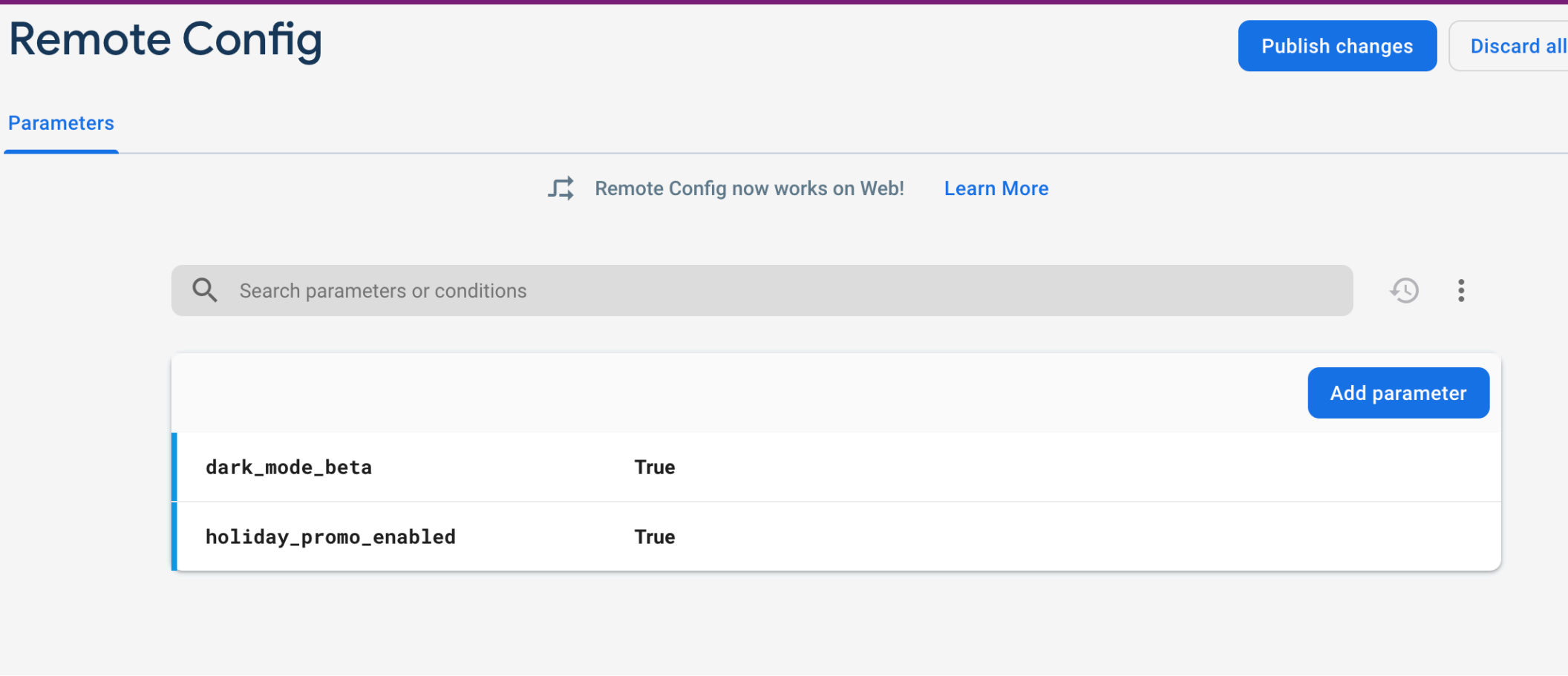
REMOTE CONFIG

```
target 'FireChat' do
  # Comment the next line if you're not using Swift and don't want to use dynamic frameworks
  use_frameworks!

  # Pods for FireChat
  pod 'Firebase/Core'
  pod 'Firebase/Database'

  pod 'Firebase/Auth'
  pod 'GoogleSignIn'

  pod 'Firebase/Crash'

  pod 'Firebase/Storage'
  pod 'Firebase/Invites'

  pod 'Firebase/DynamicLinks'

  pod 'Firebase/RemoteConfig'

  pod 'SDWebImage'

end
```

REMOTE CONFIG



REMOTE CONFIG



REMOTE CONFIG

- Set default values
  - Programmatically
  - Include a plist
- Fetch latest content
- Activate Fetched values

```swift
import Foundation
import Firebase

class RemoteConfigManager {

  static let sharedInstance = RemoteConfigManager()
  let remoteConfig = FIRRemoteConfig.remoteConfig()

  private init() {
    activateDebugMode()
    loadDefaultValues()
  }

  func loadDefaultValues() {
    let appDefaults: [String: NSObject] = ["joke_of_the_day": "Default Joke" as NSObject]
    remoteConfig.setDefaults(appDefaults)
  }

  ///
  func fetchCloudValues() {


    let fetchDuration: TimeInterval = 0
    activateDebugMode()
    remoteConfig.fetch(withExpirationDuration: TimeInterval(fetchDuration)) { (status, error) -> Void in


      if status == .success {
        print("Config fetched!")
        self.remoteConfig.activateFetched()



      } else {
        print("Config not fetched")
        print("Error \(error!.localizedDescription)")
      }
      print("Joke of the day: \(self.remoteConfig["joke_of_the_day"].stringValue!)")
    }
  }


  func activateDebugMode() {
    let debugSettings = FIRRemoteConfigSettings(developerModeEnabled: true)
    remoteConfig.configSettings = debugSettings!
  }
```

SET DEFAULTS FOR FIRST TIME AND NO CONNECTION

REMOTE CONFIG

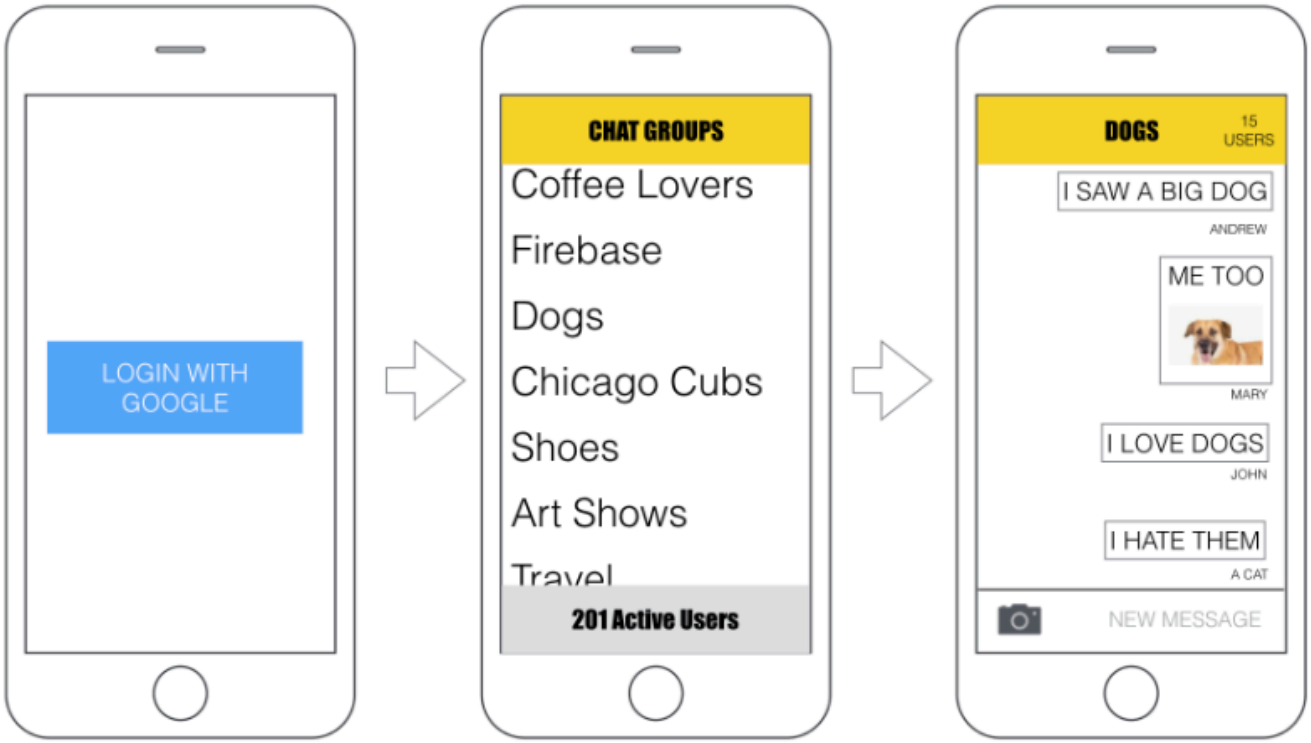
- Set default values
  - Programmatically
  - Include a plist
- Fetch latest content
- Activate Fetched values

```
import Foundation
import Firebase

class RemoteConfigManager {

  static let sharedInstance = RemoteConfigManager()
  let remoteConfig = FIRRemoteConfig.remoteConfig()

  private init() {
    activateDebugMode()
    loadDefaultValues()
  }

  func loadDefaultValues() {
    let appDefaults: [String: NSObject] = ["joke_of_the_day": "Default Jo as NSObject]
    remoteConfig.setDefaults(appDefaults)
  }

  ///
  func fetchCloudValues() {


    let fetchDuration: TimeInterval = 0
    activateDebugMode()
    remoteConfig.fetch(withExpirationDuration: TimeInterval(fetchDuration)) { (status, error) -> Void in


      if status == .success {
        print("Config fetched!")
        self.remoteConfig.activateFetched()



      } else {
        print("Config not fetched")
        print("Error \(error!.localizedDescription)")
      }
      print("Joke of the day: \(self.remoteConfig["joke_of_the_day"].stringValue!)")
    }
  }

  func activateDebugMode() {
    let debugSettings = FIRRemoteConfigSettings(developerModeEnabled: true)
    remoteConfig.configSettings = debugSettings!
  }
```

FETCH CLOUD VALUES

ASSIGNMENT 5

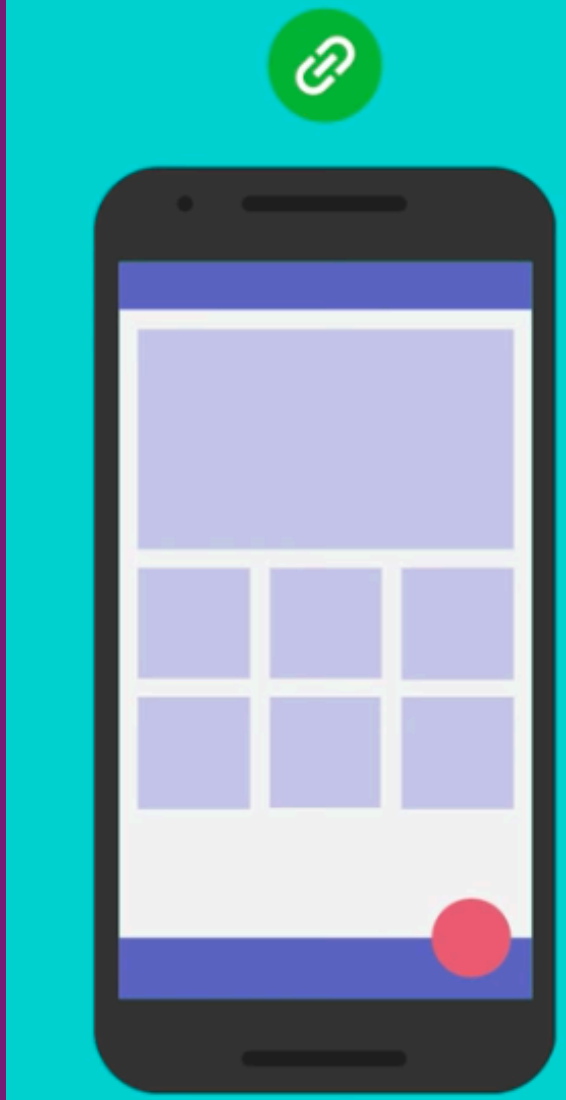
ASSIGNMENT 5



FIREBASE DYNAMIC LINKS

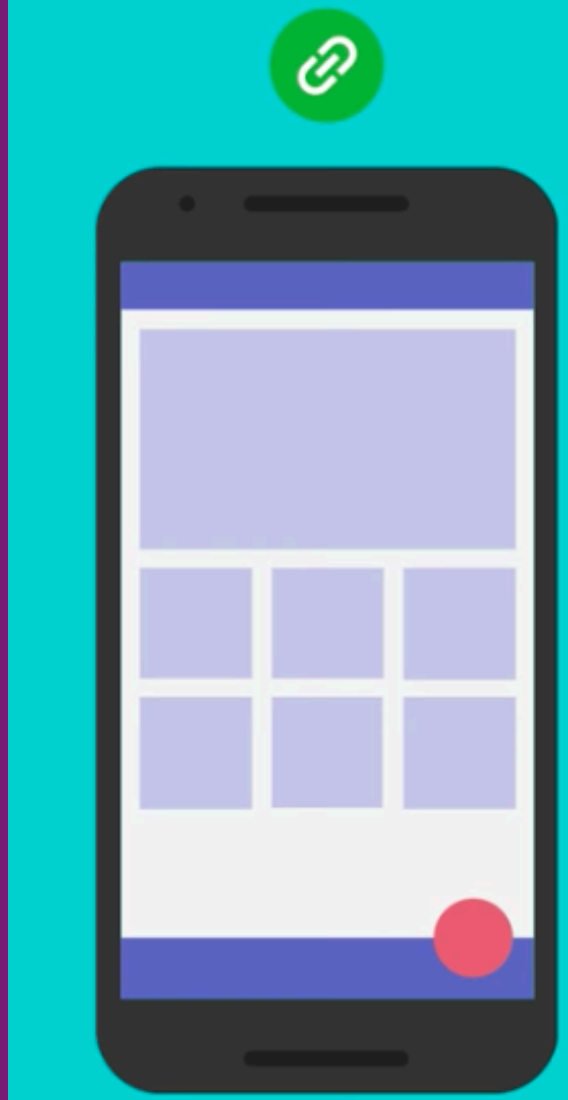
DYNAMIC LINKS

- Firebase dynamic links
  - https://youtu.be/LvY1JMcrPF8?list=PLl-K7zZEsYLmOF_07IayrTntevxtbUxDL



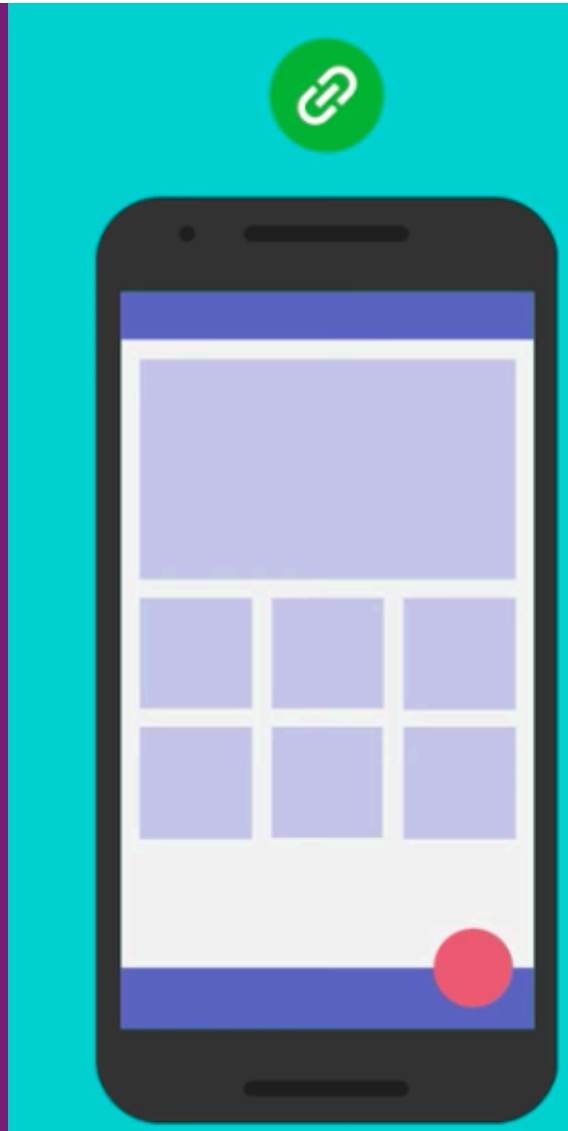
DYNAMIC LINKS

- Dynamic links can be taken directly to the linked content in your native app
- If a user opens the same Dynamic Link in a desktop browser, they can be taken to the equivalent content on your website



DYNAMIC LINKS

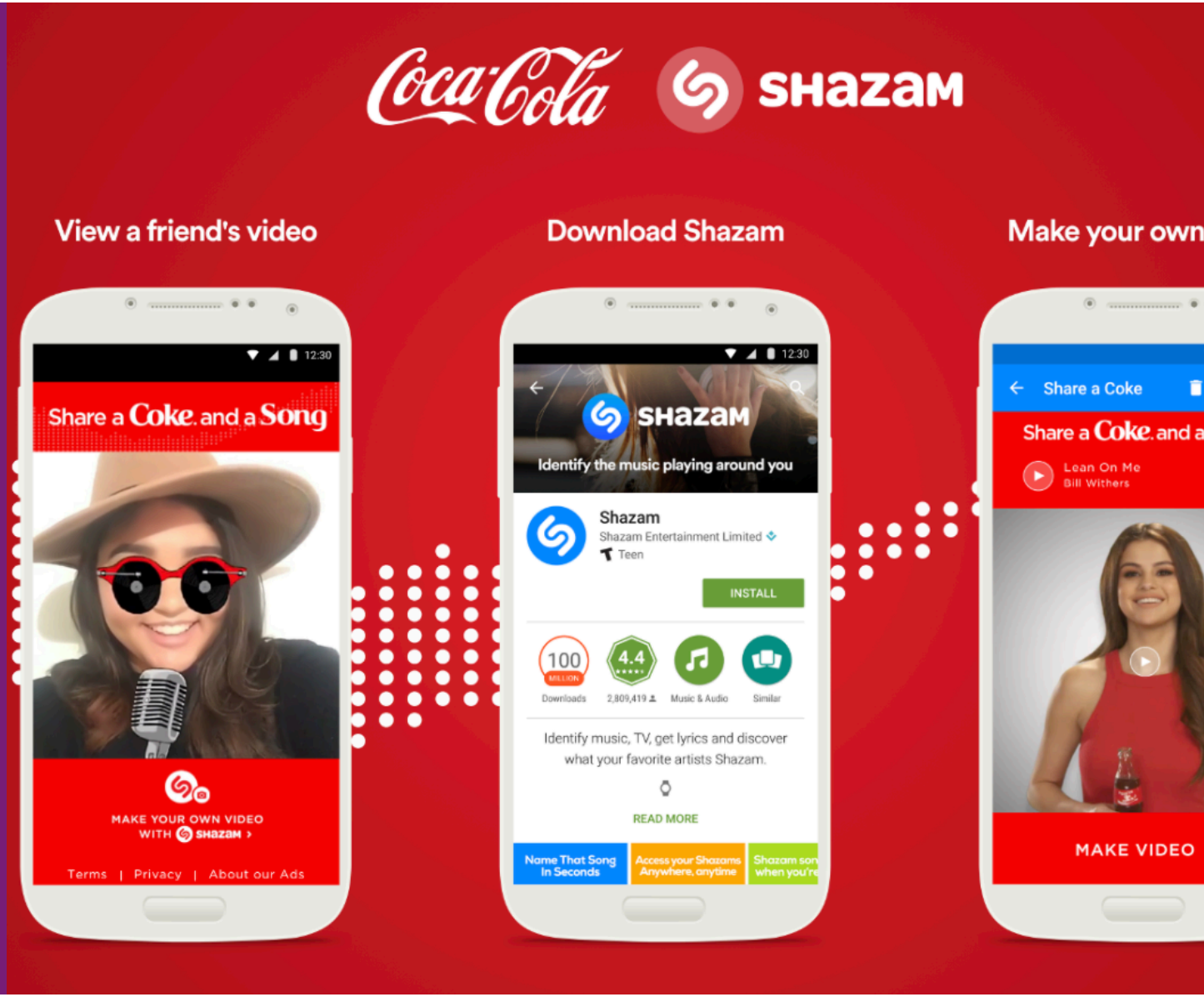
- Options for creating links
  - Firebase console
  - REST API
  - iOS Builder API
  - Programmatically



DYNAMIC LINKS

USE CASES

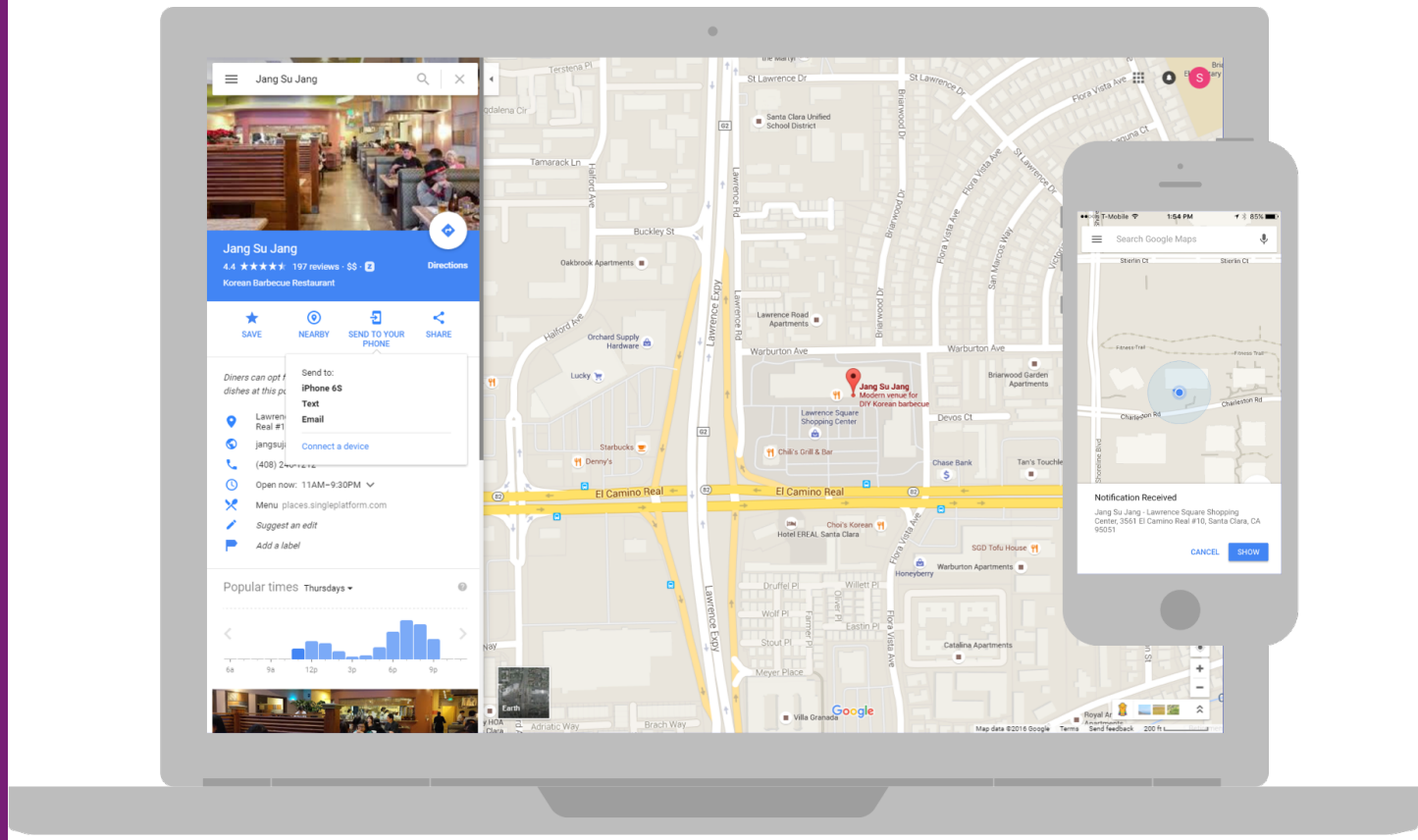
- Converting web (or desktop) users to app users



