



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)

Think about this

Think about this

CLASS NEWS

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

The screenshot shows the Slack interface. On the left, a sidebar lists channels: All Threads, Channels (# assignment1, # general, # random), Direct Messages (slackbot, Andrew (you), acham1, Xuefeng, Yi Wang), and Apps. The # general channel is selected and highlighted in green. The main area shows a conversation in the # general channel. Andrew posted a message at 10:37 AM about adding a Twitter integration. Andrew then responded at 9:09 PM, stating he's not going to have formal office hours tomorrow but will let know if there are any questions. He plans to have regular office hours Thursday around 10 starting next week. Below this, a message from Andrew at 10:58 AM links to a Google Cloud Platform blog post about the new code editor in Google Cloud Shell. A reply from Yi Wang at 5 hours ago says "OK. Thank you!" with a red circle highlighting the reply icon in the message bar.

#general

Andrew 10:37 AM added an integration to this channel: twitter

Andrew 9:09 PM Hi. I'm not going to have any formal office hours tomorrow but let me know if you have any questions. I plan on having regular office hours Thursday around 10 starting next week.

Thursday, September 28th

Andrew 10:58 AM <https://cloudplatform.googleblog.com/2016/10/introducing-Google-Cloud-Shells-new-code-editor.html?m=1>

Google Cloud Platform Blog [Introducing Google Cloud Shell's new code editor](#)
Posted by Sachin Kotwani, Product Manager We've heard from a lot of Google Cloud Platform (GCP) users that they like to edit code and c... (9kB)

Here's an example of how you can use the Cloud Shell code editor to create a sample app, push your changes to Google Cloud Source Repository, deploy the app to Google App Engine Standard, and use Stackdriver Debugger.

Andrew 11:15 AM

Message #general @

Thread
Yi Wang and you

Yi Wang Today at 10:56 AM Direct message
I received the coupon email in my uchicago email box, but when I clicked the url to redeem, it redirected to a page that is logged in with my gmail. I didn't notice that and finished the whole process.

2 replies

Andrew 6 hours ago If it worked with your gmail, then we can go ahead and try it. I'll take a look in class before we actually start using it to make sure it looks ok. I will send an email to my contact at google just to double check.

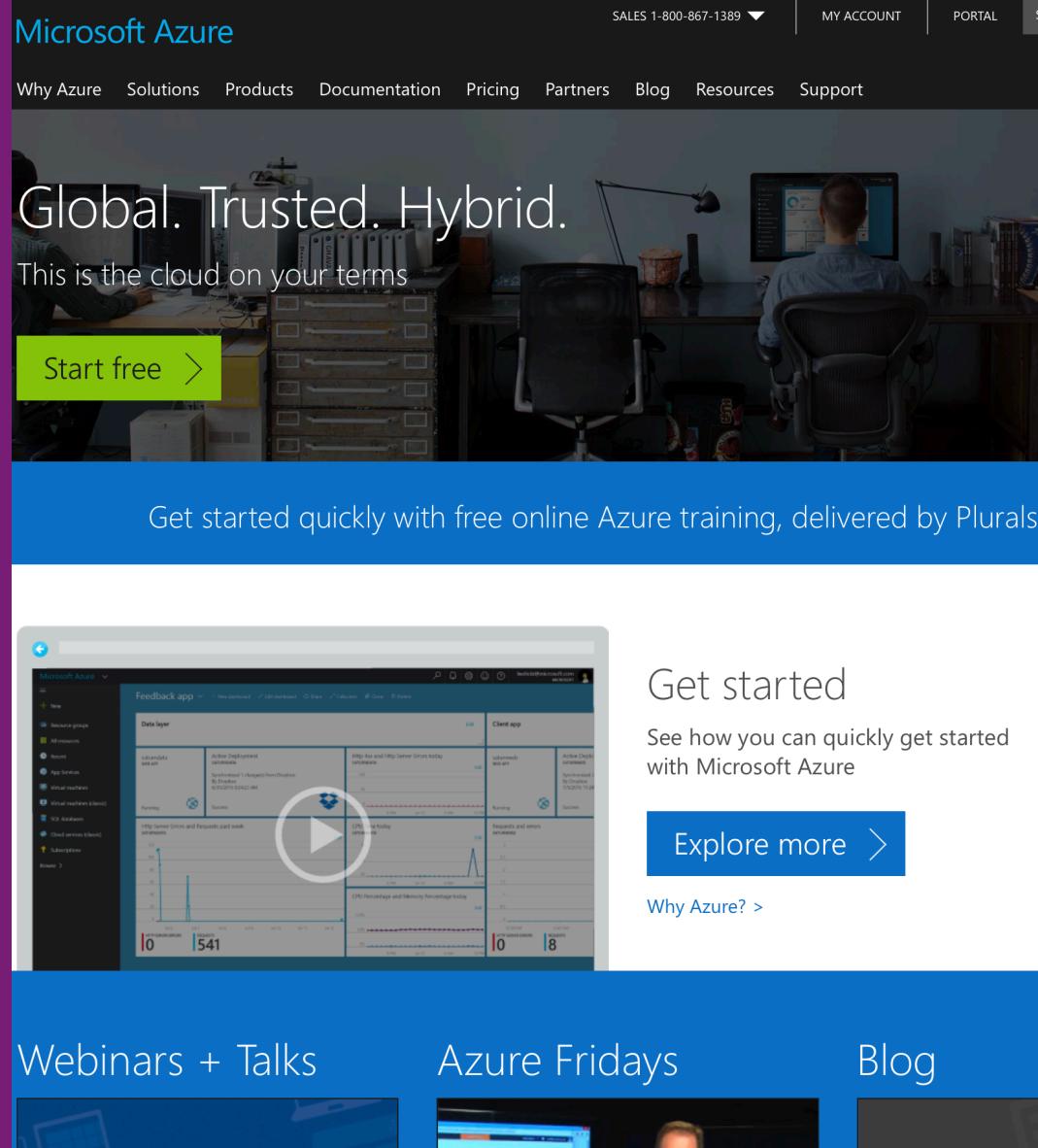
Yi Wang 5 hours ago OK. Thank you!