DYNAMIC LINKS

USE CASES

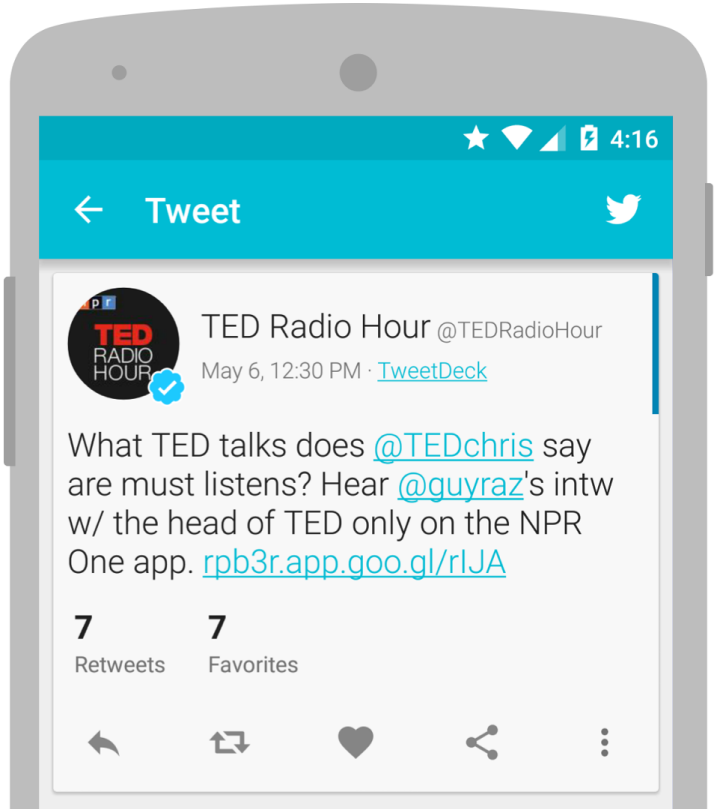
- Don't loose place

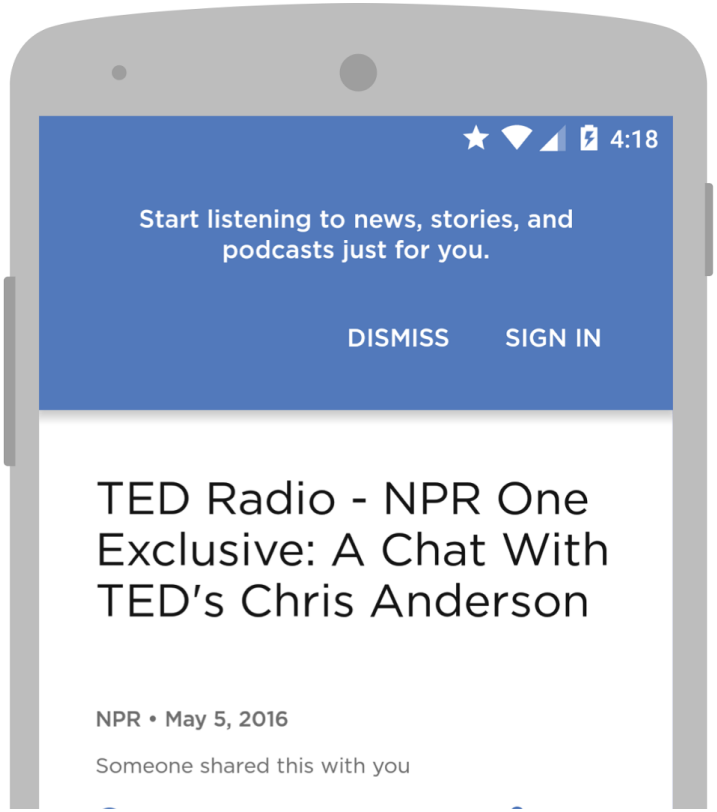


DYNAMIC LINKS

USE CASES

- Send links through social media
  - Direct content links
  - Track engagement

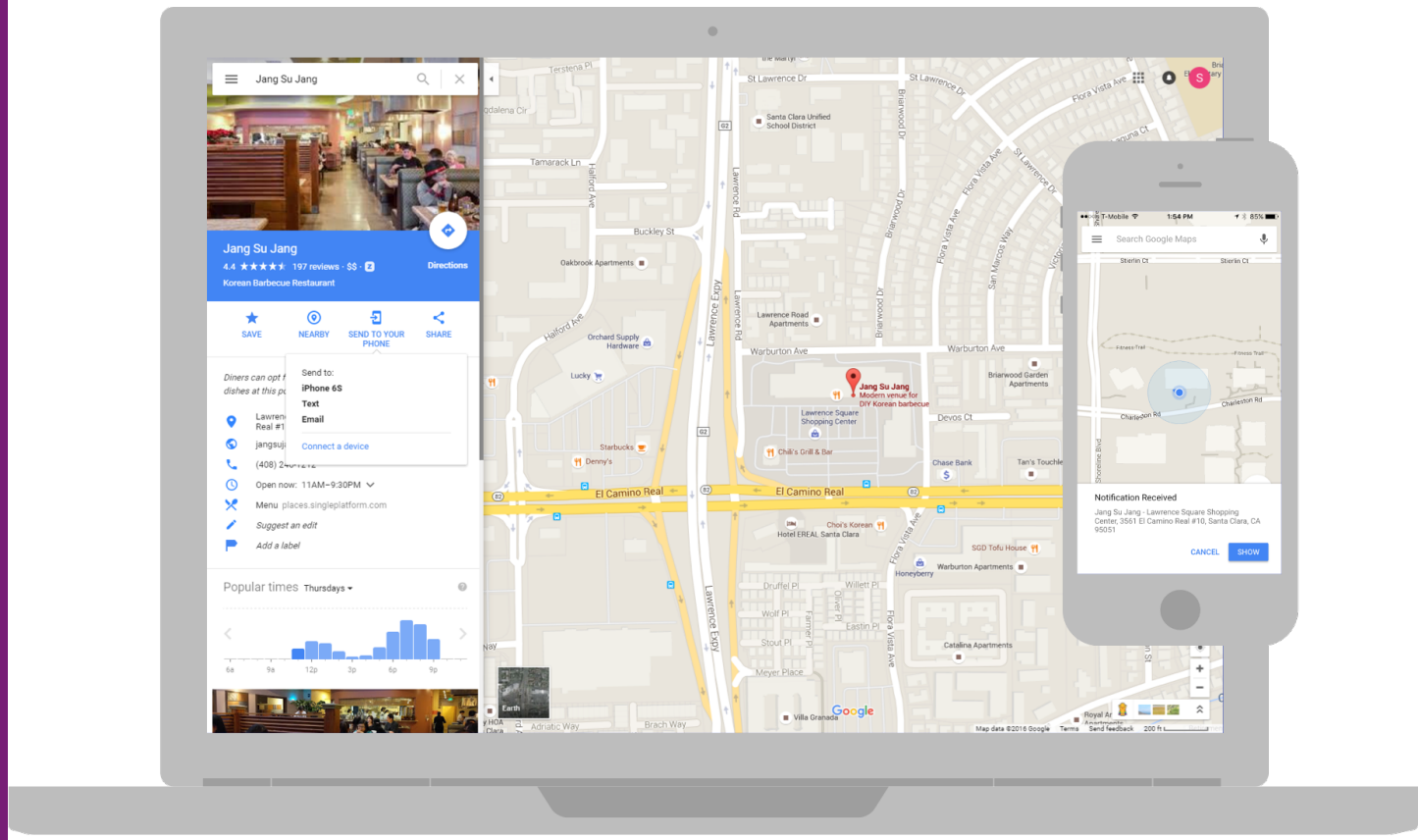




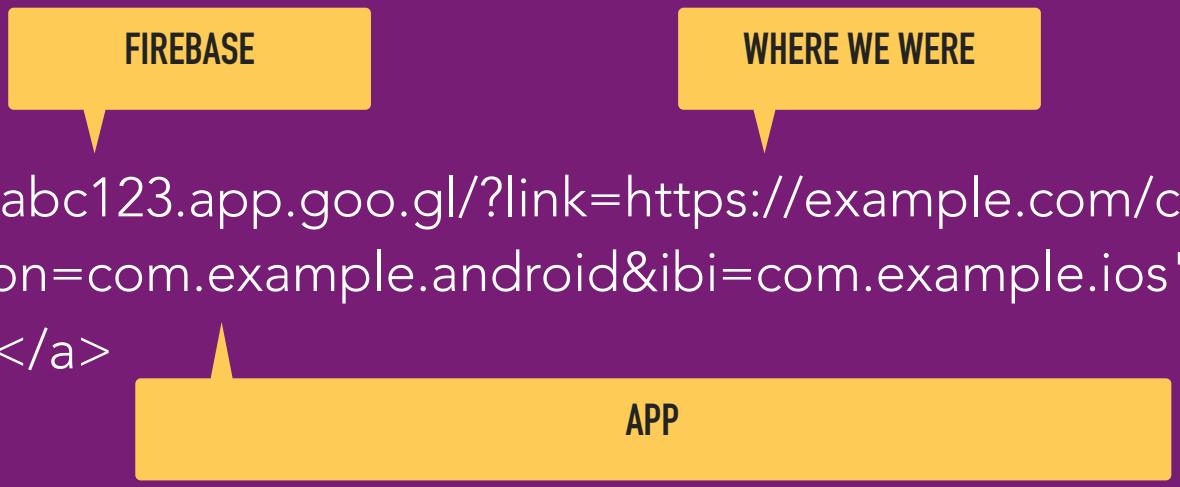
DYNAMIC LINKS

USE CASES

- Send links through social media
  - Direct content links
  - Track engagement



DYNAMIC LINKS



- Open this page in our app!

DYNAMIC LINKS

DYNAMIC LINKS

URL SHORTENER

- https://abc123.app.goo.gl/WXYZ

DYNAMIC LINKS

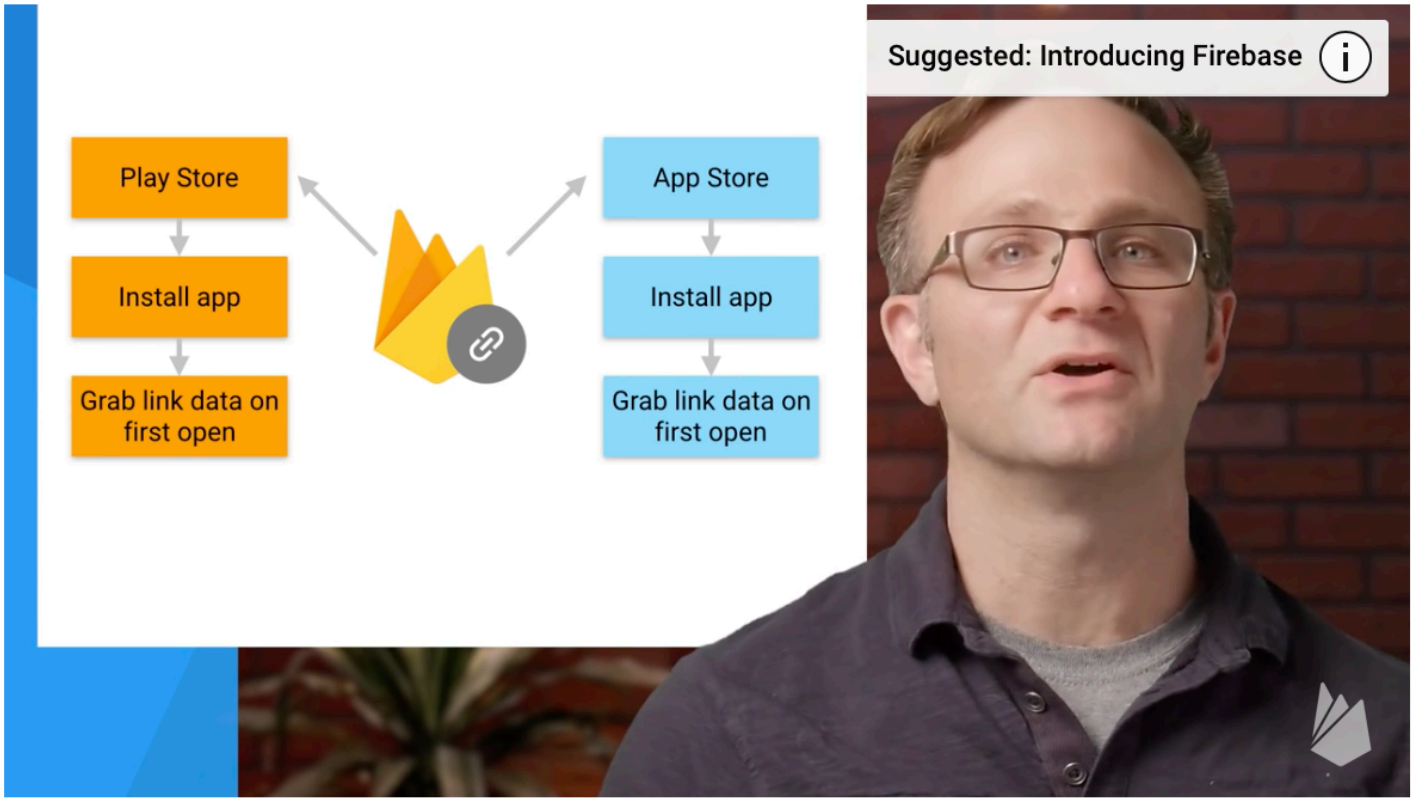
- Implementing deep links

1	Set up Firebase and the Dynamic Links SDK	Enable Firebase Dynamic Links for your Firebase project in the Firebase console. Then, include the Dynamic Links SDK in your app.
2	Create Dynamic Links	You can create Dynamic Links programmatically or by using the Firebase console.
3	Handle Dynamic Links in your app	When your app opens, use the Dynamic Links SDK to check if a Dynamic Link was passed to it. If so, get the link from the Dynamic Link data and handle the link as necessary.
4	View analytics data	Track the performance of your Dynamic Links in the Firebase console.

SETTING UP DYNAMICS LINKS

SETTING UP DYNAMICS LINKS

- https://youtu.be/sFPo296OQqk
- 15 minutes Firecast

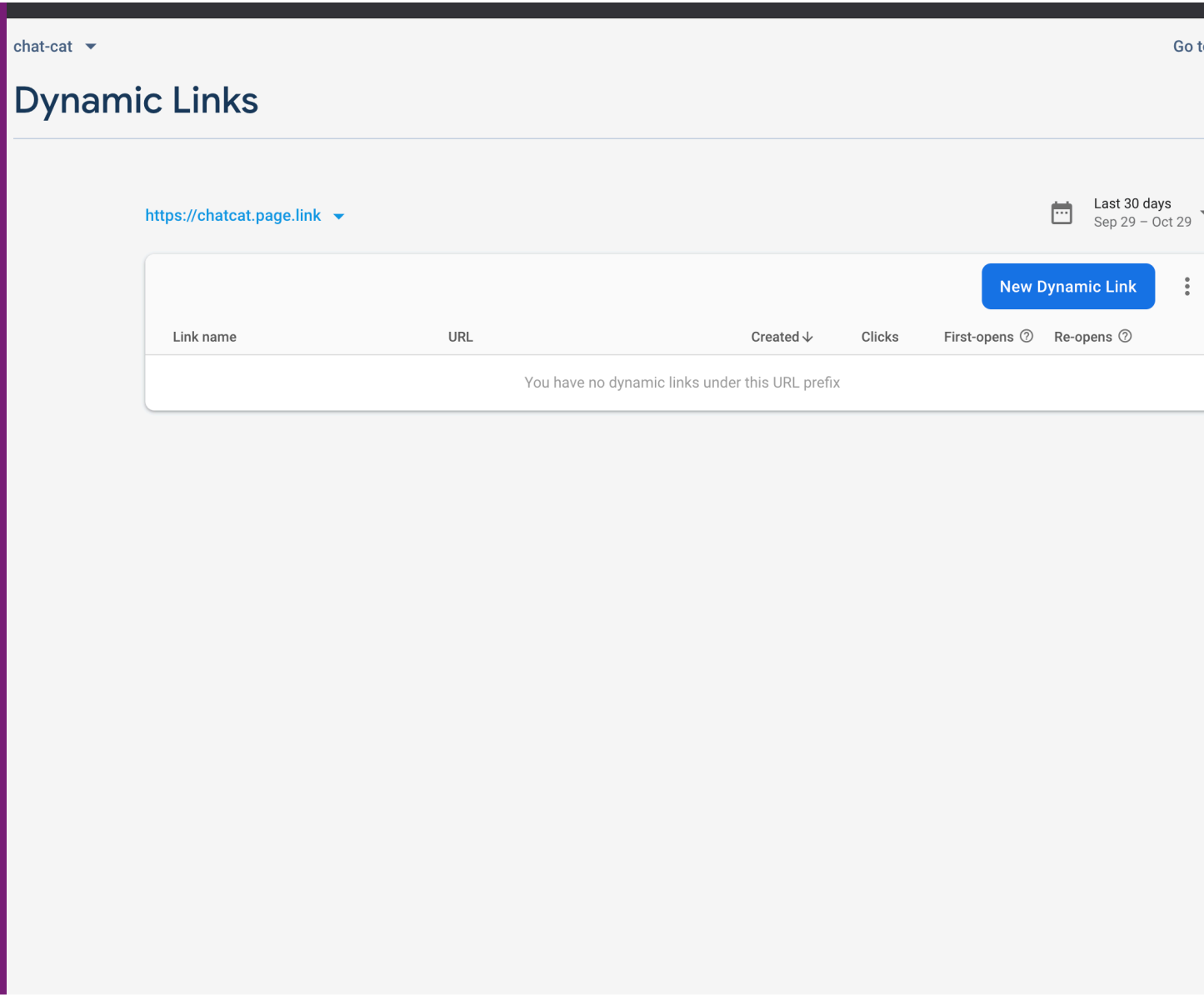


SETTING UP DYNAMICS LINKS

- Custom URL for your application

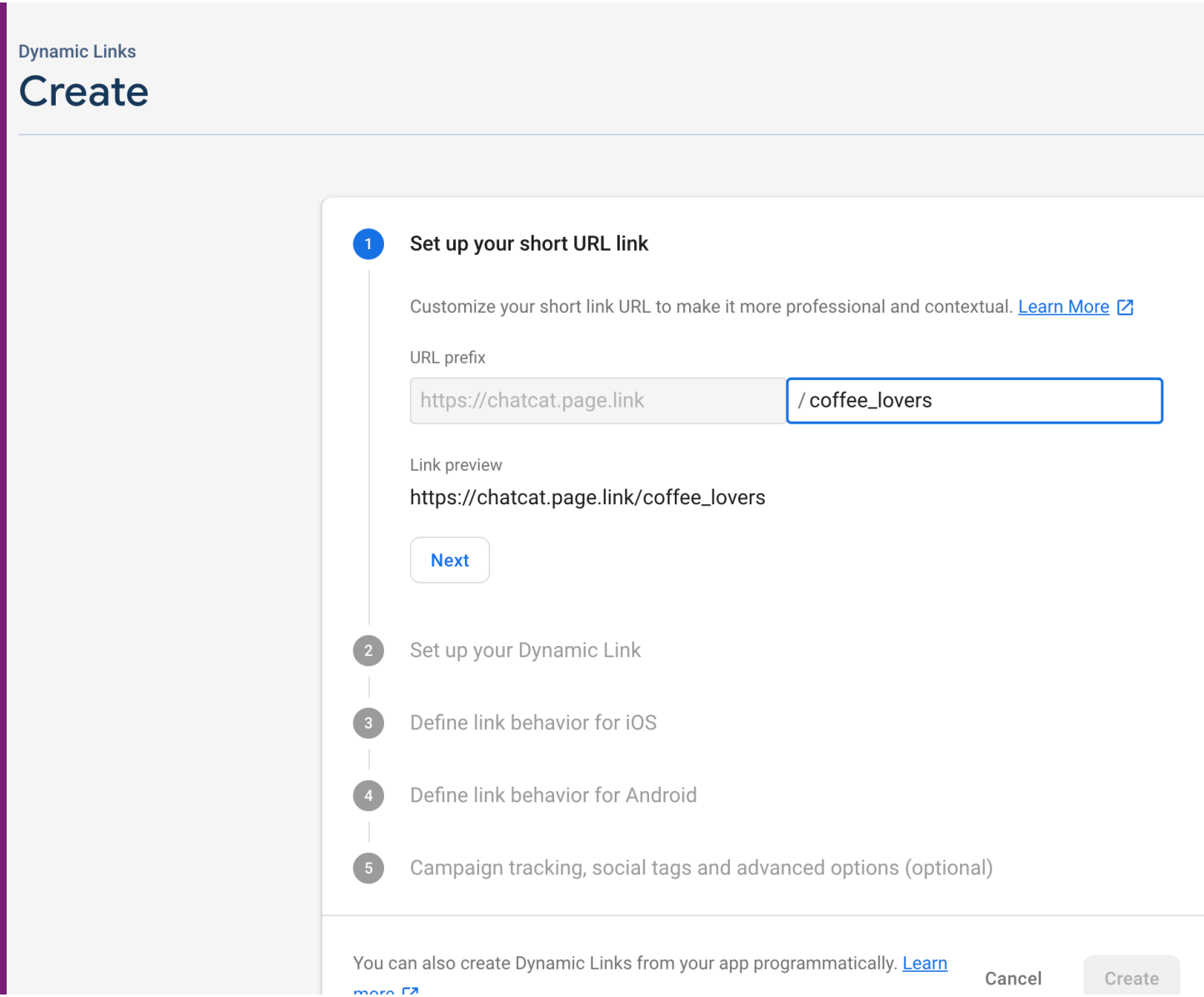
SETTING UP DYNAMICS LINKS

- Set up in the console
- youapp.page.link



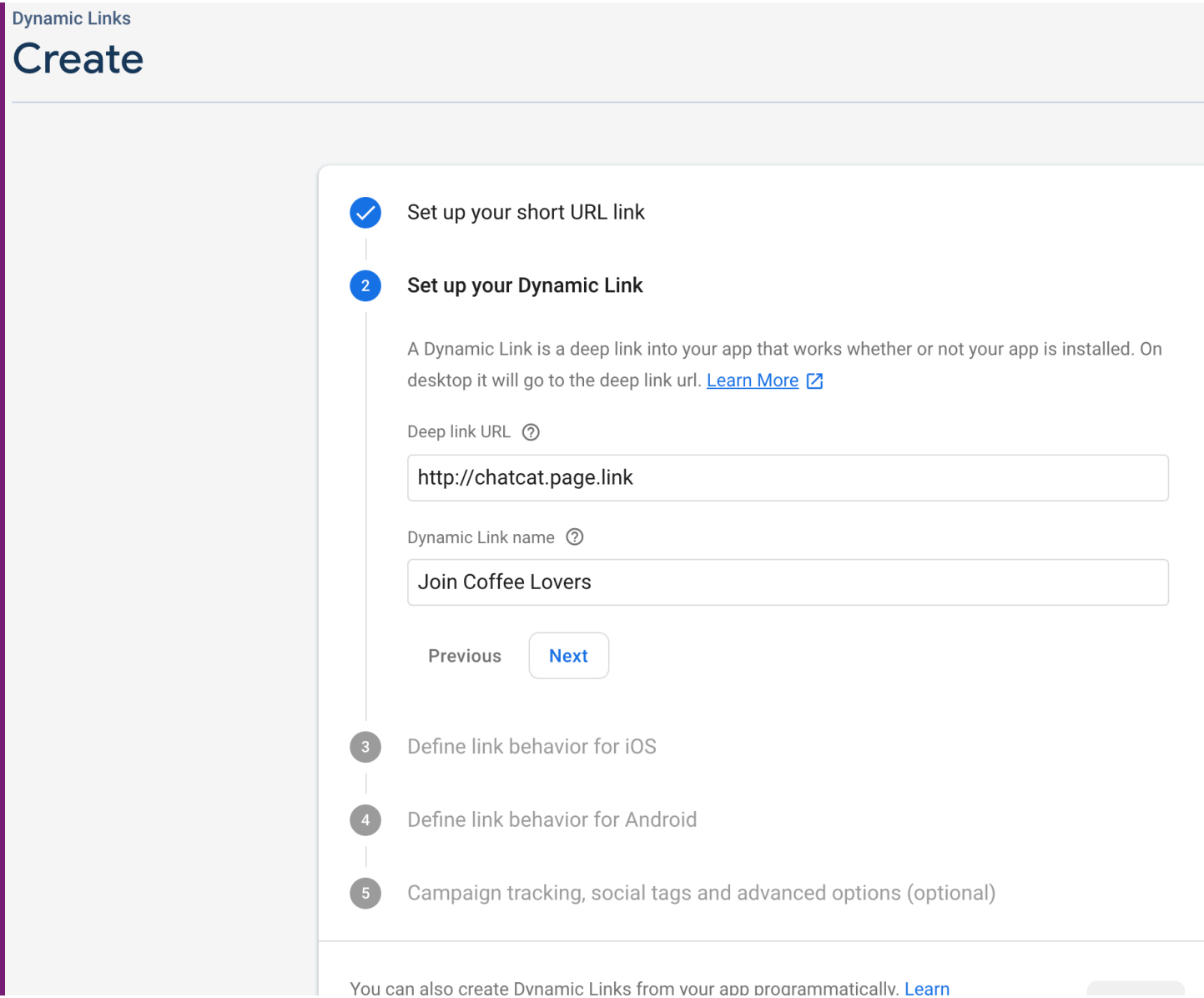
SETTING UP DYNAMICS LINKS

- Set up in the console
- youapp.page.link



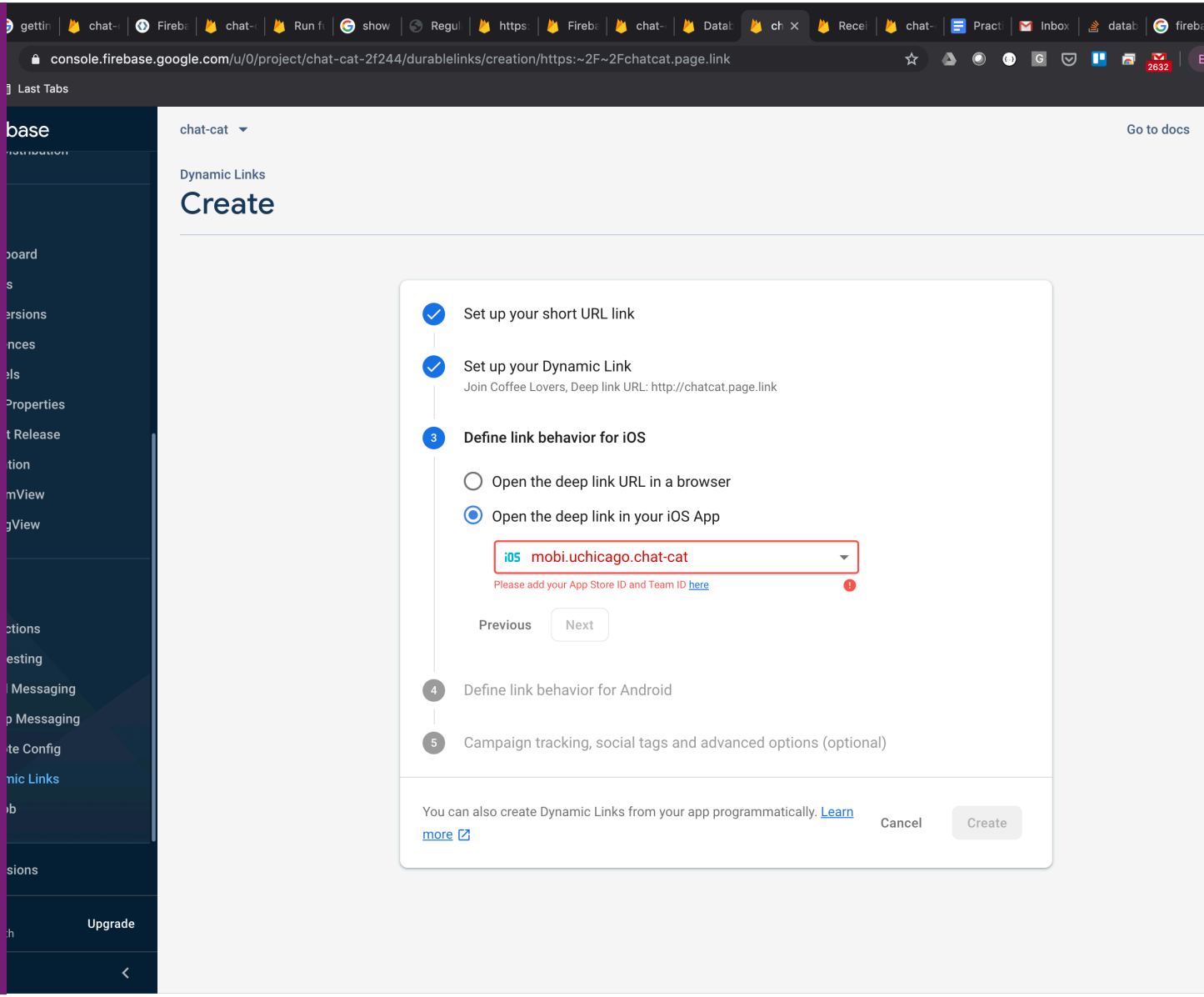
SETTING UP DYNAMICS LINKS

- Set up in the console
- youapp.page.link



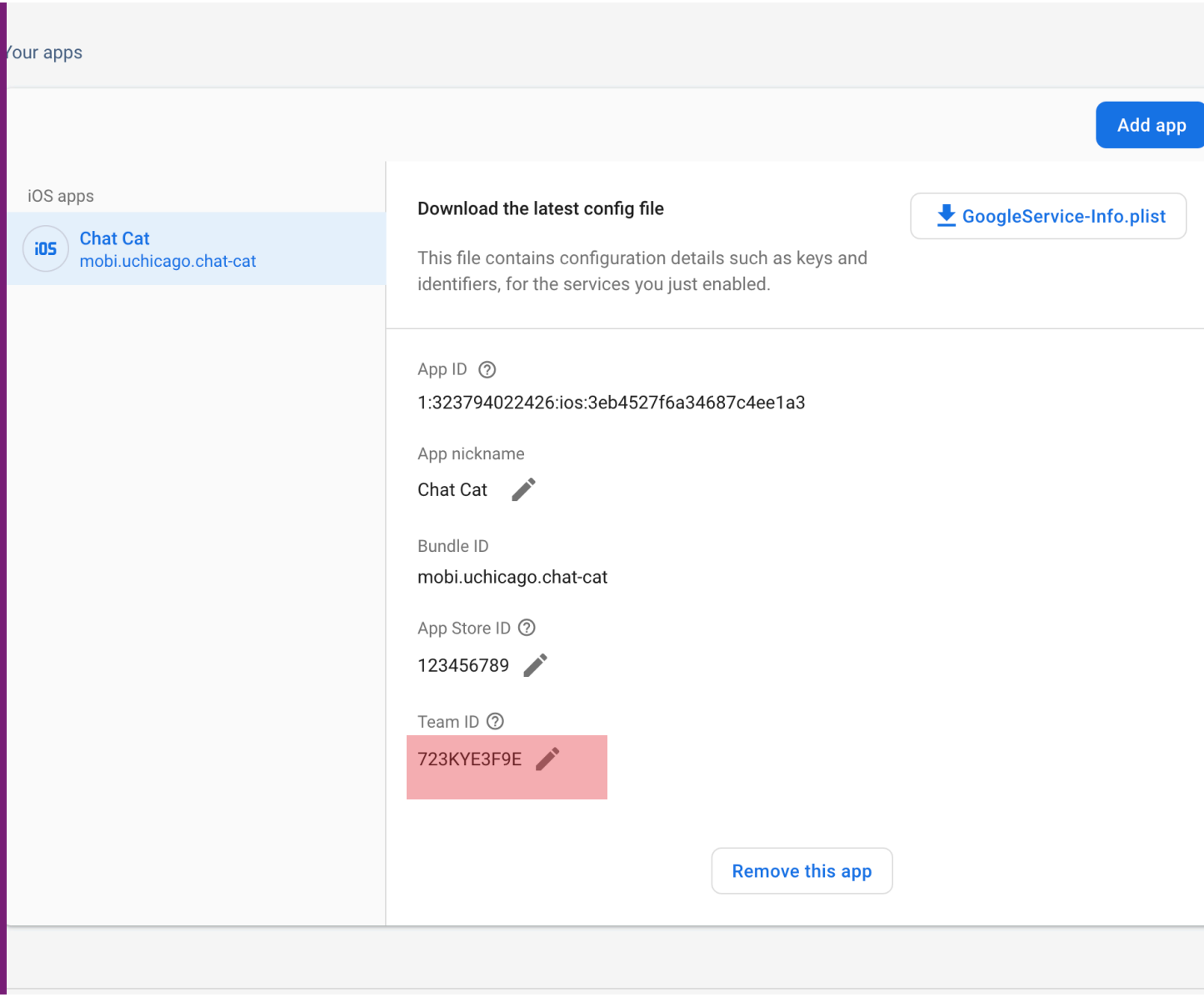
SETTING UP DYNAMICS LINKS

- Set up in the console
- youapp.page.link



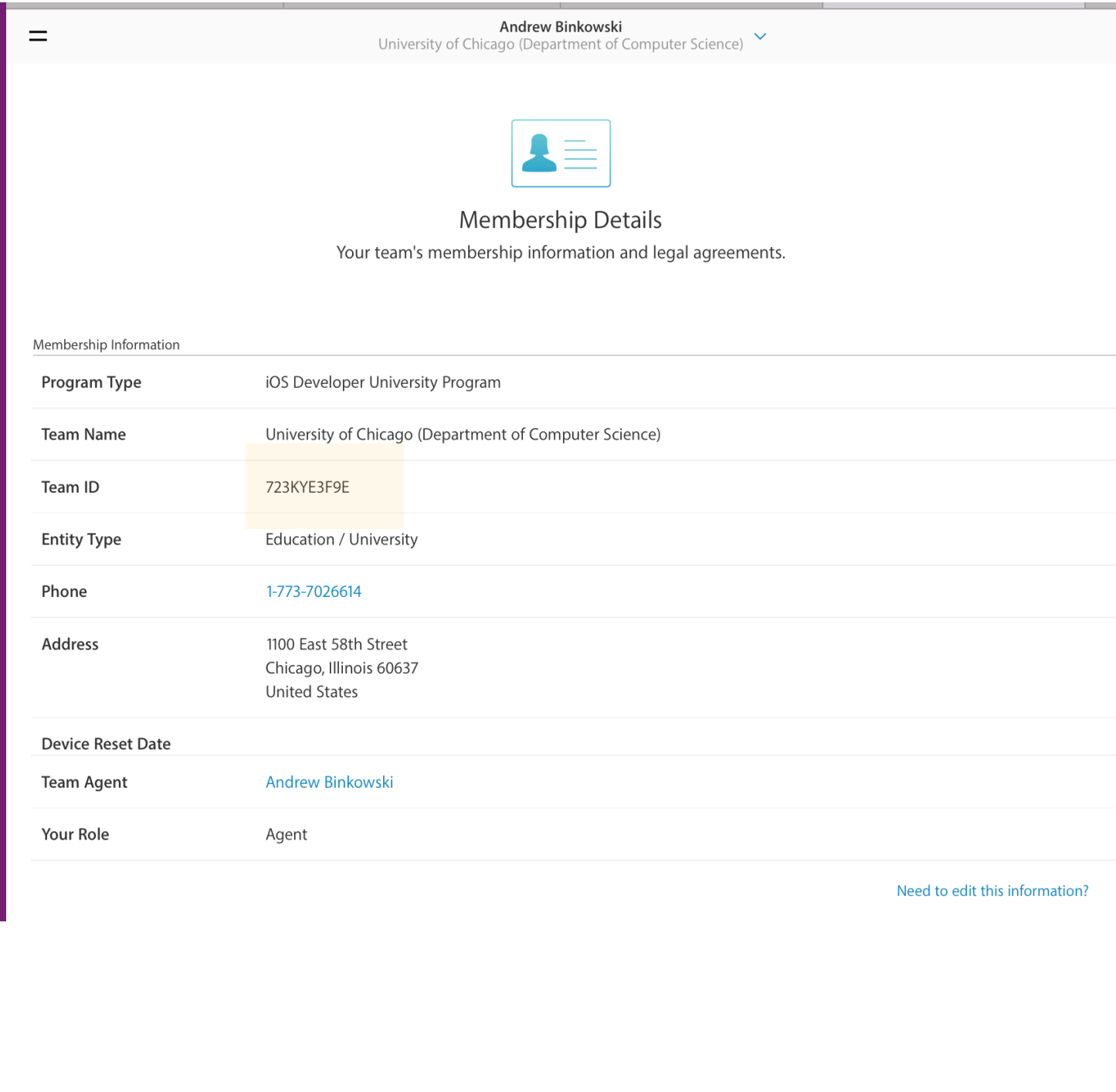
SETTING UP DYNAMICS LINKS

- Project settings
- Uniquely identifies the developer



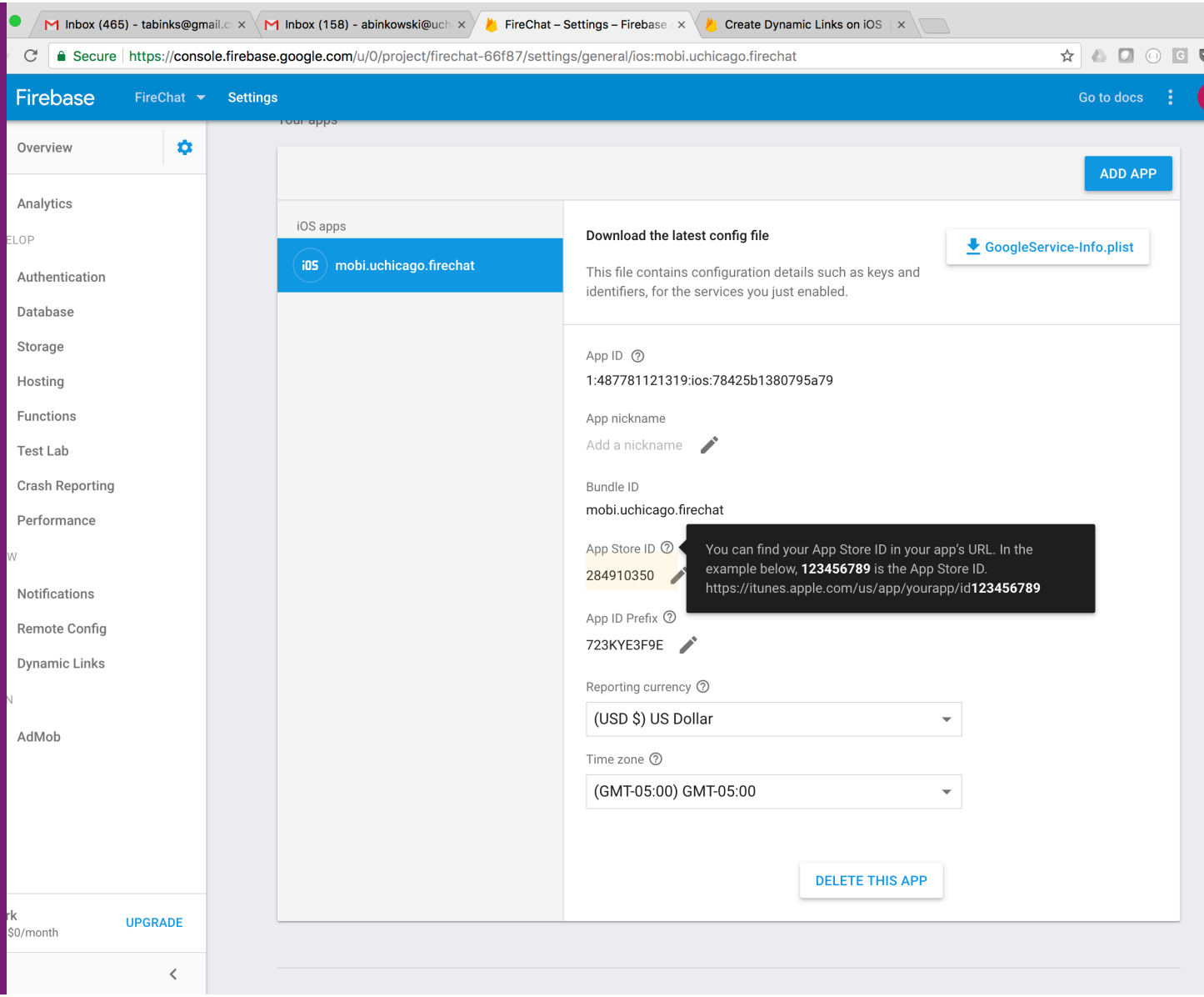
SETTING UP DYNAMICS LINKS

- Apple developer > Membership
- Team ID



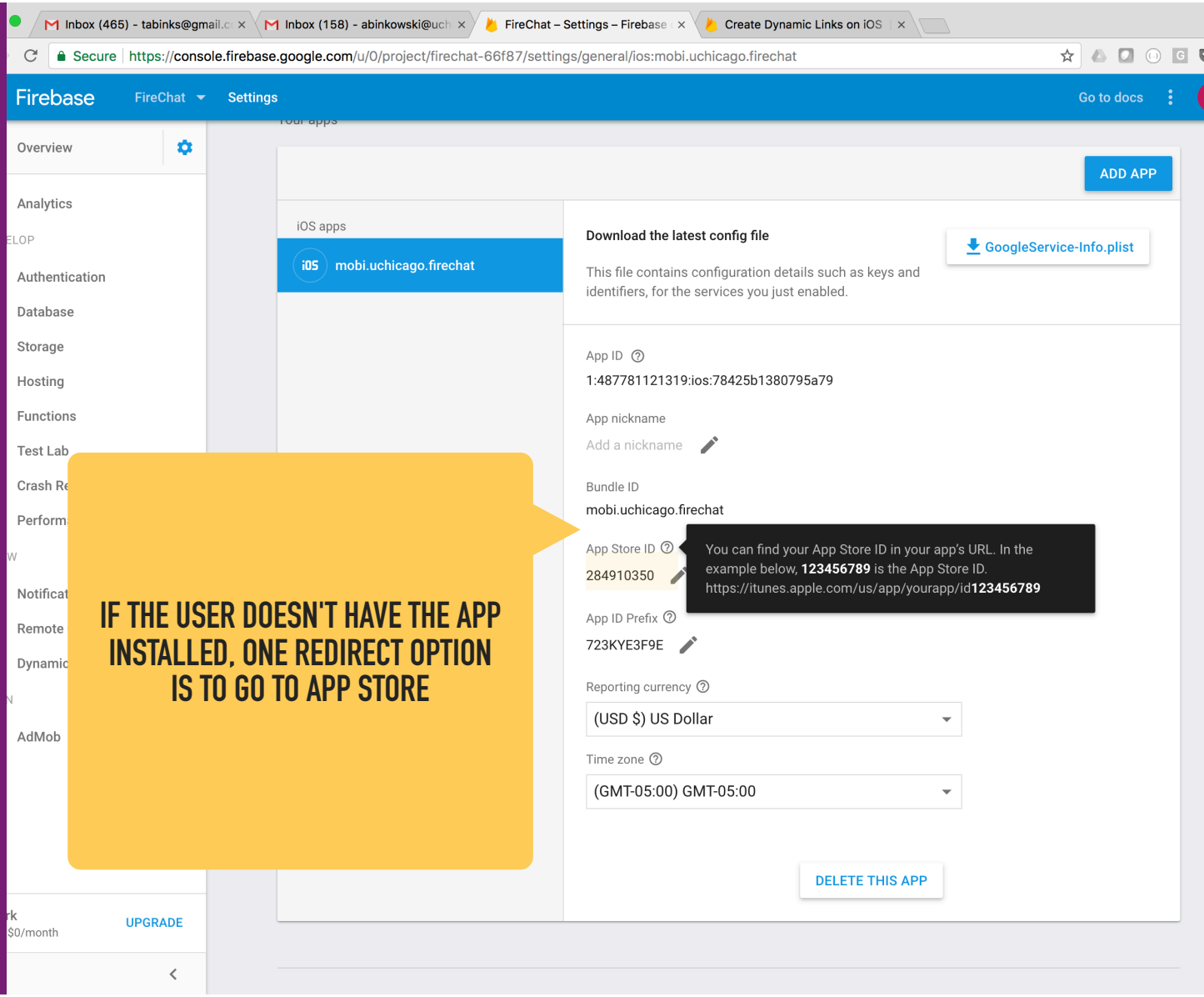
SETTING UP DYNAMICS LINKS

- Only published apps will have a App Store ID
  - Substitute your favorite app
  - during development



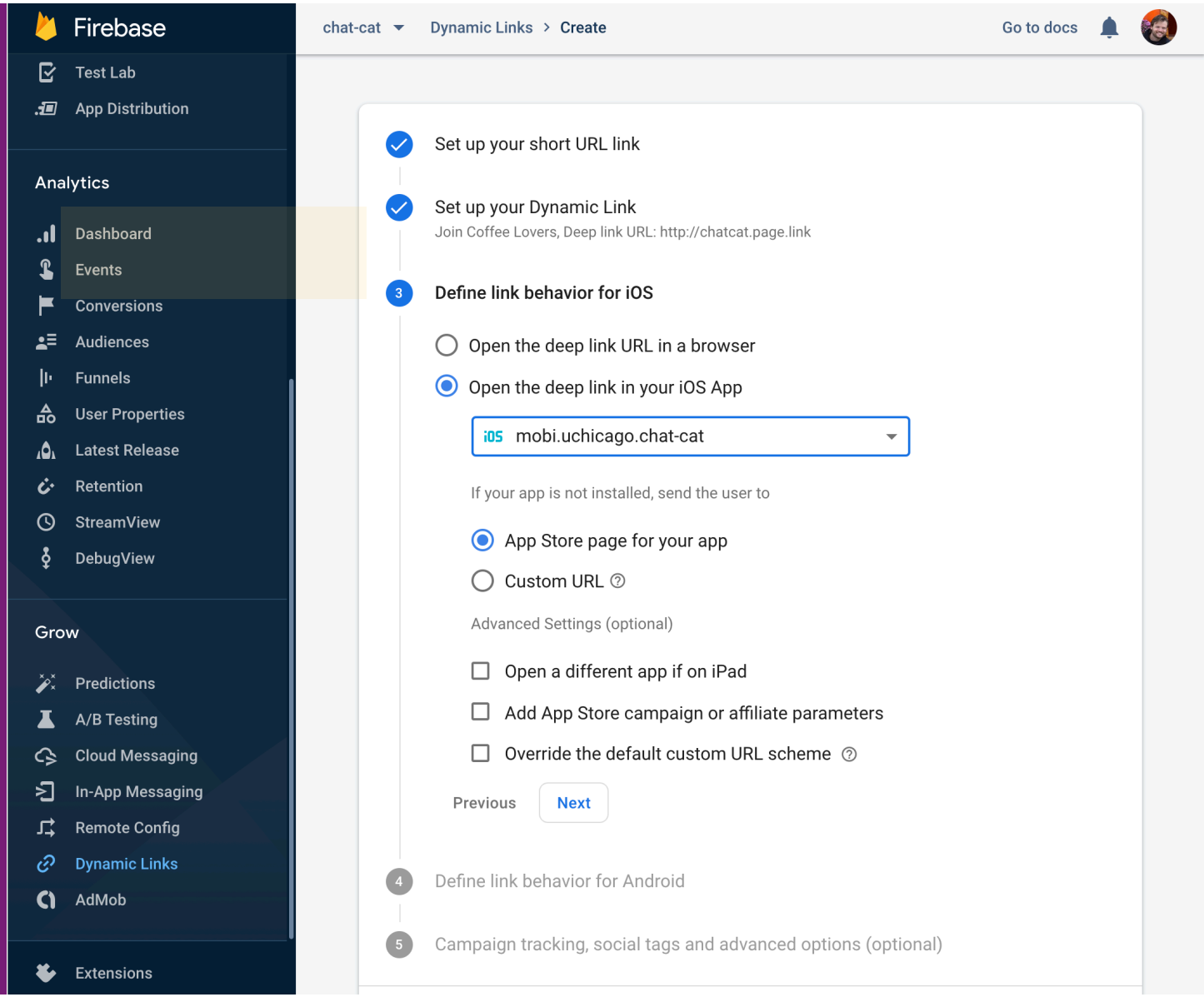
SETTING UP DYNAMICS LINKS

- Only published apps will have a App Store ID
  - Substitute your favorite app during development