Reply...

CASE STUDIES

CASE STUDIES

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



The screenshot shows the Microsoft Azure homepage. At the top, there's a dark header with the "Microsoft Azure" logo, a "SALES 1-800-867-1389" phone number, "MY ACCOUNT", and a "PORTAL" link. Below the header is a navigation bar with links for "Why Azure", "Solutions", "Products", "Documentation", "Pricing", "Partners", "Blog", "Resources", and "Support". The main content area features a large banner with the text "Global. Trusted. Hybrid." and "This is the cloud on your terms". A green button labeled "Start free >" is visible. Below the banner, a video player shows a person working at a desk with multiple monitors, with the text "Get started quickly with free online Azure training, delivered by Pluralsight". To the left, a sidebar shows a list of services: "New", "Resource groups", "All resources", "Compute", "App Service", "Virtual machines", "Virtual machines (classic)", "SQL databases", "Cloud services (classic)", and "Storage". On the right, there's a "Feedback app" dashboard showing metrics like CPU usage, memory, and network traffic. At the bottom, there are sections for "Webinars + Talks", "Azure Fridays", and "Blog".

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

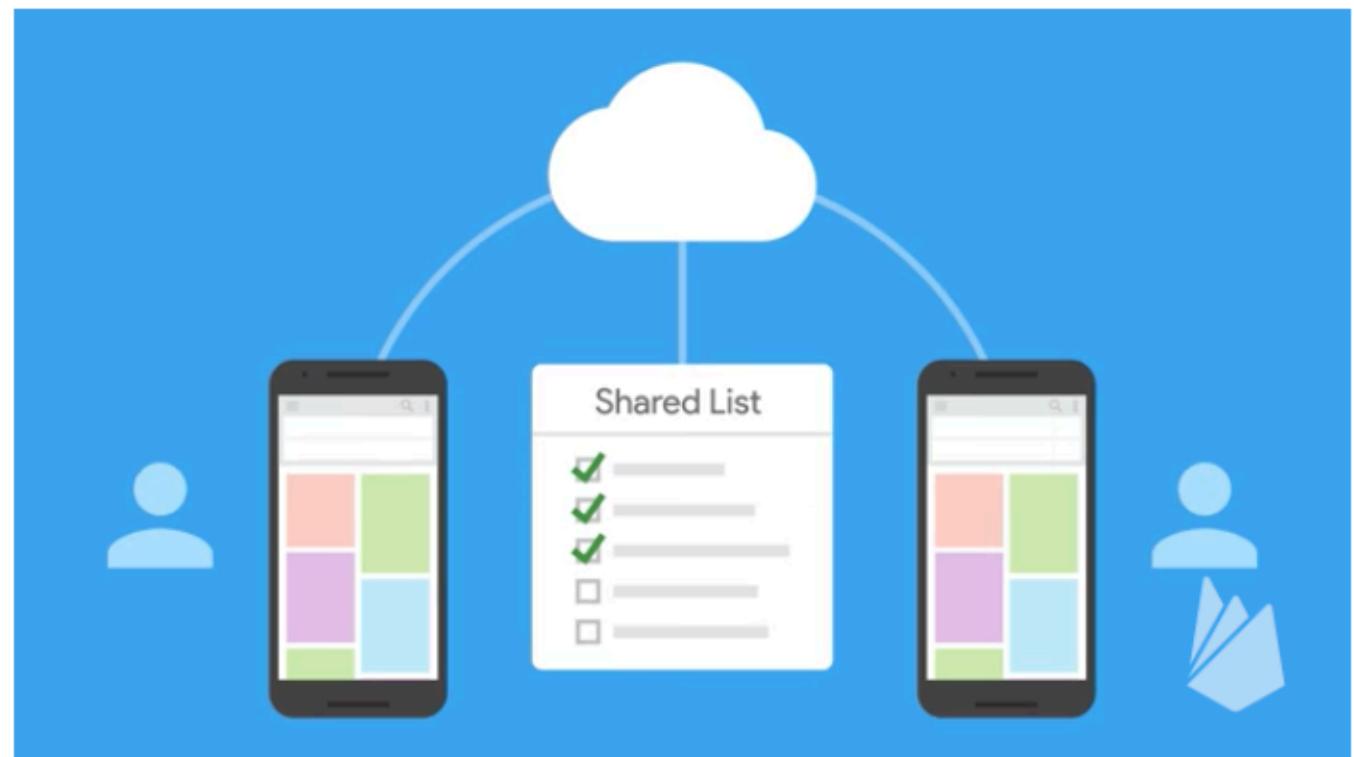
At the beginning of this year, we created a group inside our video engineering team to implement a new solution for the ingesting, encoding, publishing and the syndication of our growing library of video content. The main goal of the team was to implement a job processing pipeline that was vendor agnostic 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=PLI-K7zZEesYLmOF_07layrTntevxtbUxDL