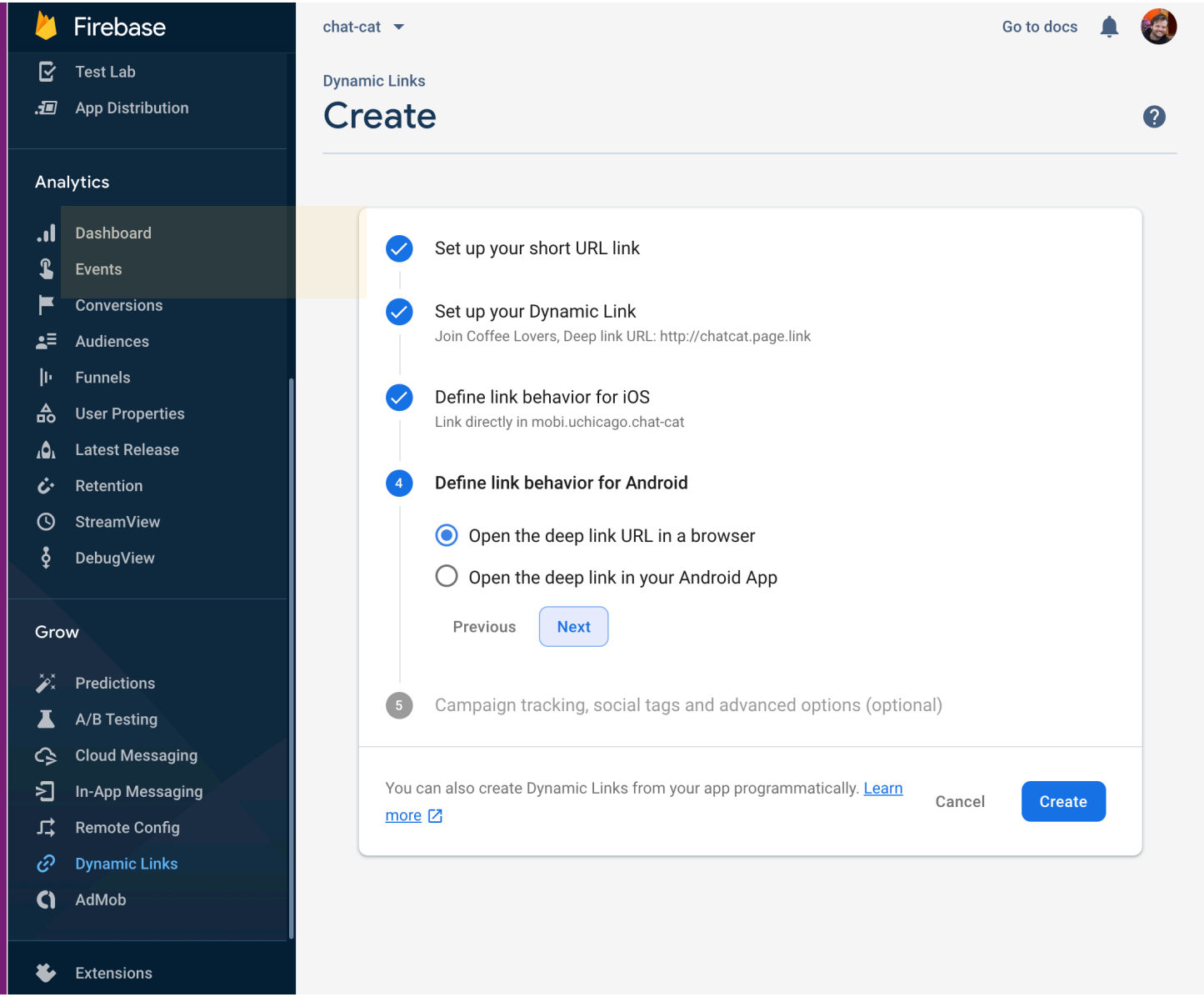
SETTING UP DYNAMICS LINKS

- Dynamic links domain
- Hosts redirects based on client

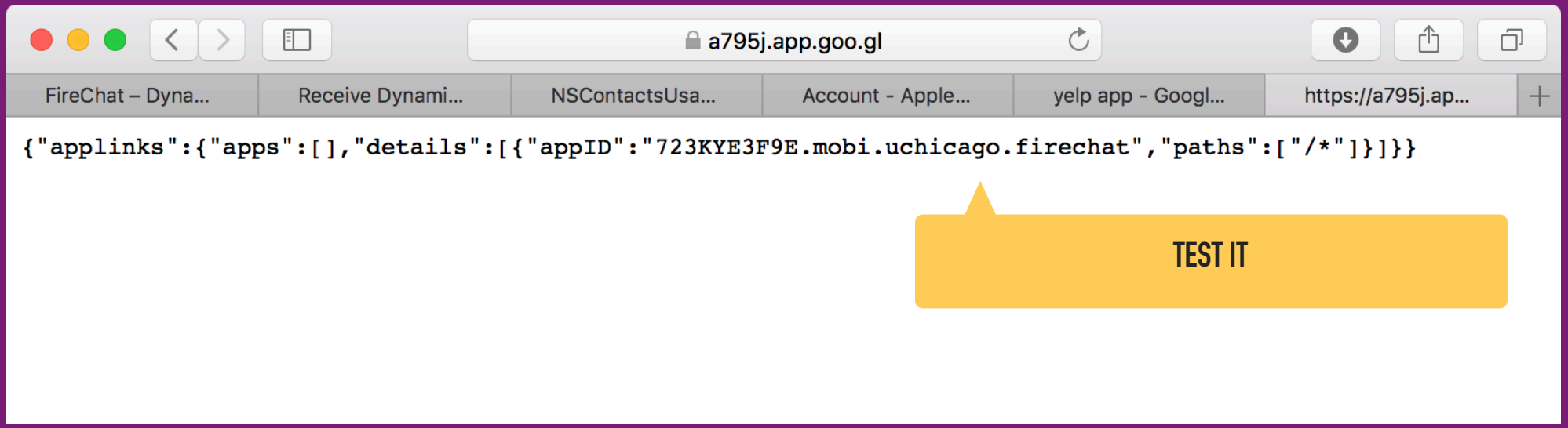


SETTING UP DYNAMICS LINKS

- Dynamic links domain
- Hosts redirects based on client



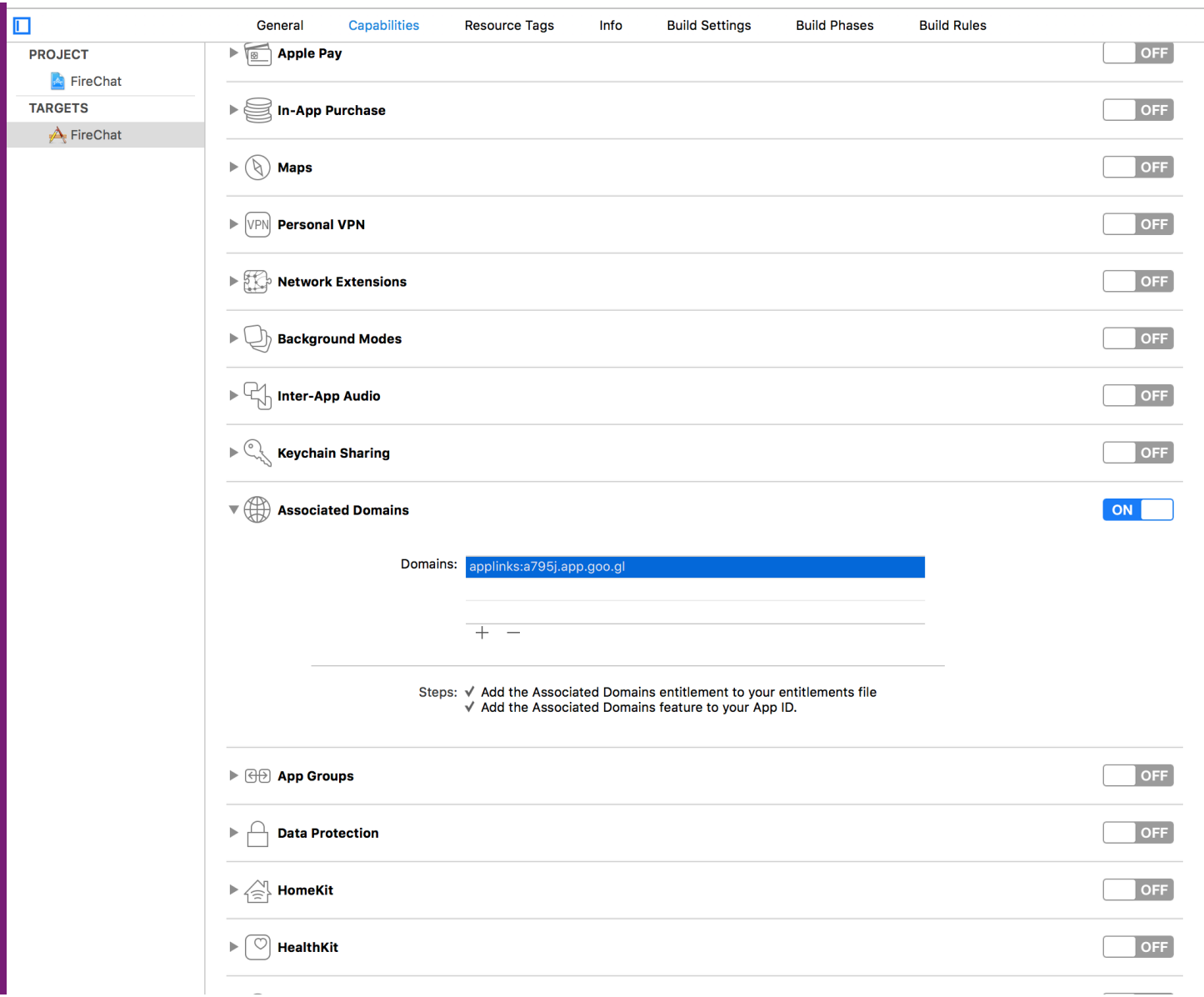
SETTING UP DYNAMICS LINKS



- https://a795j.app.goo.gl/apple-app-site-association
- This URL will redirect to my application

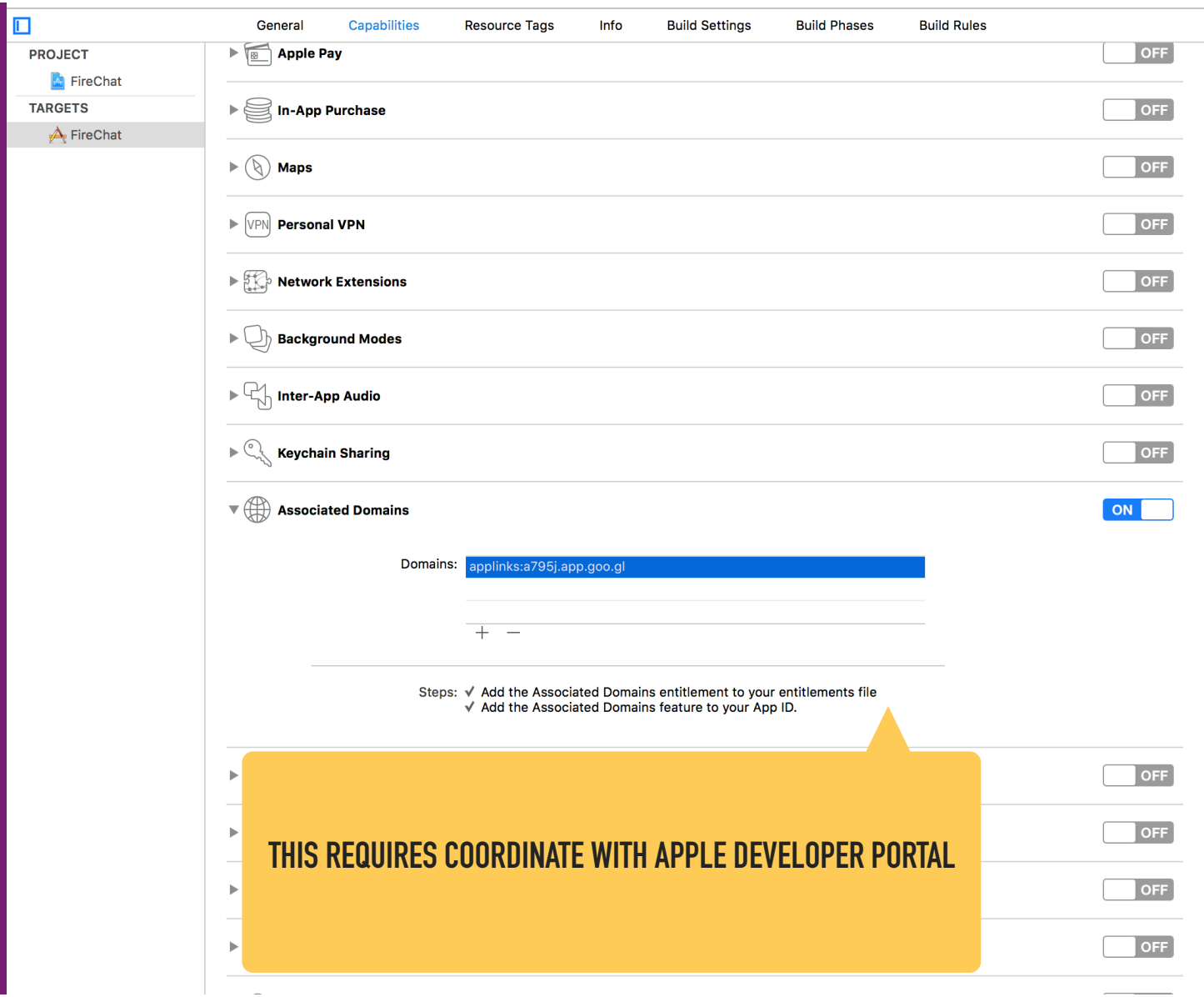
SETTING UP DYNAMICS LINKS

- Set up iOS app to handled Universal Links
- Firebase is hosting our associated domain



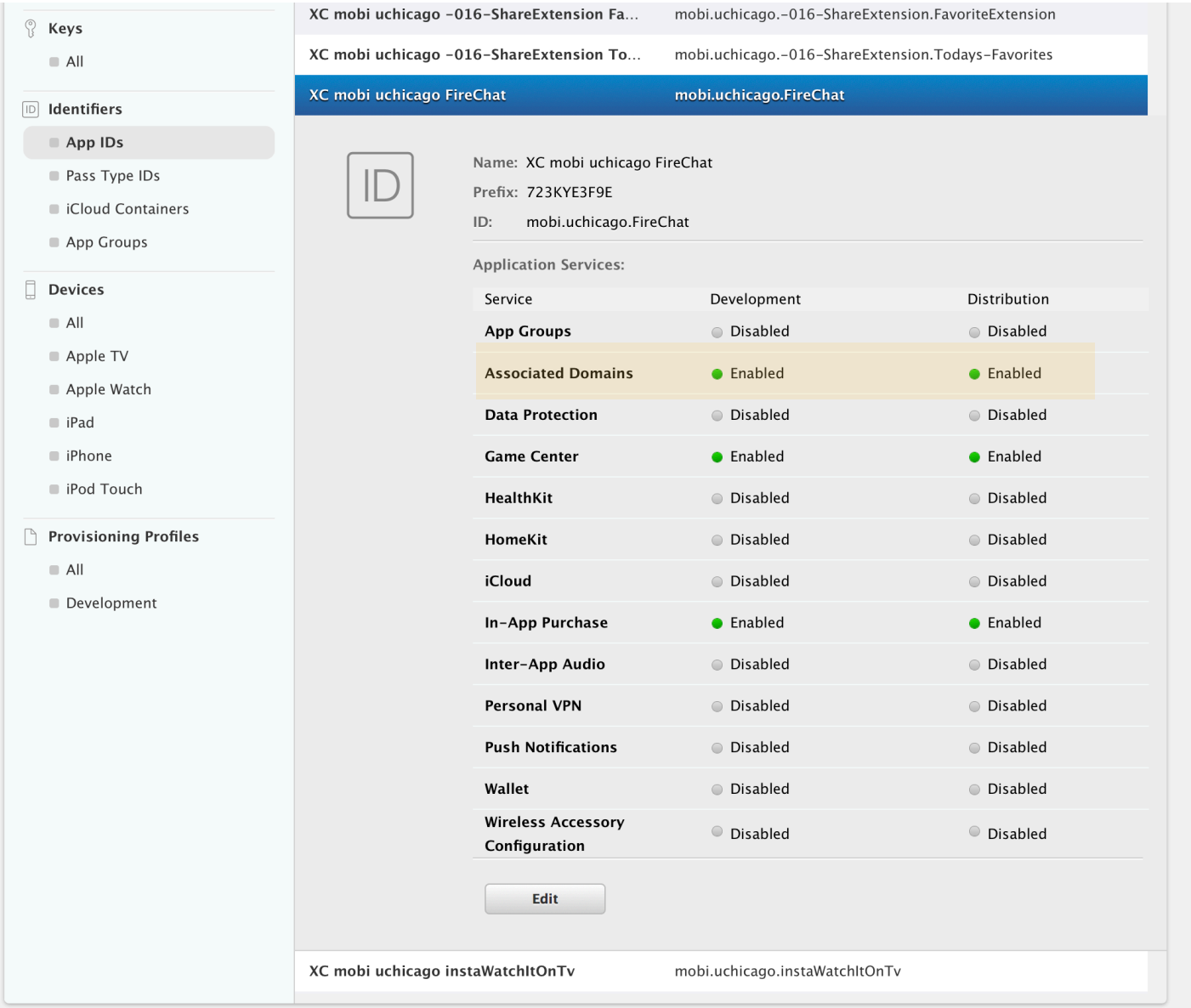
SETTING UP DYNAMICS LINKS

- Set up iOS app to handled Universal Links
- Firebase is hosting our associated domain



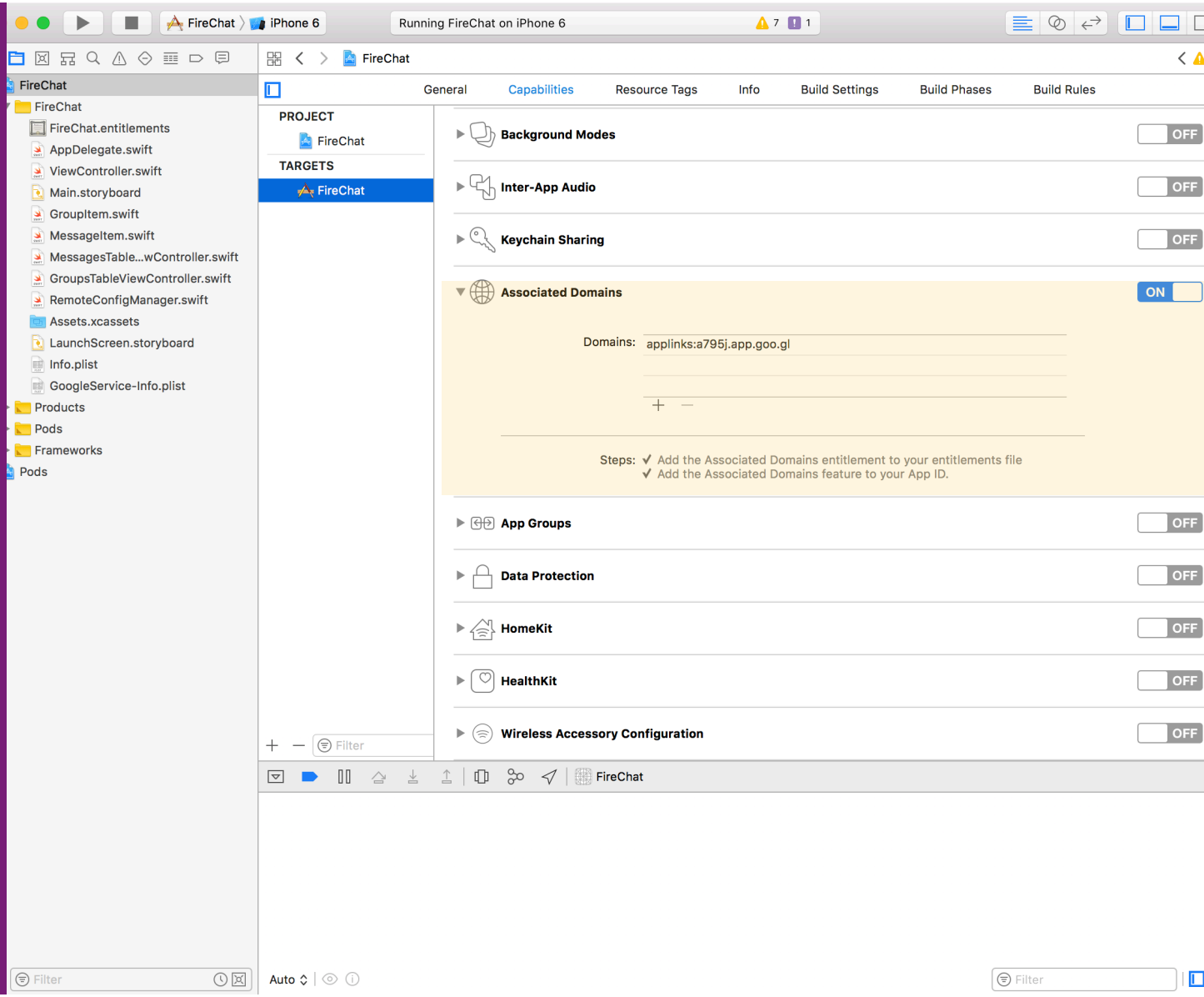
SETTING UP DYNAMICS LINKS

- Enable Associated Domains for App
- Xcode will (should) do this automatically for you

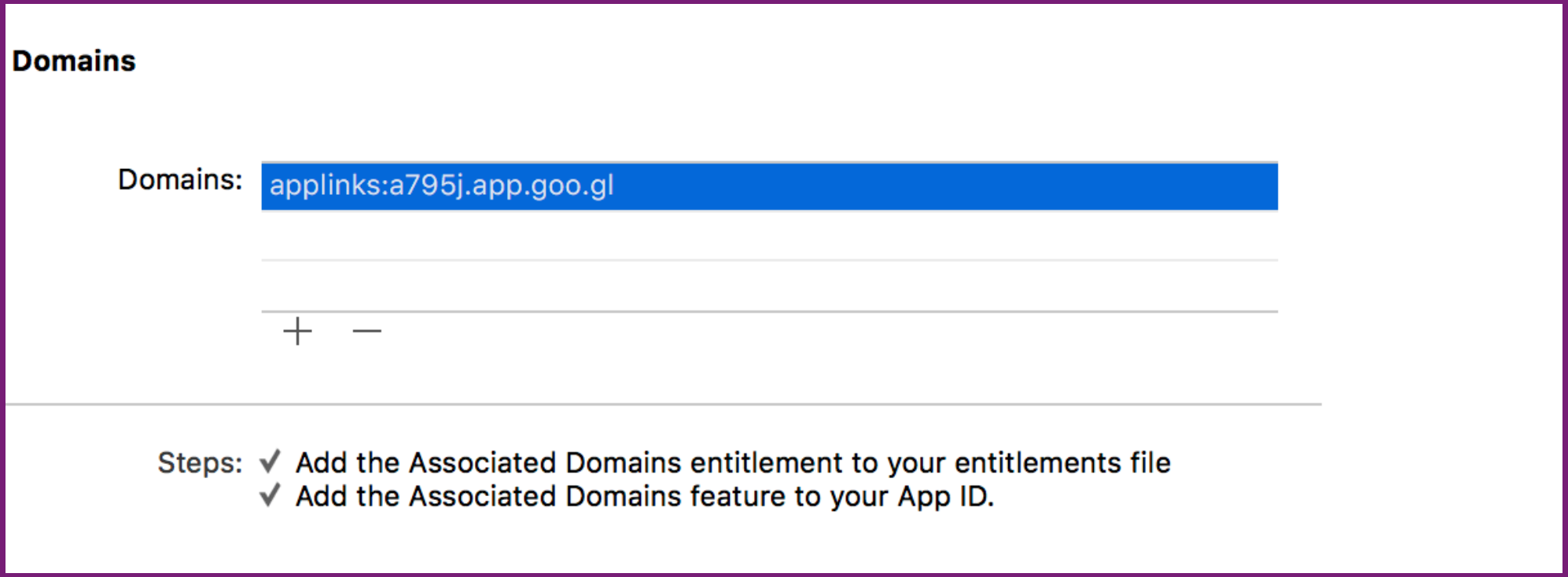


SETTING UP DYNAMICS LINKS

- Enable Associated Domains for App
- Xcode will (should) do this automatically for you



SETTING UP DYNAMICS LINKS



- https://a795j.app.goo.gl

SETTING UP DYNAMICS LINKS

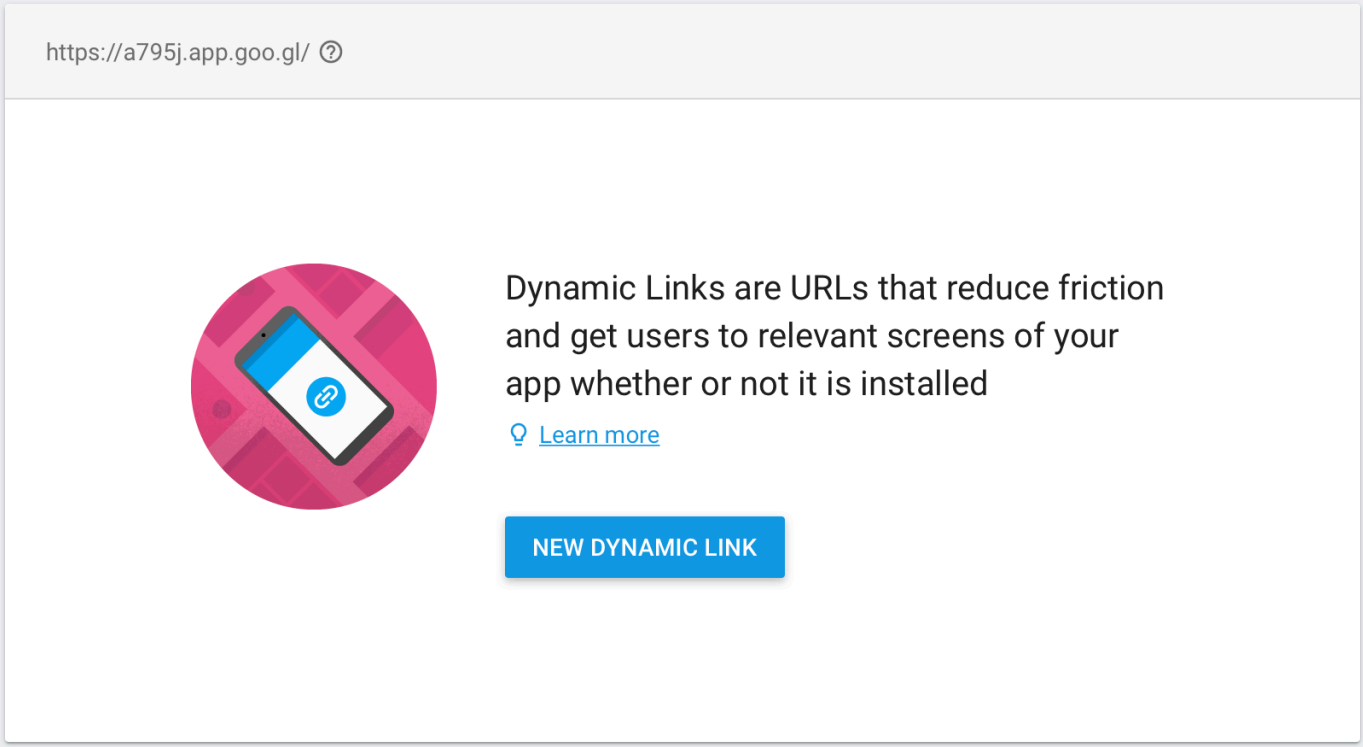
TEST ON DEVICE

DYNAMIC LINKS NEED TO BE CLICKED ON, THEY DON'T WORK BY TYPING ADDRESS IN TO BROWSER. WORKAROUNDS: SEND AS TEXT MESSAGE, PUT IN NOTE, ETC.

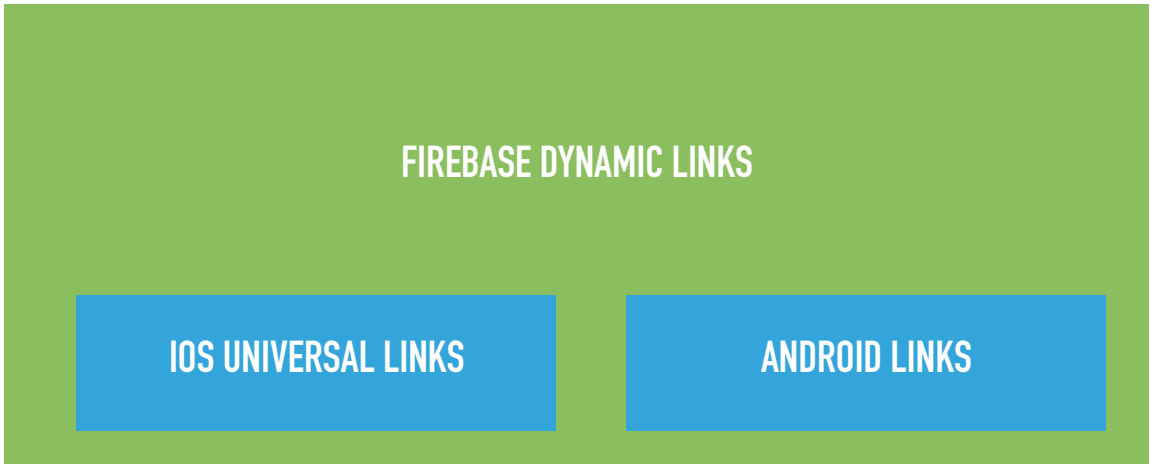
CREATING A DYNAMIC LINK

CREATING A DYNAMIC LINK

- Set up iOS app to handled Universal Links
- Firebase is hosting our associated domain

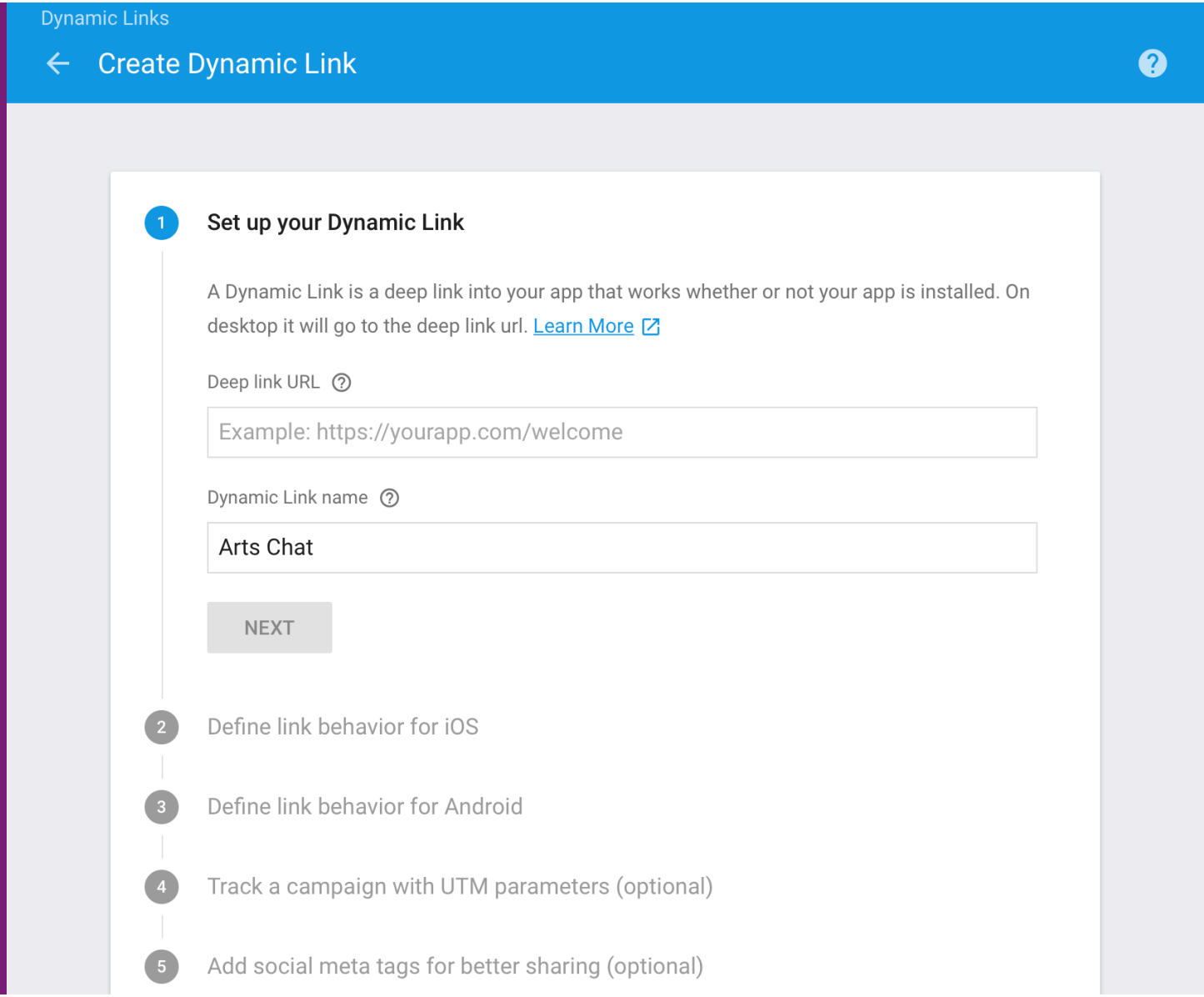


CREATING A DYNAMIC LINK



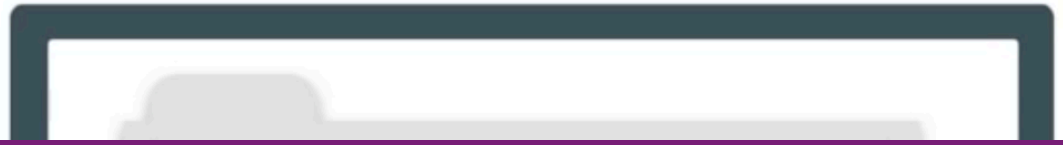
CREATING A DYNAMIC LINK

- Set up Dynamic link in the console



CREATING A DYNAMIC LINK

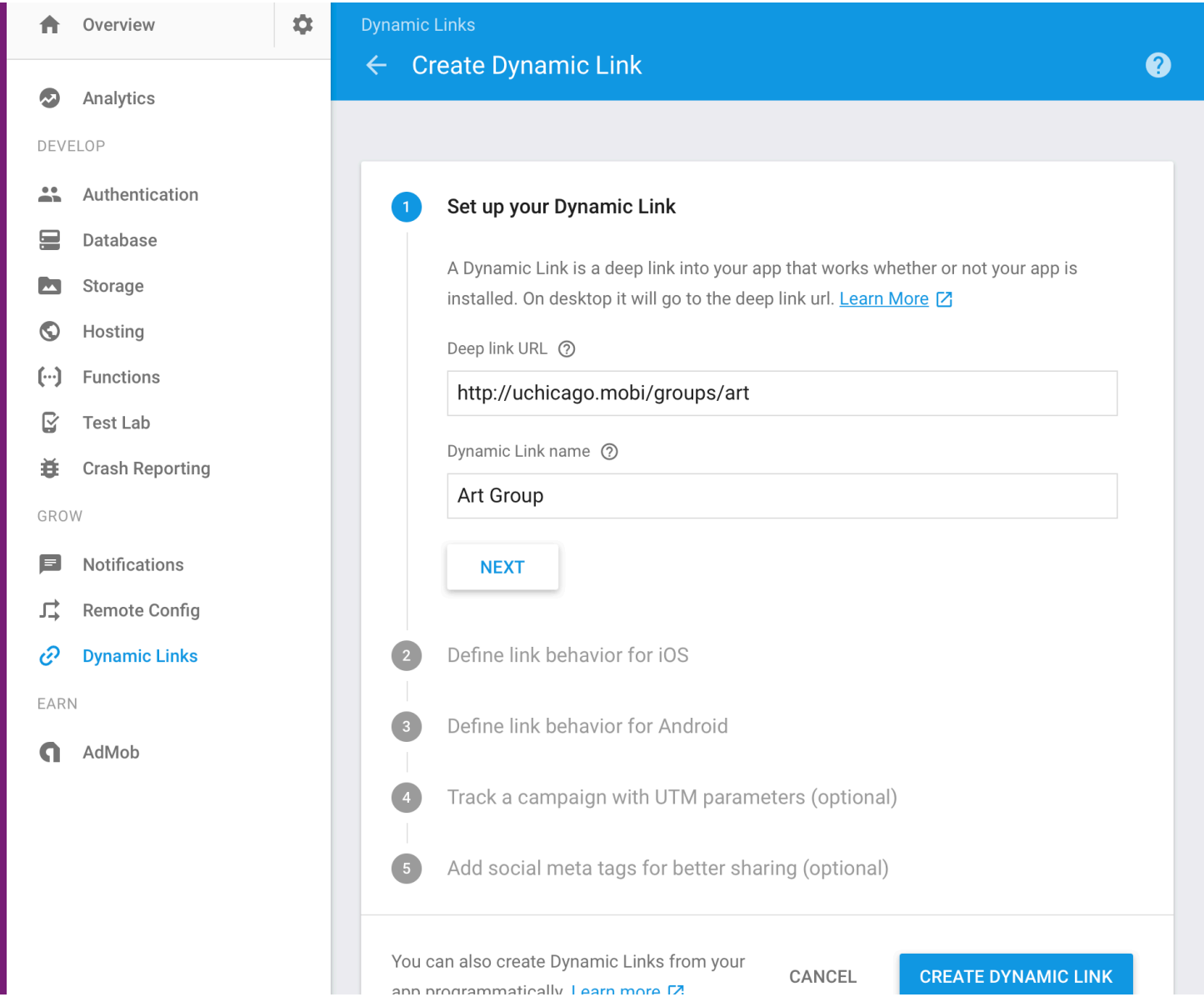
```
https://abcde.app.goo.gl/?
link=http://moviereviews.example.com/reviewID%eD42663
&isi=48151632&ibi=com.example.moviereviews
```



- All data is passed in through URL parameters
- Allows the links to be "universal"

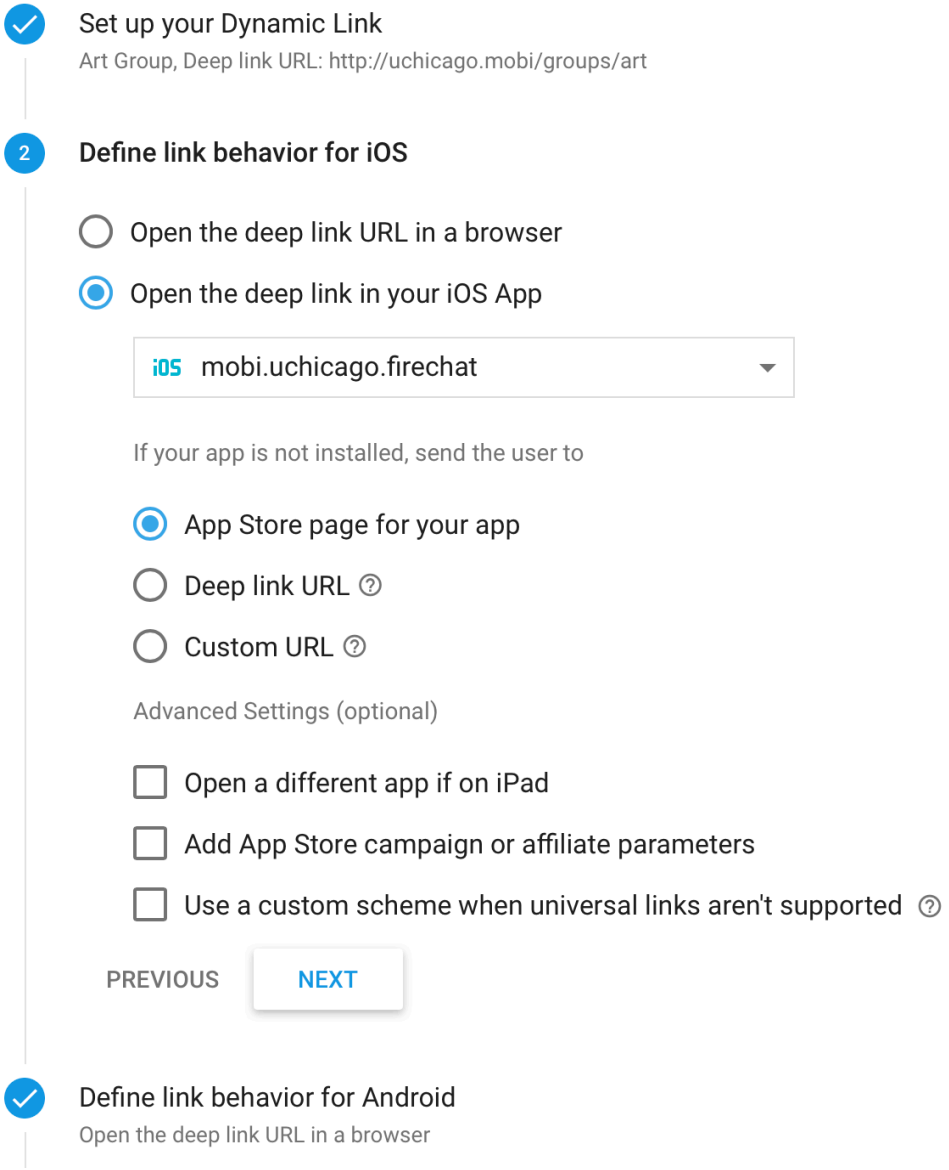
CREATING A DYNAMIC LINK

- URL contains information that will be read in the iOS app



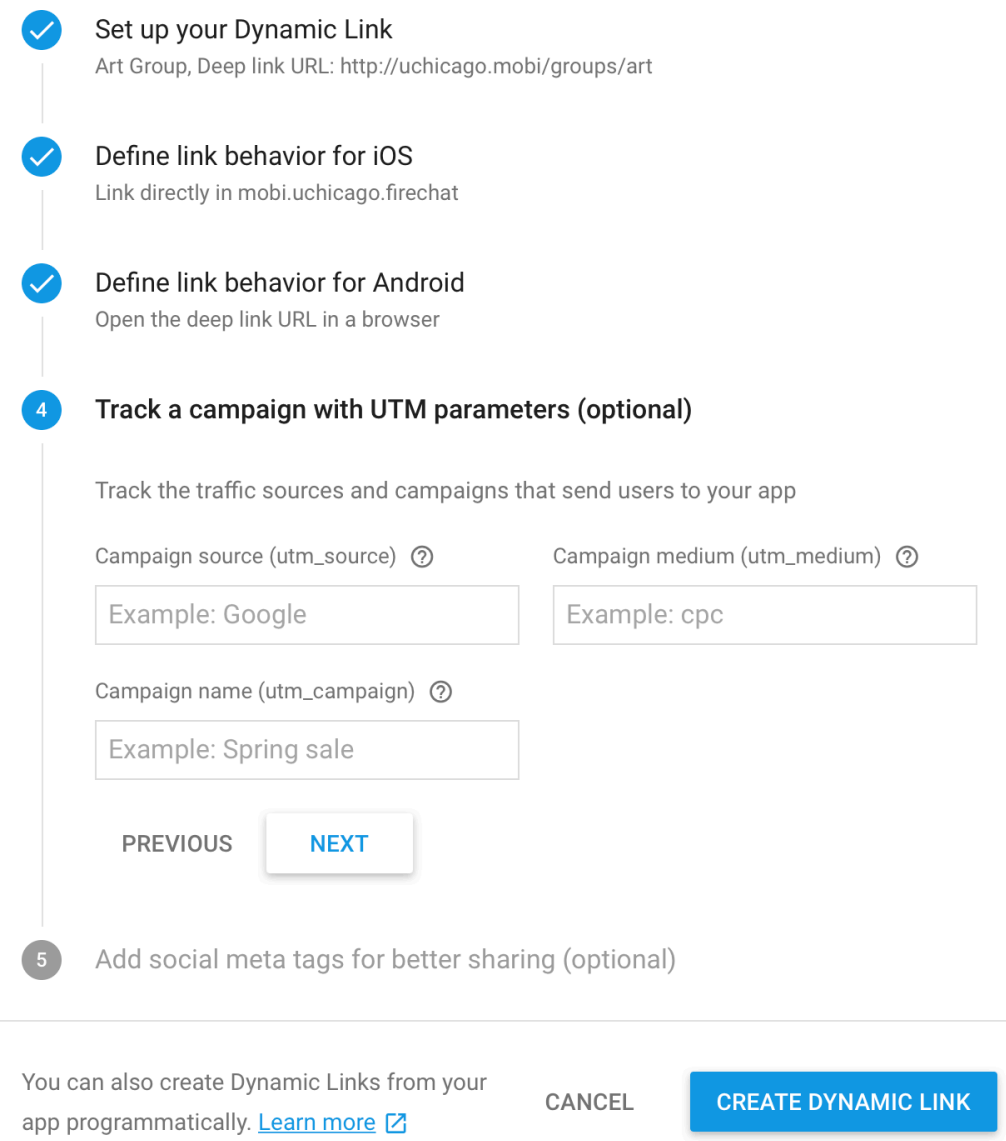
CREATING A DYNAMIC LINK

- Define the behavior of the link in the selected application



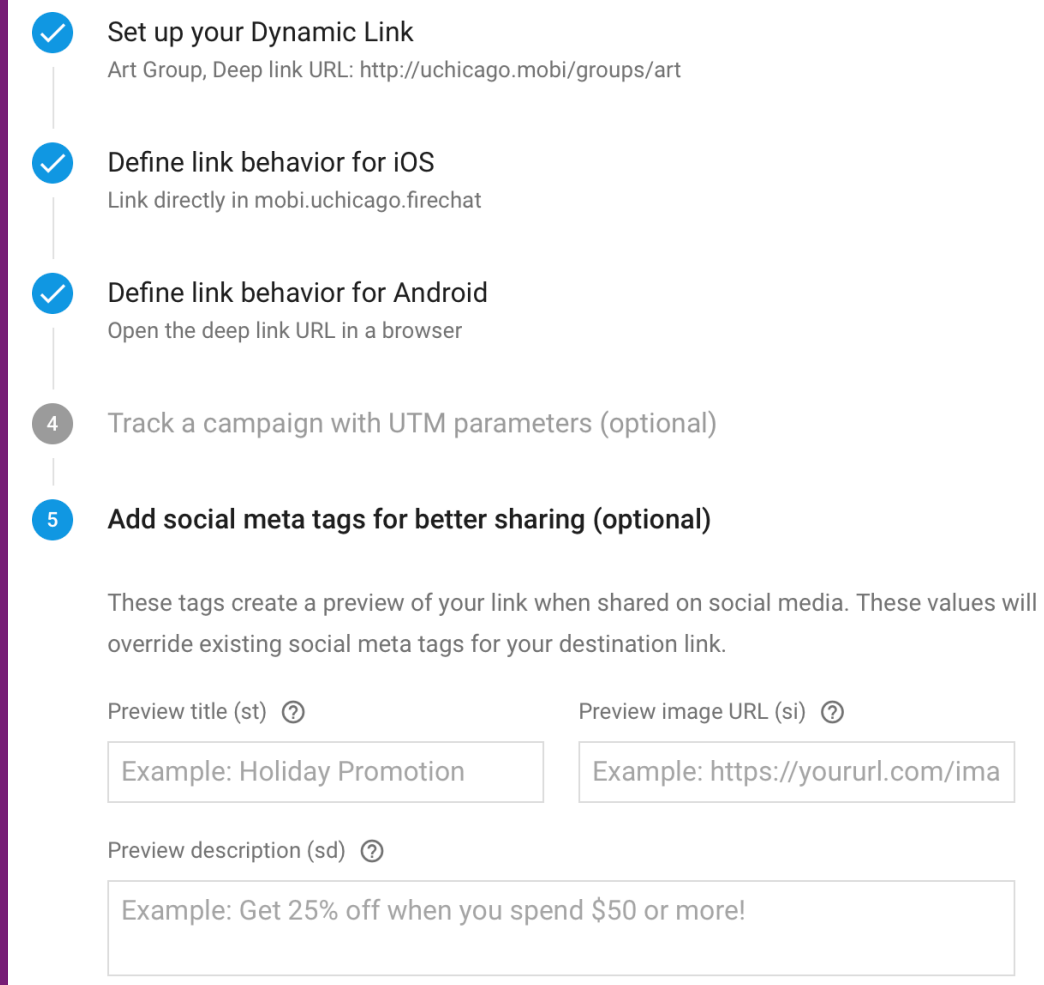
CREATING A DYNAMIC LINK

- Analytics tags
- Mirrors Google Analytics campaign tracking

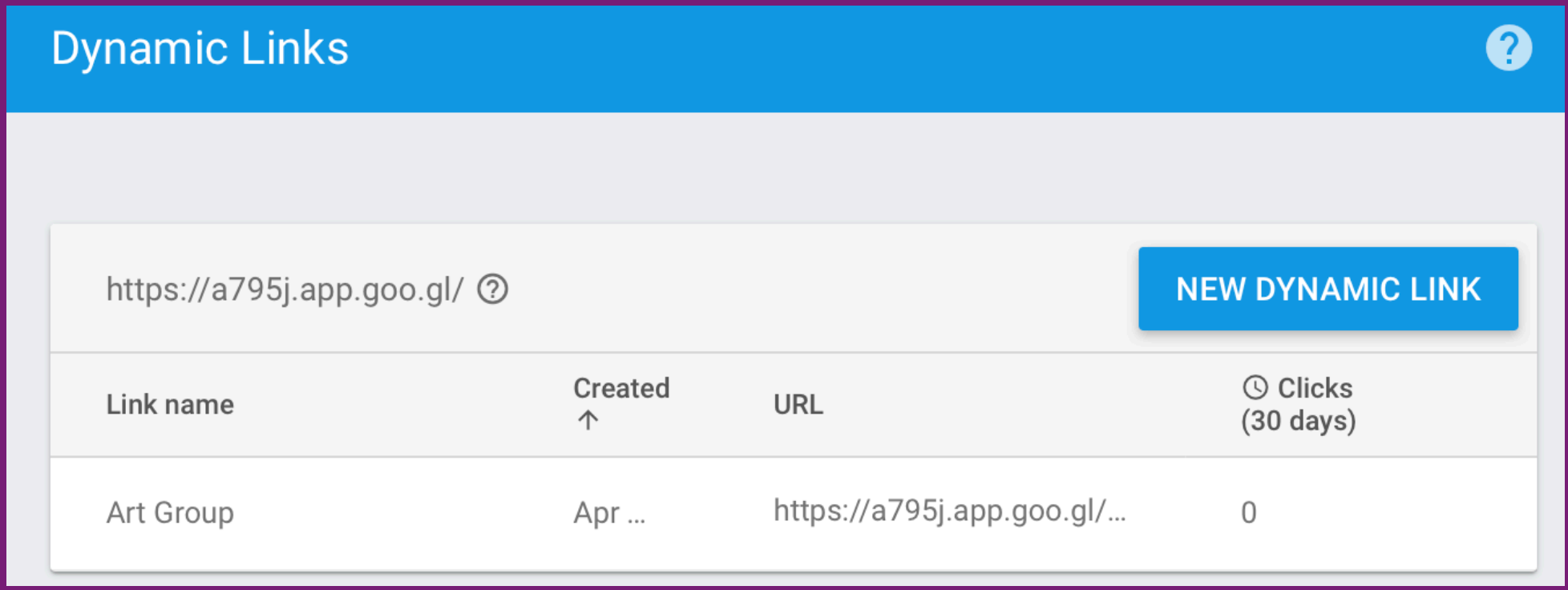


CREATING A DYNAMIC LINK

- Social media tags and data



CREATING A DYNAMIC LINK



- Your link 🎉

CREATING A DYNAMIC LINK

Link name

Art Group

Deep link

http://uchicago.mobi/groups/art

FALL BACK

iOS app

iOS mobi.uchicago.firechat

FULL LINK

Long Dynamic Link

https://a795j.app.goo.gl/?link=http://uchicago.mobi/groups/art&isi=284910350&ibi=mobi.uchicago.firechat

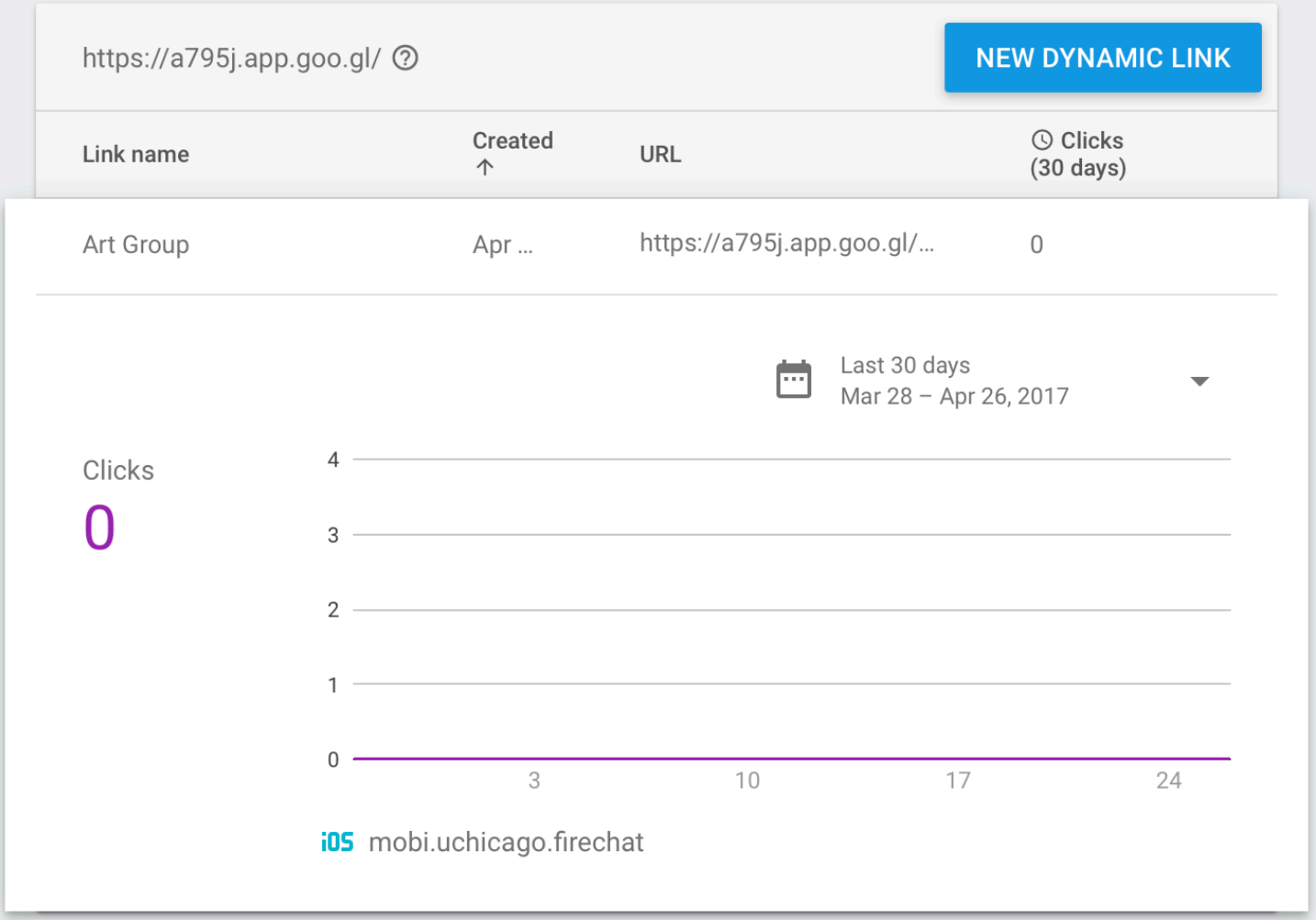
Short Dynamic Link

https://a795j.app.goo.gl/6SuK

SHORTENED LINK

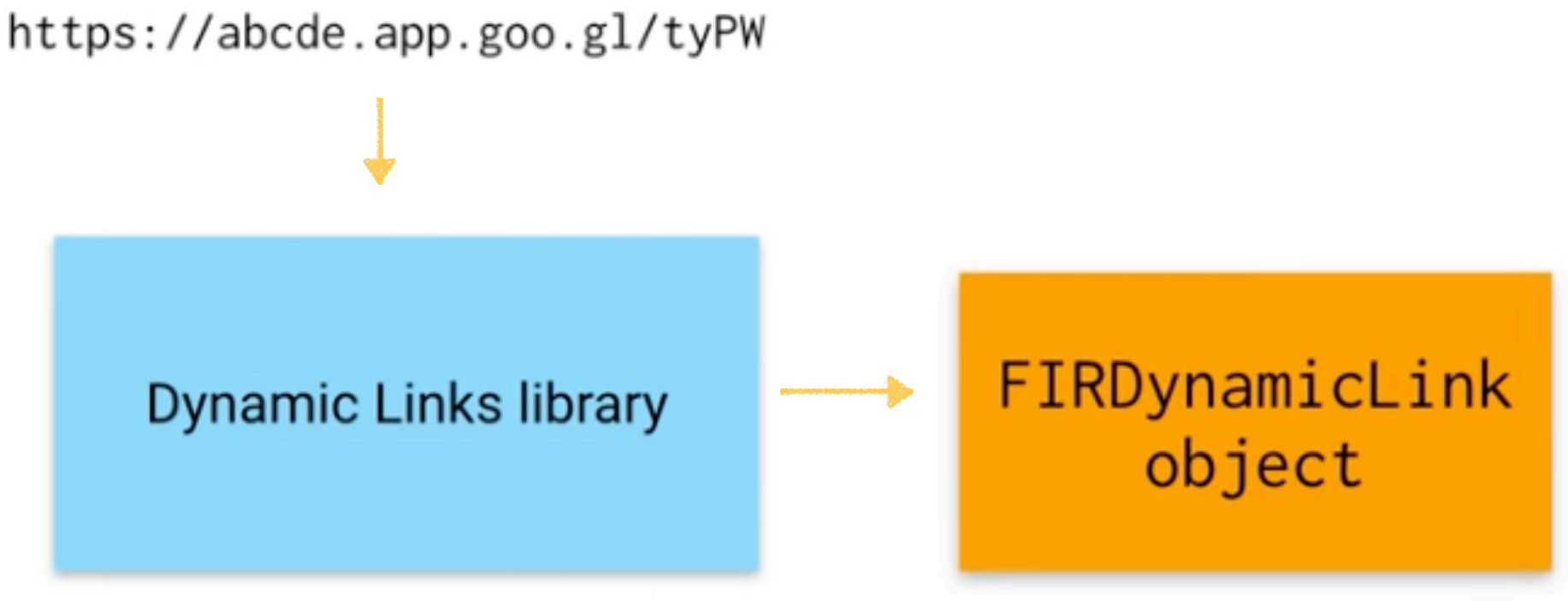
CREATING A DYNAMIC LINK

- Analytics for specific link



HANDLING DYNAMIC LINKS

HANDLING DYNAMIC LINKS

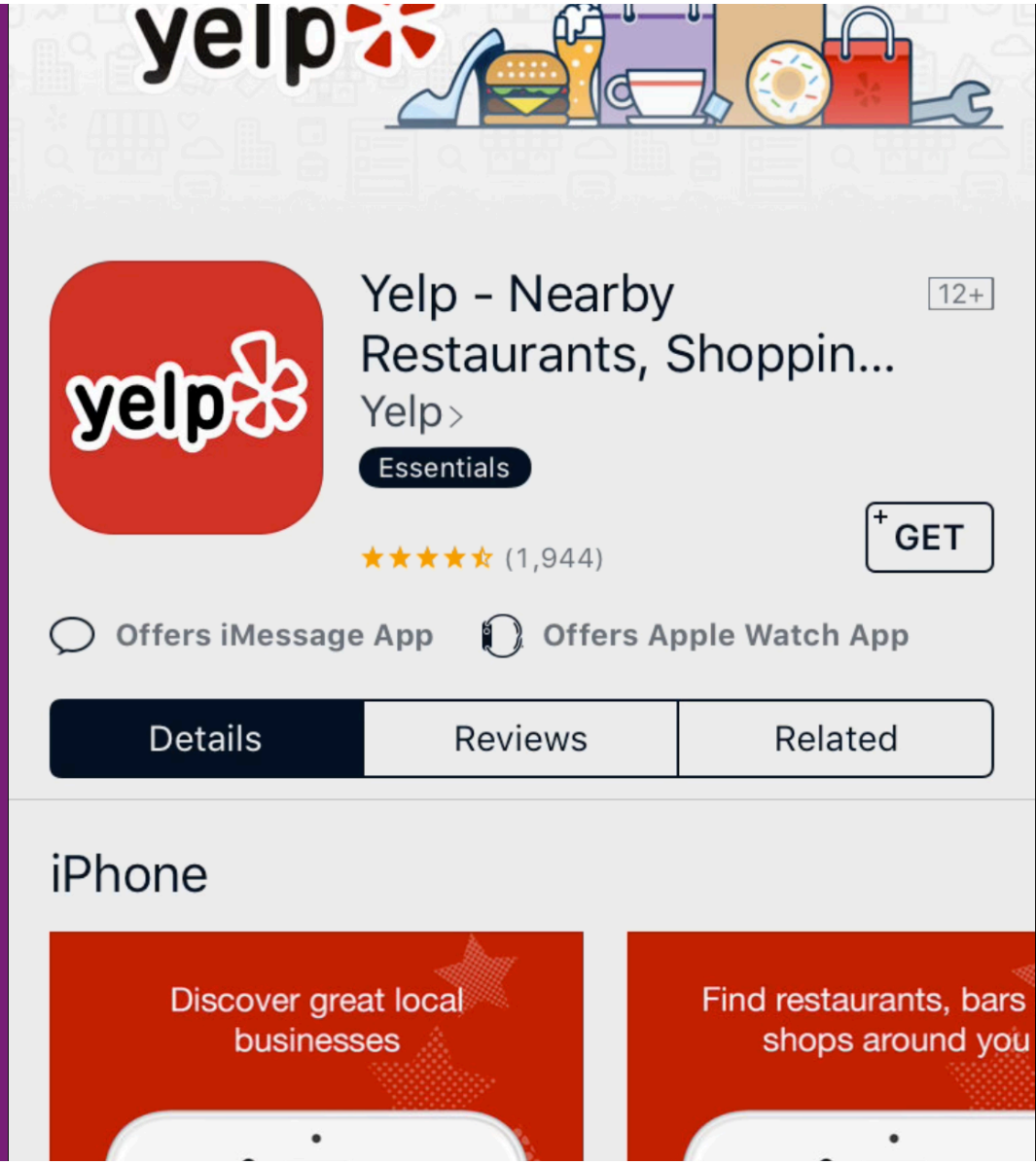


HANDLING DYNAMIC LINKS

```
# Uncomment the next line to define a global platform for your project
 platform :ios, '9.0'

target 'FireChat' do
  # Comment the next line if you're not using Swift and don't want to use dynamic frameworks
  use_frameworks!

  # Pods for FireChat
  pod 'Firebase/Core'
  pod 'Firebase/Database'

  pod 'Firebase/Auth'
  pod 'GoogleSignIn'

  pod 'Firebase/Crash'

  pod 'Firebase/Storage'
  pod 'Firebase/Invites'

  pod 'Firebase/DynamicLinks'

  pod 'SDWebImage'

end
```

HANDLING DYNAMIC LINKS

- Two different flows
  - Custom URL Scheme
    - First install
    - Will direct to the app store id that you specified
  - Deep link
    - iOS functions
    - Will launch the app and pass the link



HANDLING DYNAMIC LINKS

```
// Handle dynamic links
@available(iOS 8.0, *)
func application(_ application: UIApplication,
                 continue userActivity: NSUserActivity,
                 restorationHandler: @escaping ([Any]?) -> Void) -> Bool {
  guard let dynamicLinks = FIRDynamicLinks.dynamicLinks() else {
    return false
  }
  let handled = dynamicLinks.handleUniversalLink(userActivity.webpageURL!) { (dynamiclink, error) in
    if let dynamiclink = dynamiclink, let _ = dynamiclink.url {
      self.handleIcomingLink(dynamiclink)
    } else {
      print("Error: \(String(describing: error?.localizedDescription))")
    }
  }
  return handled
}
```

- In App Delegate detect open from dynamic links

HANDLING DYNAMIC LINKS

```
// Open the app from URL callback (ie. mobi.uchicago.firechat) during
// Google authentication OR from an intial install on a deep link
@available(iOS 9.0, *)
func application(_ application: UIApplication,
                 open url: URL,
                 options: [UIApplicationOpenURLOptionsKey : Any]) -> Bool {

  // Try to handle the dynamic link first
  let dynamicLink = FIRDynamicLinks.dynamicLinks()?.dynamicLink(fromCustomSchemeURL: url)
  if dynamicLink != nil {
    // Handle the deep link. For example, show the deep-linked content or
    // apply a promotional offer to the user's account.
    self.handleIcomingLink(dynamicLink!)
    print("🔗🔗🔗 Deep link of first launch or old device")
    return true
  }

  // Handle the Google ID Authentication
  return GIDSignIn.sharedInstance().handle(url,
                                           sourceApplication:options[UIApplicationOpenURLOptionsKey.sourceApplication] as? String,
                                           annotation: [:])
}
```

- In App Delegate, handle a first launch from deep link

HANDLING DYNAMIC LINKS

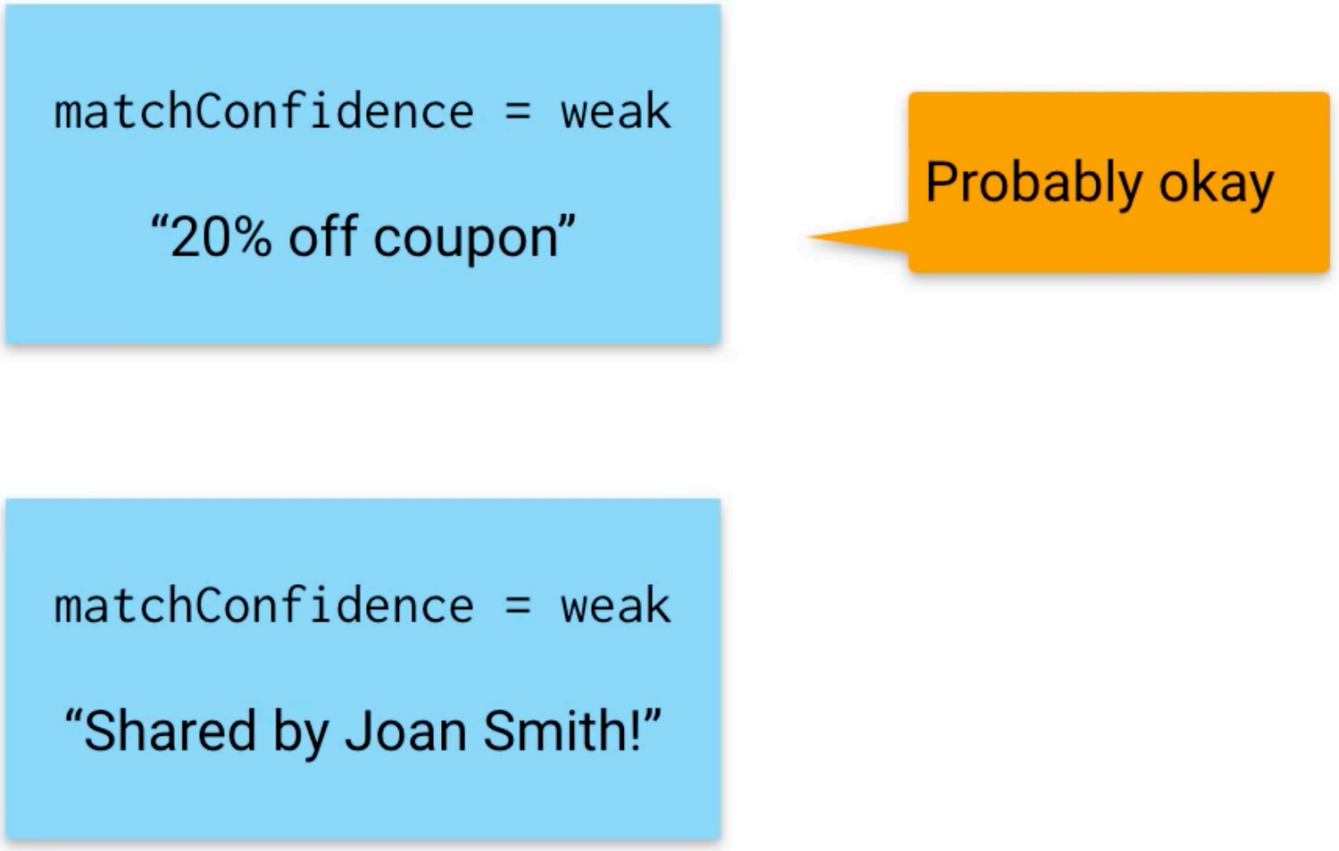
```
/// Take a dynamic link and parse it into ints components
func handleIcomingLink(_ dynamicLink: FIRDynamicLink) {
  print("🔗🔗🔗 DynamicLink: \(dynamicLink)")
  guard let pathComponents = dynamicLink.url?.pathComponents  else { return }
  for component in pathComponents {
    print("\t>> component: \(component)")
  }
}
```

```
DynamicLink: <FIRDynamicLink: 0x17024a4a0, url [http://uchicago.mobi/
groups/art], confidence: weak>
    >> component: /
    >> component: groups
    >> component: art
```

DO SOMETHING USEFUL HERE
ADD USER TO GROUP

HANDLING DYNAMIC LINKS

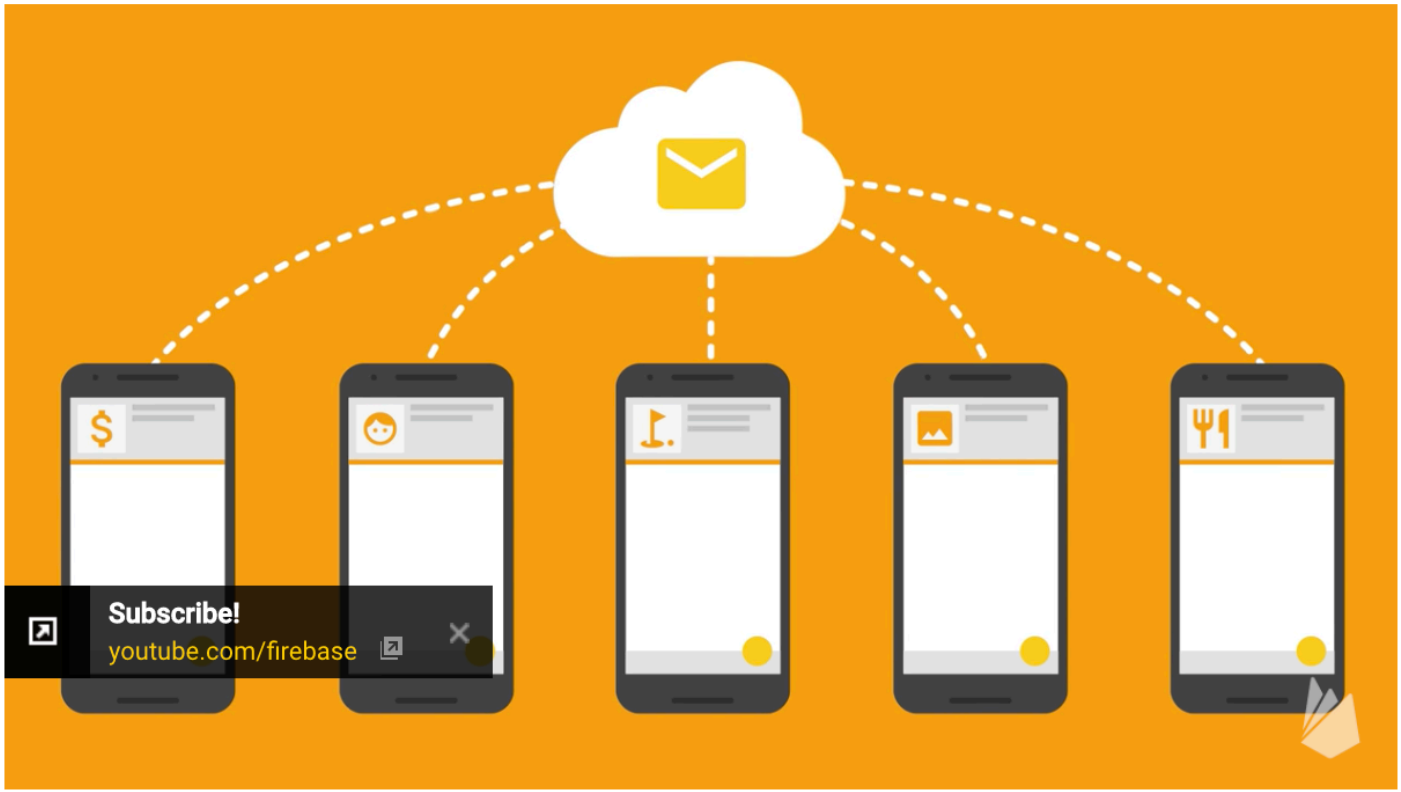
- Link confidence
- Depends on the content



FIREBASE CLOUD MESSAGING

FIREBASE CLOUD MESSAGING

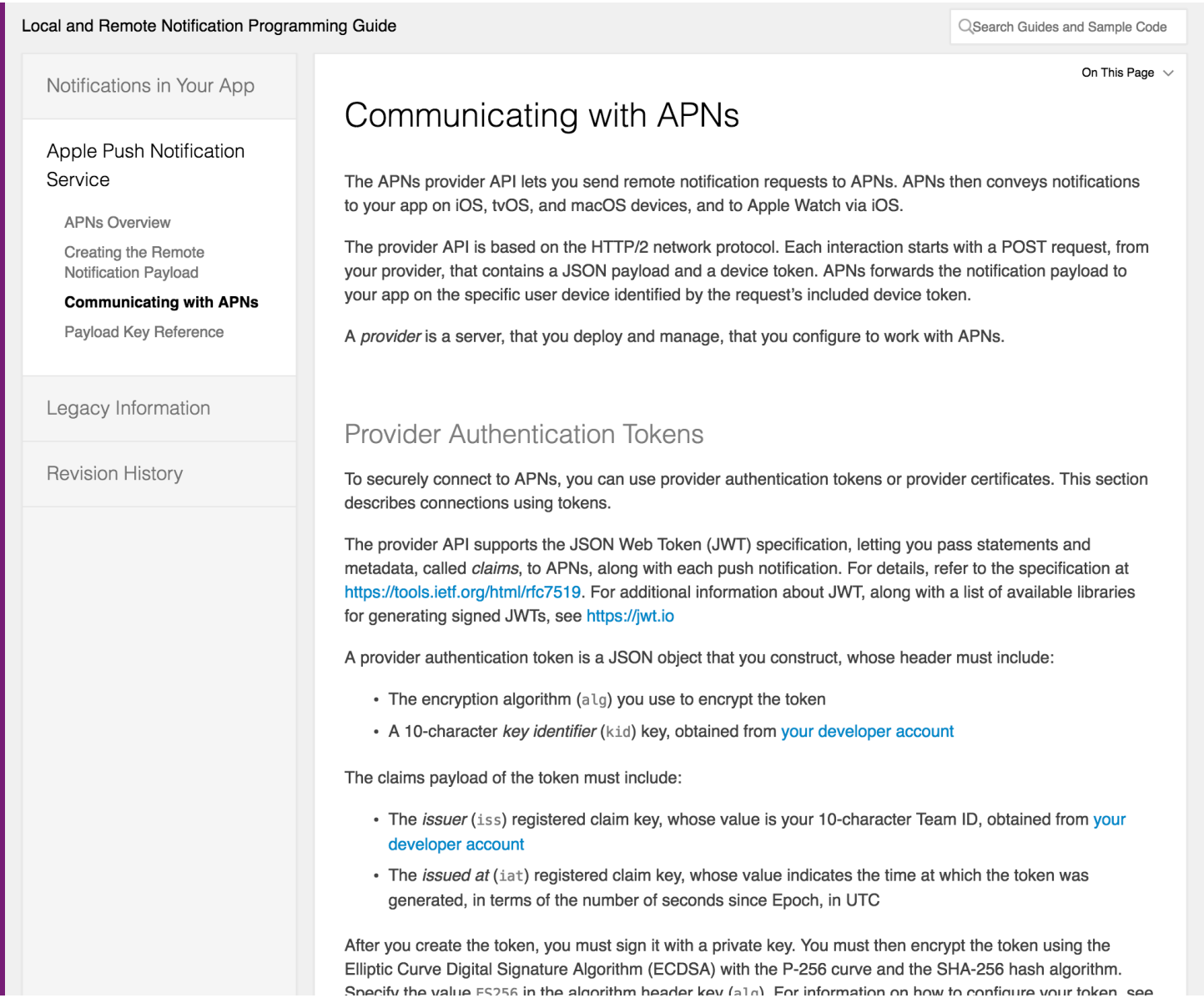
- https://youtu.be/sioEY4tWmLI?list=PLl-K7zZEsYLmOF_07IayrTntevxtbUxDL



SETTING UP FOR APNS

SETTING UP APNS

- Messaging uses native APNS services to deliver messages
  - Default



Local and Remote Notification Programming Guide

Search Guides and Sample Code

Notifications in Your App

Apple Push Notification Service

APNs Overview

Creating the Remote Notification Payload

Communicating with APNs

Payload Key Reference

Legacy Information

Revision History

On This Page

Communicating with APNs

The APNs provider API lets you send remote notification requests to APNs. APNs then conveys notifications to your app on iOS, tvOS, and macOS devices, and to Apple Watch via iOS.

The provider API is based on the HTTP/2 network protocol. Each interaction starts with a POST request, from your provider, that contains a JSON payload and a device token. APNs forwards the notification payload to your app on the specific user device identified by the request's included device token.

A *provider* is a server, that you deploy and manage, that you configure to work with APNs.

Provider Authentication Tokens

To securely connect to APNs, you can use provider authentication tokens or provider certificates. This section describes connections using tokens.

The provider API supports the JSON Web Token (JWT) specification, letting you pass statements and metadata, called *claims*, to APNs, along with each push notification. For details, refer to the specification at https://tools.ietf.org/html/rfc7519. For additional information about JWT, along with a list of available libraries for generating signed JWTs, see https://jwt.io

A provider authentication token is a JSON object that you construct, whose header must include:

- The encryption algorithm (`alg`) you use to encrypt the token
- A 10-character *key identifier* (`kid`) key, obtained from your developer account

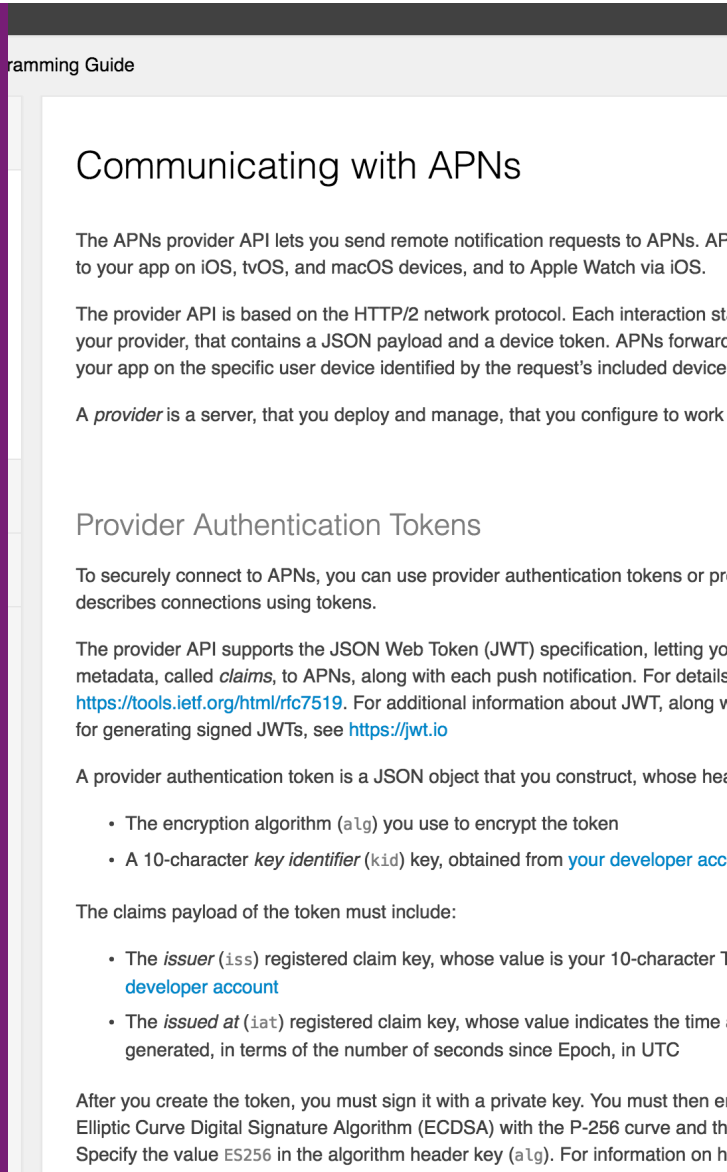
The claims payload of the token must include:

- The *issuer* (`iss`) registered claim key, whose value is your 10-character Team ID, obtained from your developer account
- The *issued at* (`iat`) registered claim key, whose value indicates the time at which the token was generated, in terms of the number of seconds since Epoch, in UTC

After you create the token, you must sign it with a private key. You must then encrypt the token using the Elliptic Curve Digital Signature Algorithm (ECDSA) with the P-256 curve and the SHA-256 hash algorithm. Specify the value `ES256` in the algorithm header key (`alg`). For information on how to configure your token, see

SETTING UP APNS

- 1. Set up app in Xcode
- 2. Create app in developer portal
- 3. Add certificates for authentication
- 4. Upload this certificate to provider (eg. Firebase)



ramming Guide

Communicating with APNs

The APNs provider API lets you send remote notification requests to APNs. AP
to your app on iOS, tvOS, and macOS devices, and to Apple Watch via iOS.

The provider API is based on the HTTP/2 network protocol. Each interaction st
your provider, that contains a JSON payload and a device token. APNs forwarc
your app on the specific user device identified by the request's included device

A *provider* is a server, that you deploy and manage, that you configure to work

Provider Authentication Tokens

To securely connect to APNs, you can use provider authentication tokens or pr
describes connections using tokens.

The provider API supports the JSON Web Token (JWT) specification, letting yo
metadata, called *claims*, to APNs, along with each push notification. For details
https://tools.ietf.org/html/rfc7519. For additional information about JWT, along w
for generating signed JWTs, see https://jwt.io

A provider authentication token is a JSON object that you construct, whose hea

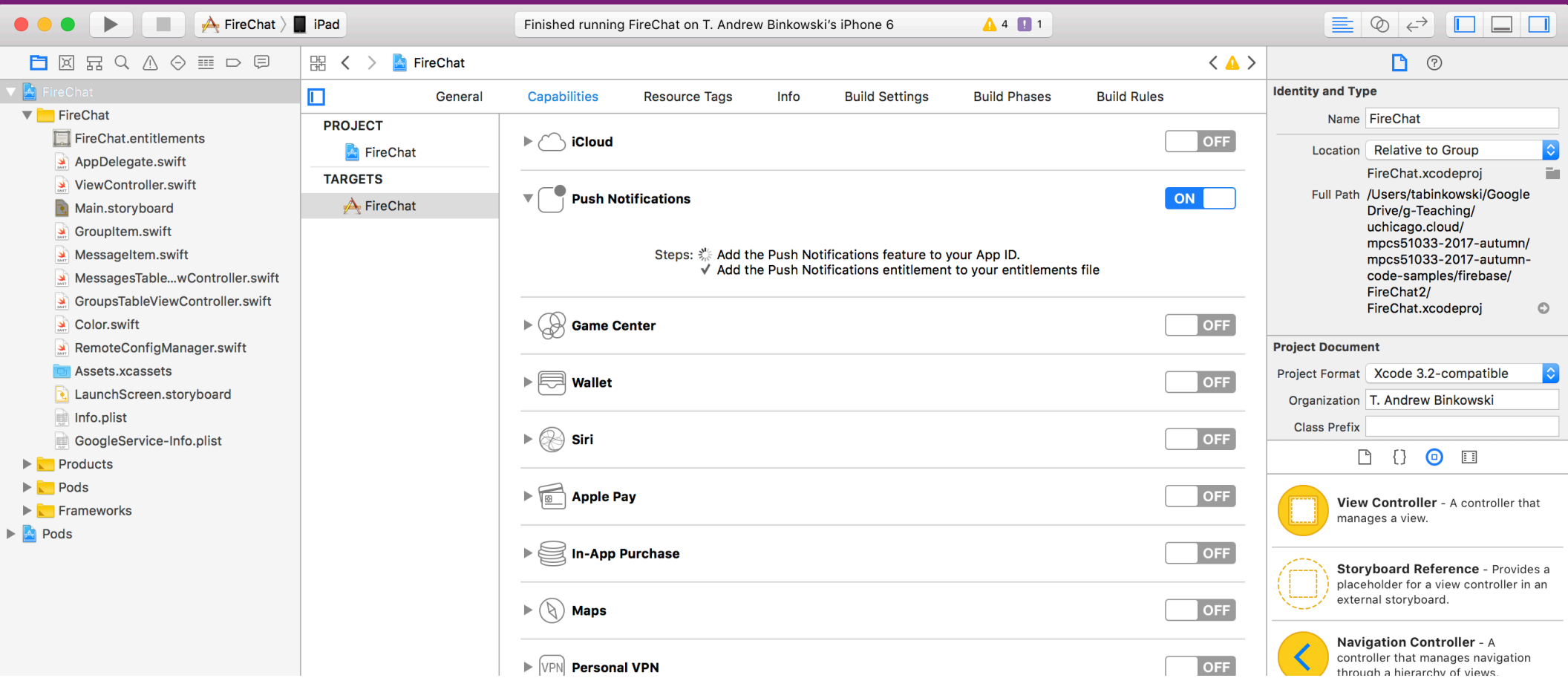
- The encryption algorithm (`alg`) you use to encrypt the token
- A 10-character *key identifier* (`kid`) key, obtained from your developer acc

The claims payload of the token must include:

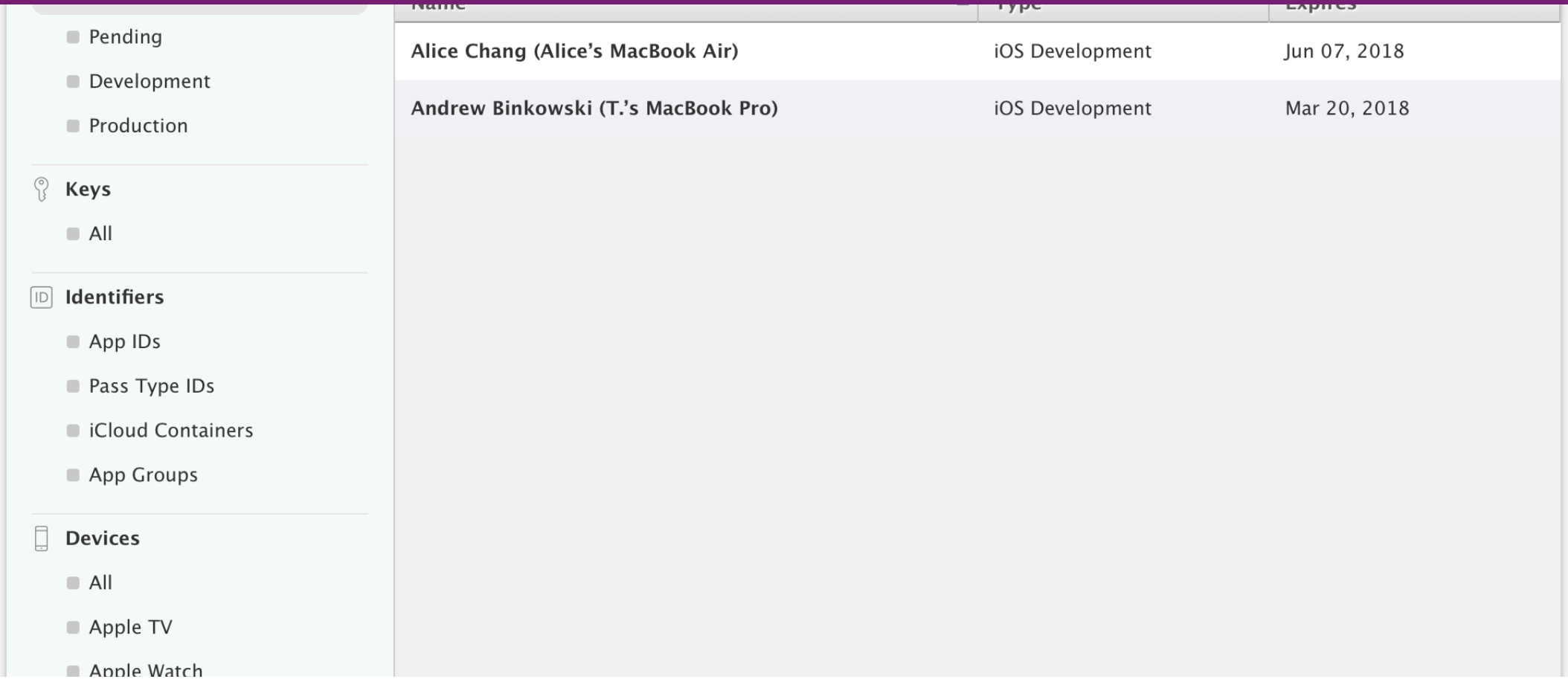
- The *issuer* (`iss`) registered claim key, whose value is your 10-character T
  developer account
- The *issued at* (`iat`) registered claim key, whose value indicates the time
  generated, in terms of the number of seconds since Epoch, in UTC

After you create the token, you must sign it with a private key. You must then e
Elliptic Curve Digital Signature Algorithm (ECDSA) with the P-256 curve and th
Specify the value `ES256` in the algorithm header key (`alg`). For information on h
"Configure push notifications" in Xcode Help.

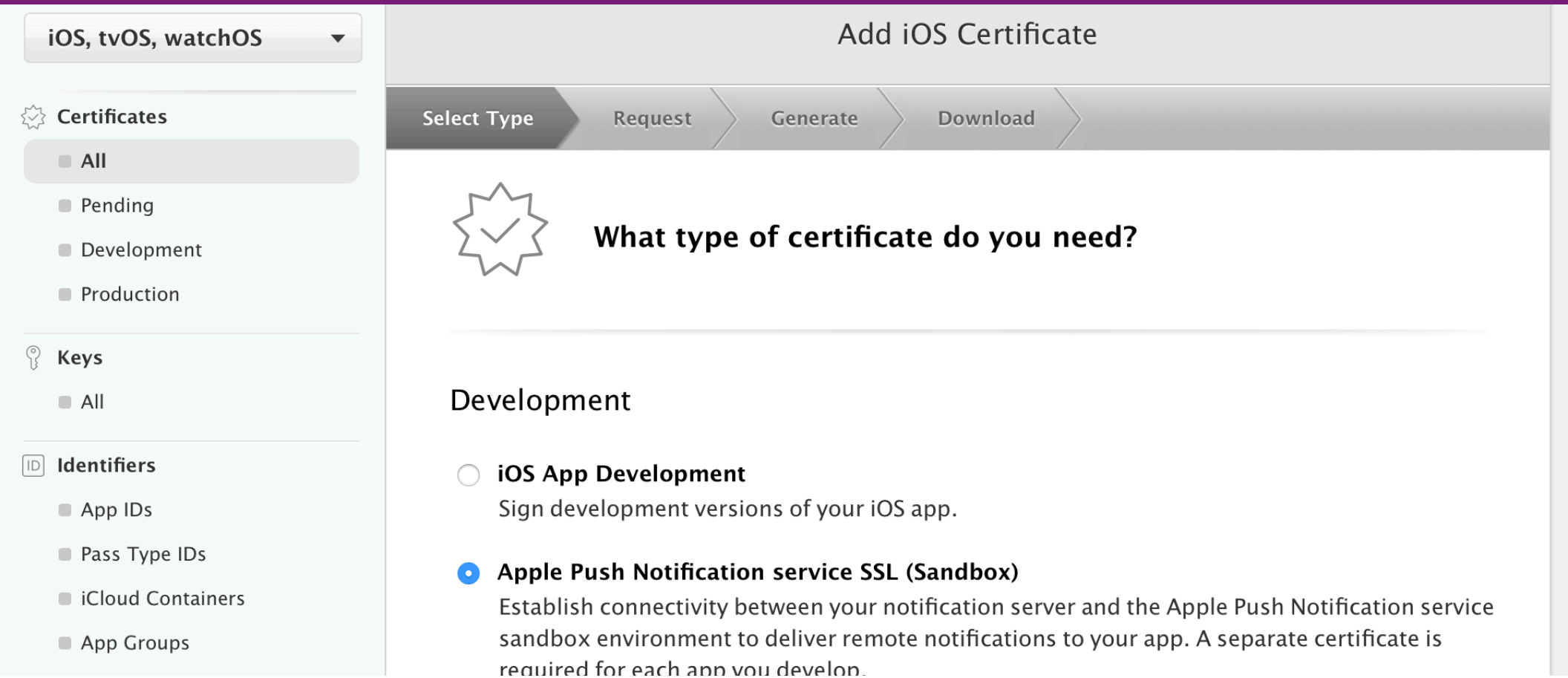
FIREBASE CLOUD MESSAGING



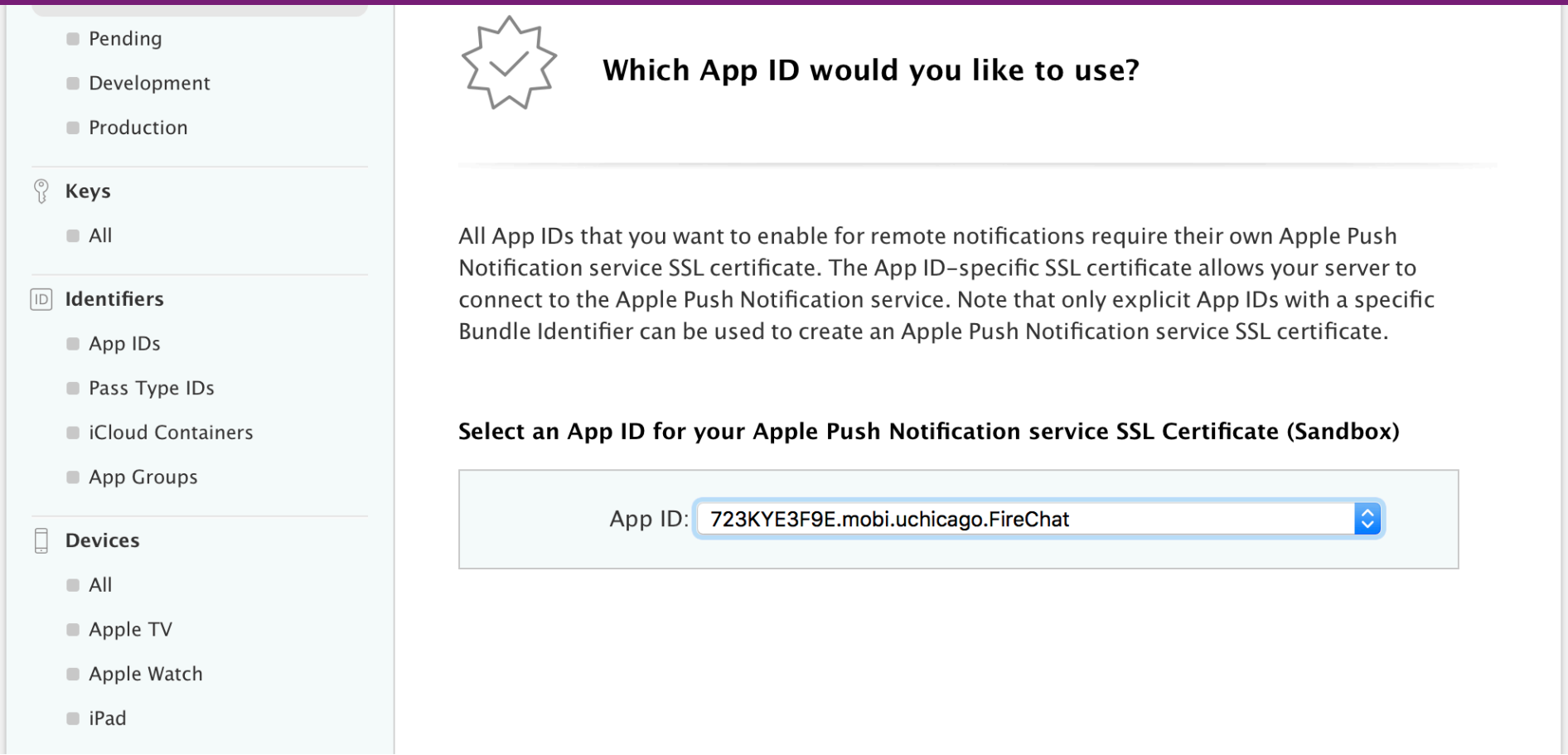
FIREBASE CLOUD MESSAGING



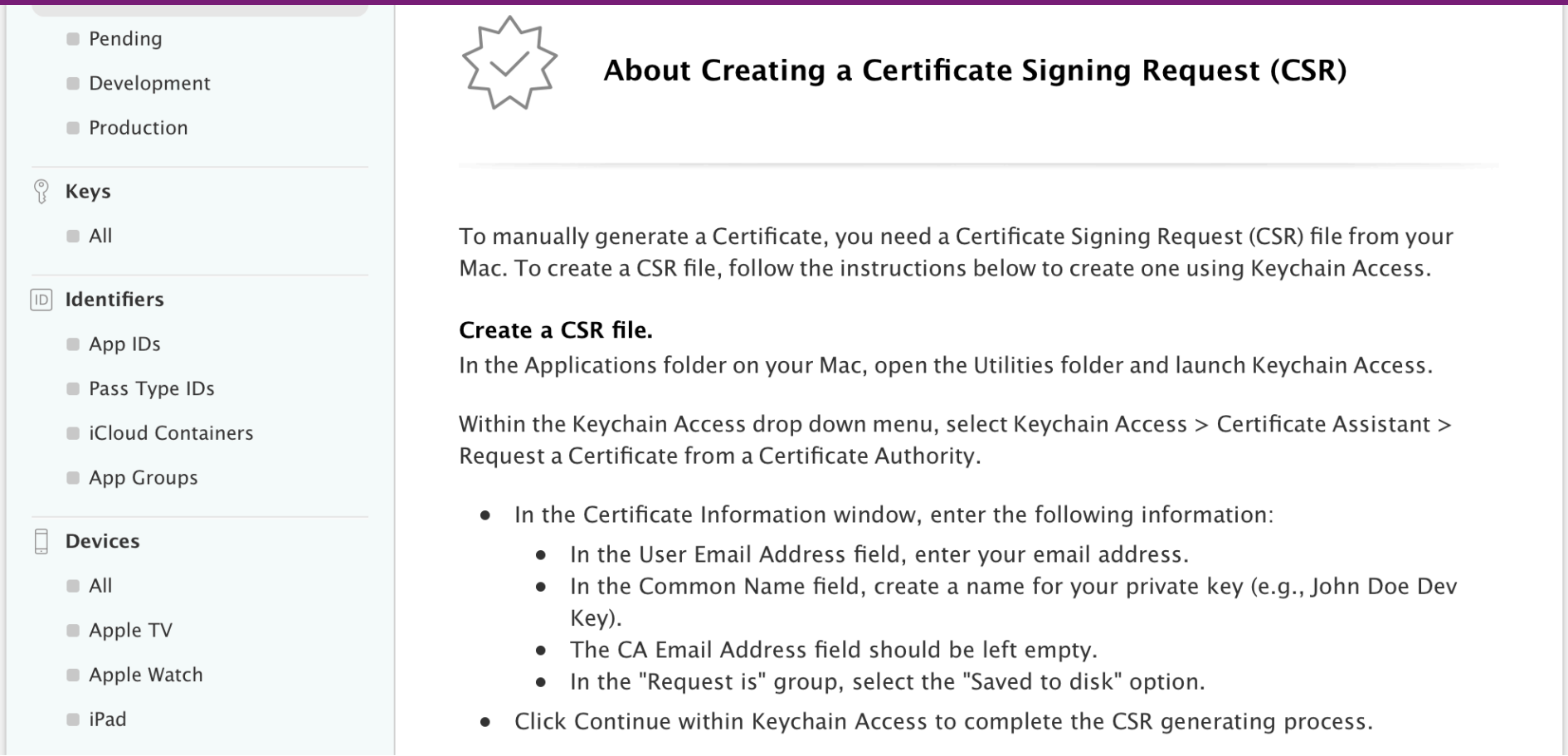
FIREBASE CLOUD MESSAGING



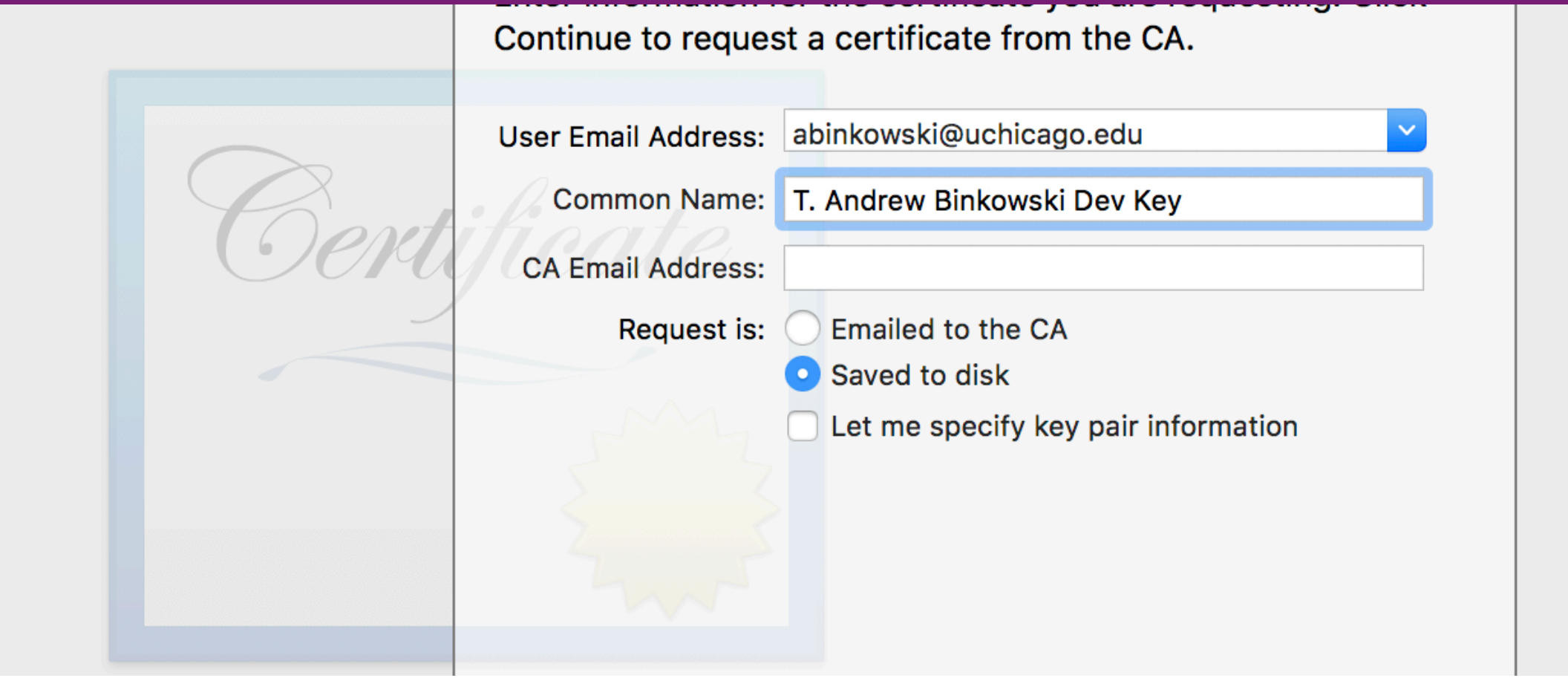
FIREBASE CLOUD MESSAGING



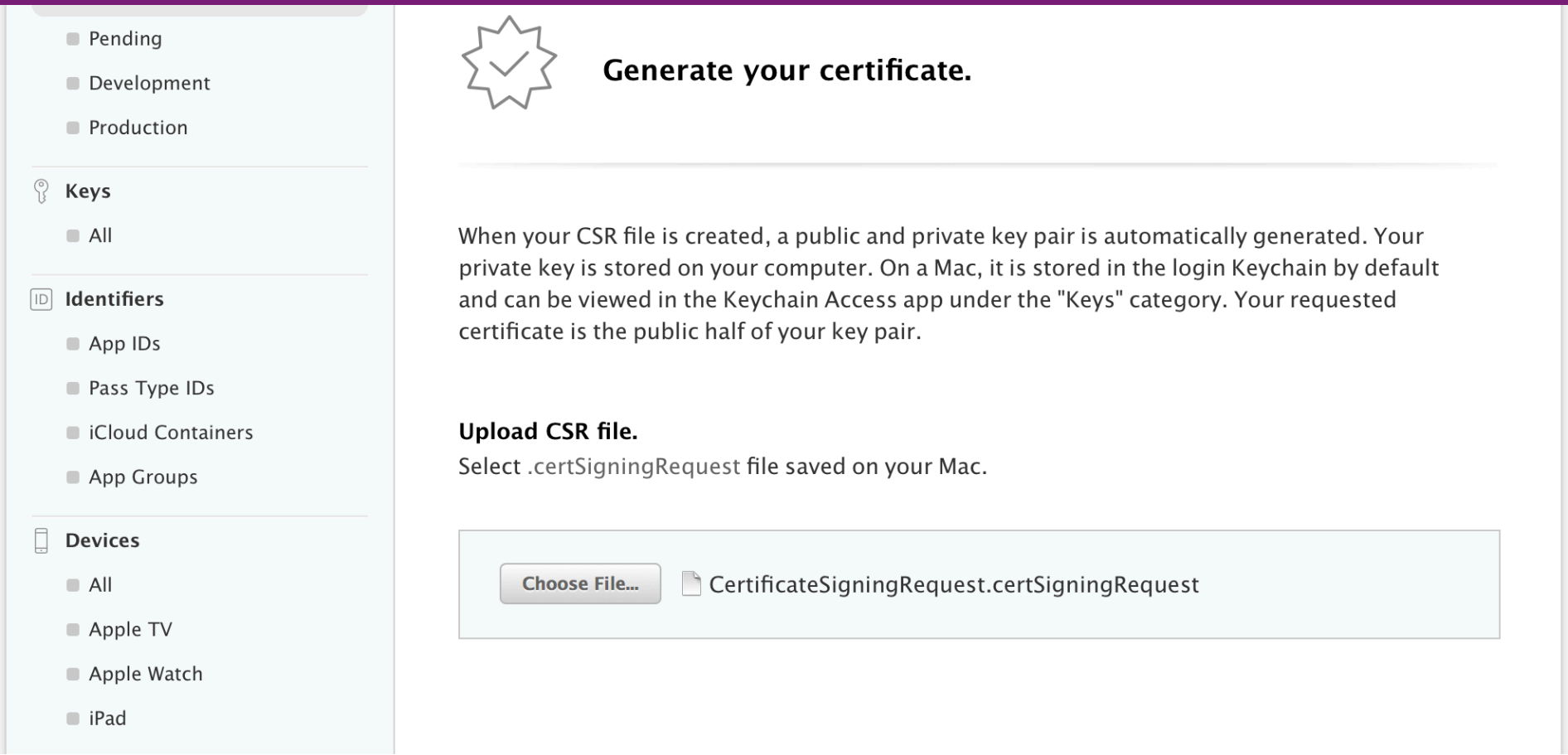
FIREBASE CLOUD MESSAGING



FIREBASE CLOUD MESSAGING

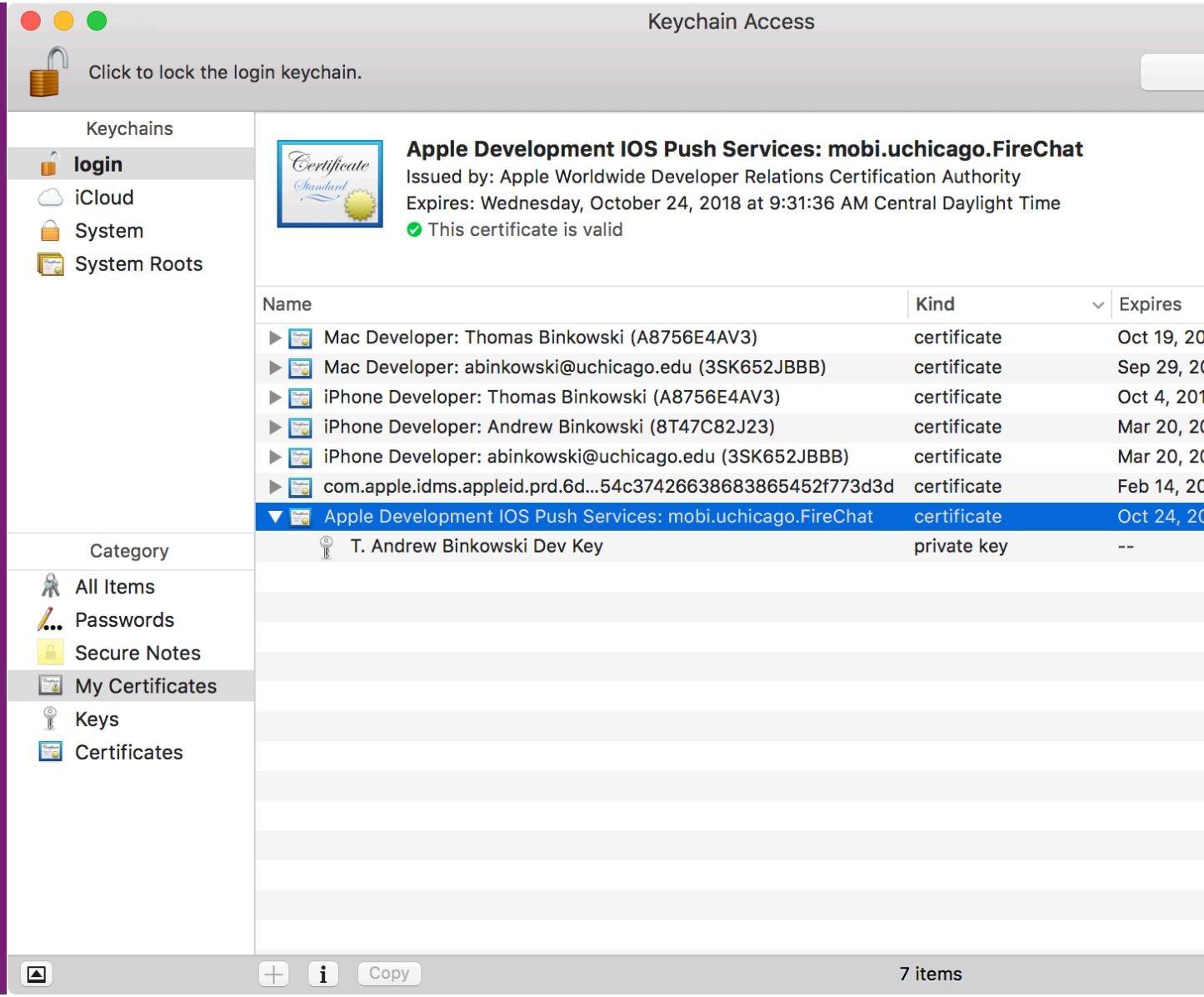


FIREBASE CLOUD MESSAGING



FIREBASE CLOUD MESSAGING

- Certificate will live in Keychain, you have to export it as .p12 files

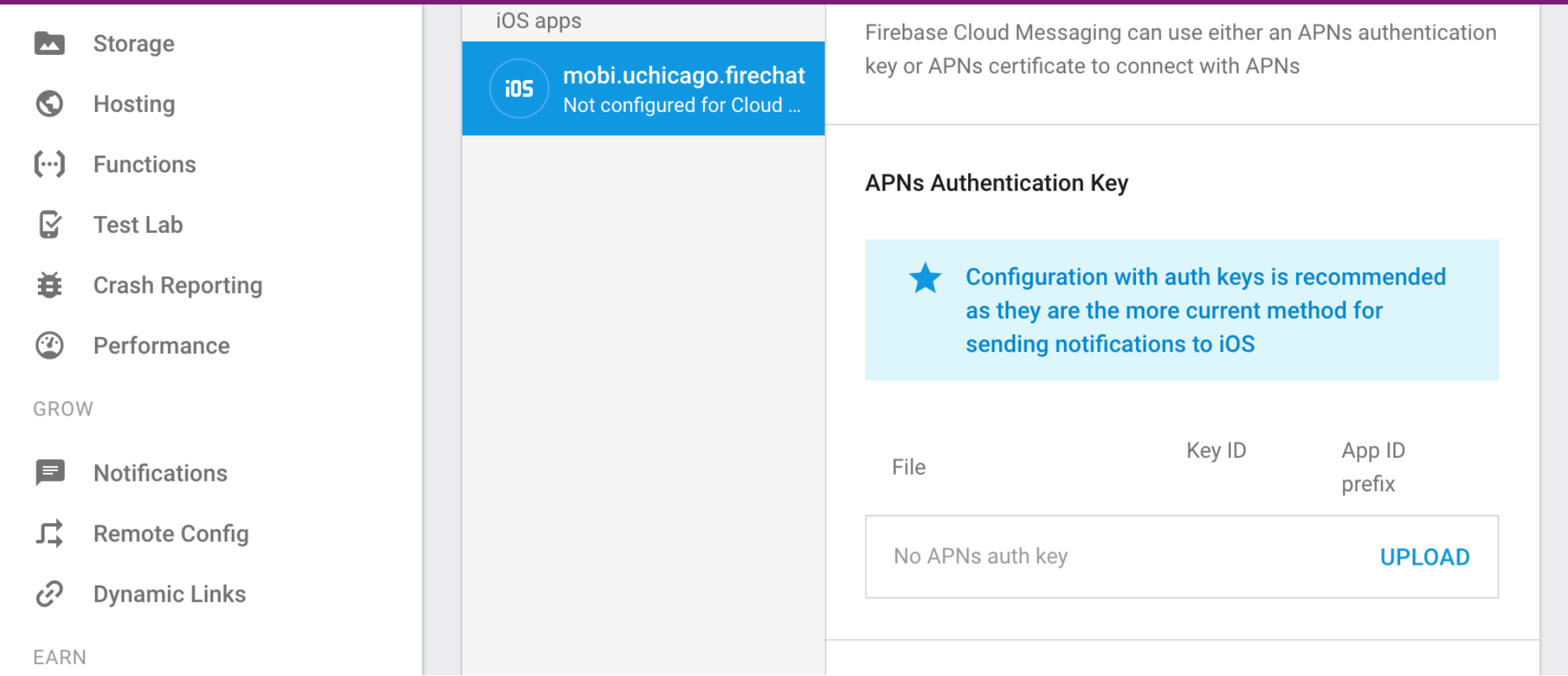


SET UP FIREBASE TO NOTIFICATIONS

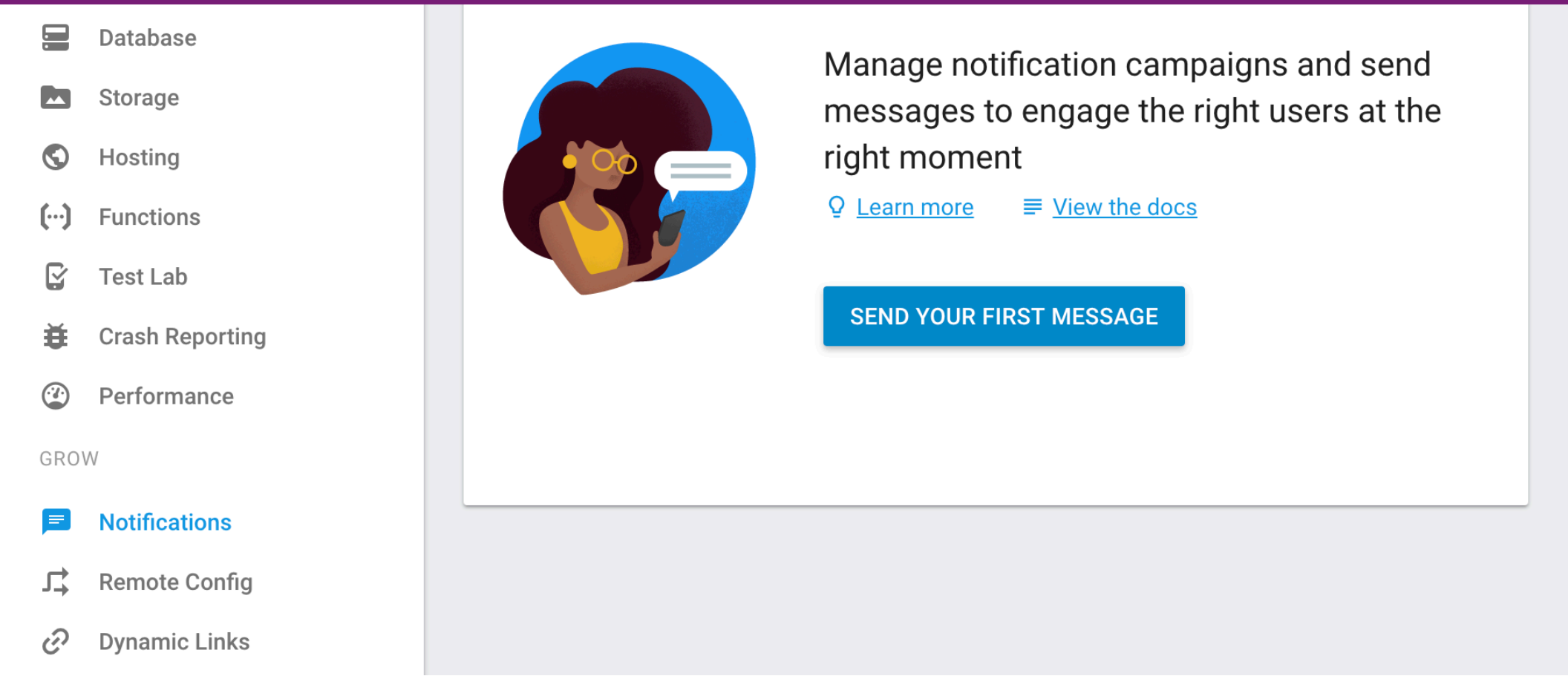
FIREBASE CLOUD MESSAGING

```
# Uncomment the next line to define a global platform for your project
# platform :ios, '9.0'

target 'FireChatAPNS' do
  # Comment the next line if you're not using Swift and don't want to use dynamic
      frameworks
  use_frameworks!

  # Pods for FireChatAPNS
pod 'Firebase/Core'
pod 'Firebase/Messaging'
end
```

FIREBASE CLOUD MESSAGING



FIREBASE CLOUD MESSAGING



HANDLE NOTIFICATIONS

HANDLE NOTIFICATIONS

- There is a lot of boilerplate code to handle all the different notification code paths
- Nothing to do with Firebase

HANDLE NOTIFICATIONS

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
    [UIApplicationLaunchOptionsKey: Any]?) -> Bool {

  // For iOS 10 display notification (sent via APNS)
  if #available(iOS 10.0, *) {
    UNUserNotificationCenter.current().delegate = self

    // Set up types of notifications preferences for users
    let authOptions: UNAuthorizationOptions = [.alert, .badge, .sound]
    UNUserNotificationCenter.current().requestAuthorization(
      options: authOptions,
      completionHandler: {_, _ in })
  } else {
    let settings: UIUserNotificationSettings = UIUserNotificationSettings(types: [.alert, .badge, .sound],
        categories: nil)
    application.registerUserNotificationSettings(settings)
  }

  // Register with APNS
  application.registerForRemoteNotifications()

  return true
}
```

REGISTER A DEVICE WITH APNS

HANDLE NOTIFICATIONS

- Notifications handled differently in different states

```
//
// MARK: - Handle the Notifications
//

/// If you are receiving a notification message while your app is in the background,
/// this callback will not be fired till the user taps on the notification launching
/// the application.
func application(_ application: UIApplication, didReceiveRemoteNotification userInfo: [AnyHashable: Any]) {

  print(userInfo)
}

/// If you are receiving a notification message while your app is in the background,
/// this callback will not be fired till the user taps on the notification launching the
/// application.
func application(_ application: UIApplication,
                 didReceiveRemoteNotification userInfo: [AnyHashable: Any],
                 fetchCompletionHandler completionHandler: @escaping (UIBackgroundFetchResult) -> Void) {
  print(userInfo)

  // This is required to let iOS know you are down handling the notification (not much time)
  completionHandler(UIBackgroundFetchResult.newData)
}

/// Callback from APNS that will be save in Firebase
func application(_ application: UIApplication,
                 didRegisterForRemoteNotificationsWithDeviceToken deviceToken: Data) {
  Messaging.messaging().apnsToken = deviceToken
}
```

HANDLE NOTIFICATIONS

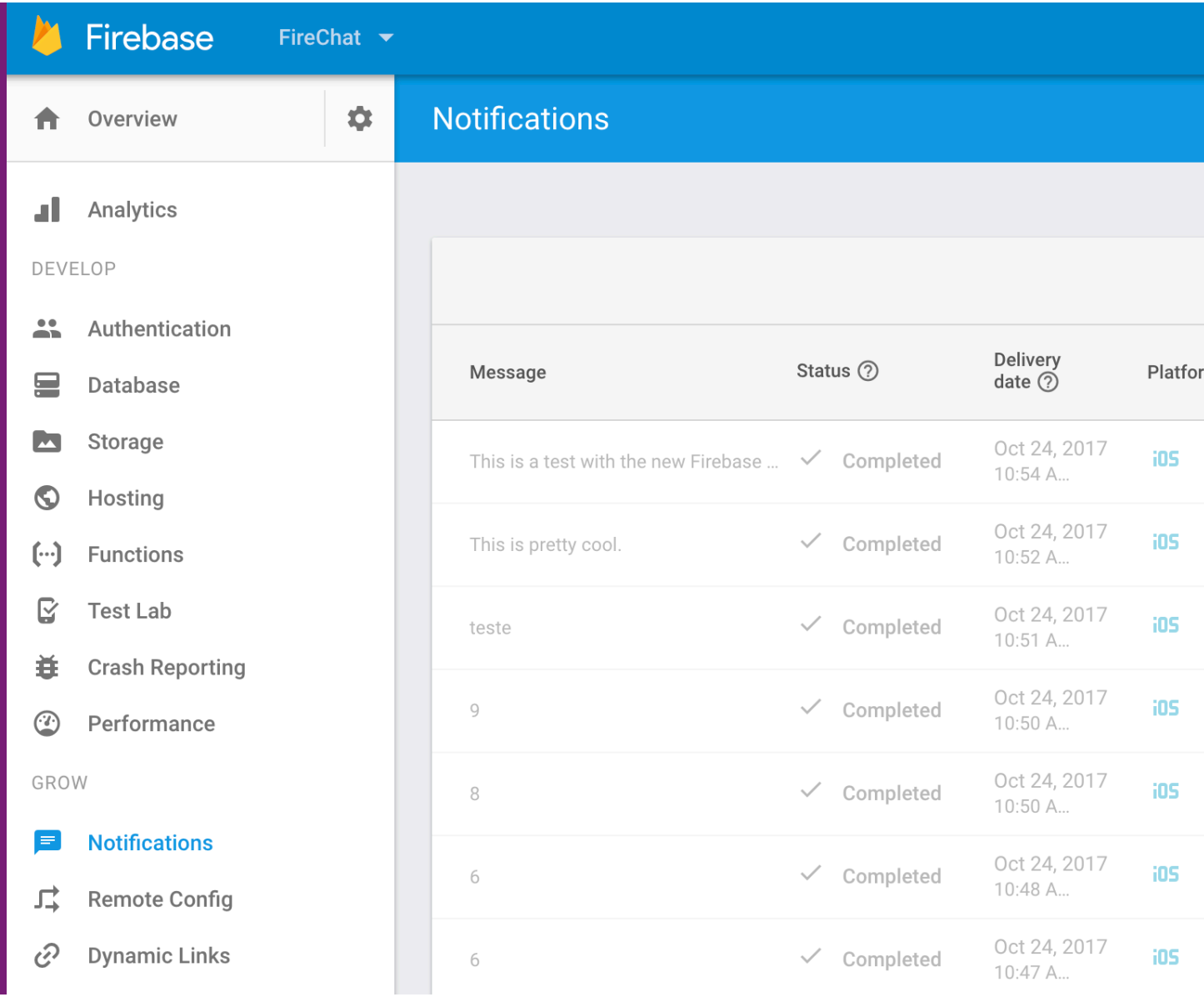
- Notifications handled differently in different states

```swift
//
// MARK: - User Notification Extensions
//


@available(iOS 10, *)
extension AppDelegate : UNUserNotificationCenterDelegate {

  /// Receive displayed notifications for iOS 10 devices if the device is in the
  /// foreground
  func userNotificationCenter(_ center: UNUserNotificationCenter,
                              willPresent notification: UNNotification,
                              withCompletionHandler completionHandler: @escaping (UNNotificationPresentationOptions) -> Void) {
    let userInfo = notification.request.content.userInfo

    // Print full message.
    print(userInfo)

    // Change this to your preferred presentation option
    completionHandler([.alert])
  }

  ///
  func userNotificationCenter(_ center: UNUserNotificationCenter,
                              didReceive response: UNNotificationResponse,
                              withCompletionHandler completionHandler: @escaping () -> Void) {
    let userInfo = response.notification.request.content.userInfo
    // Print full message.
    print(userInfo)

    completionHandler()
  }
}
```

SEND FIREBASE NOTIFICATIONS

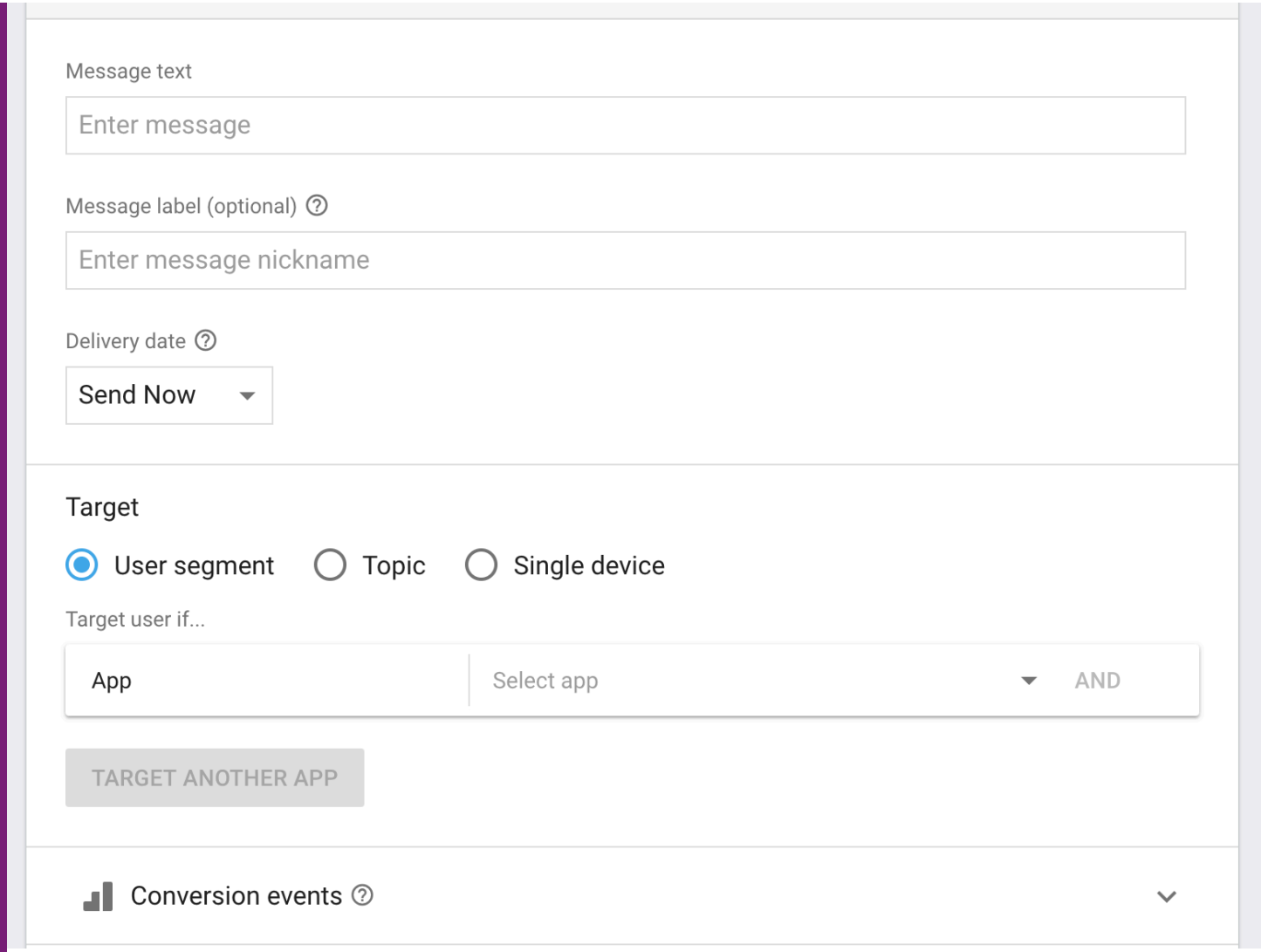
SEND FIREBASE NOTIFICATIONS

- Send from console
- Different options for sending



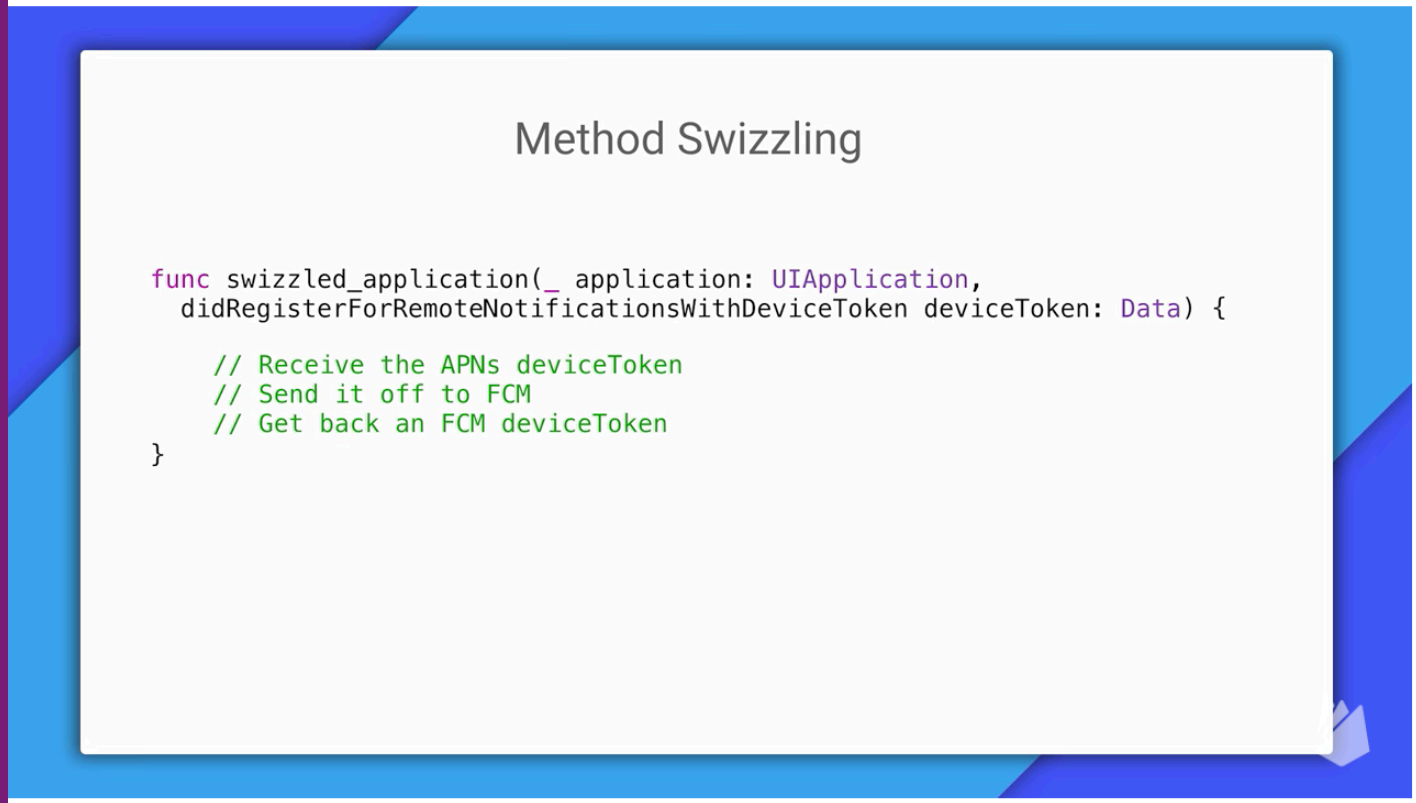
SEND FIREBASE NOTIFICATIONS

- If you want to target specific users, you need to manage that as part of your database



FIREBASE CLOUD MESSAGING

- https://www.youtube.com/watch?v=A1SDBIViRtE&t=6s
- How does it work behind the scenes



Understanding Firebase Cloud Messaging on iOS - Firecasts

13,141 views 122 4 SHARE

Firebase
Published on Jan 9, 2017

SUBSCRIBED 89K

THE UNIVERSITY OF CHICAGO

MPCS 51033 • AUTUMN 2019 • SESSION 5

BACKENDS FOR MOBILE APPLICATIONS