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'

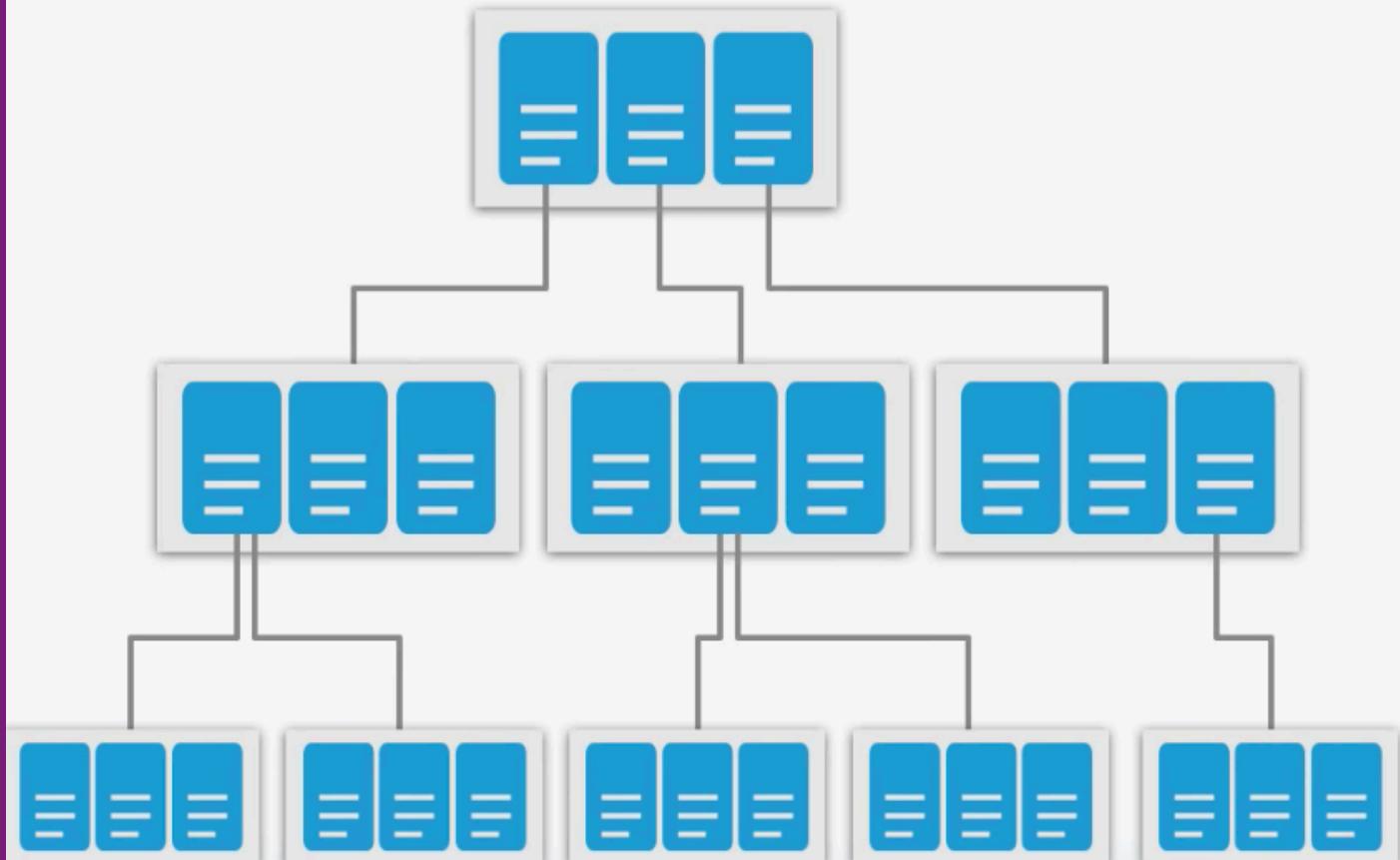
  pod 'Firebase/RemoteConfig'
  pod 'Firebase/Firestore'

  pod 'SDWebImage'

end
```

FIRESTORE

- Data is stored as documents and collections



FIRESTORE

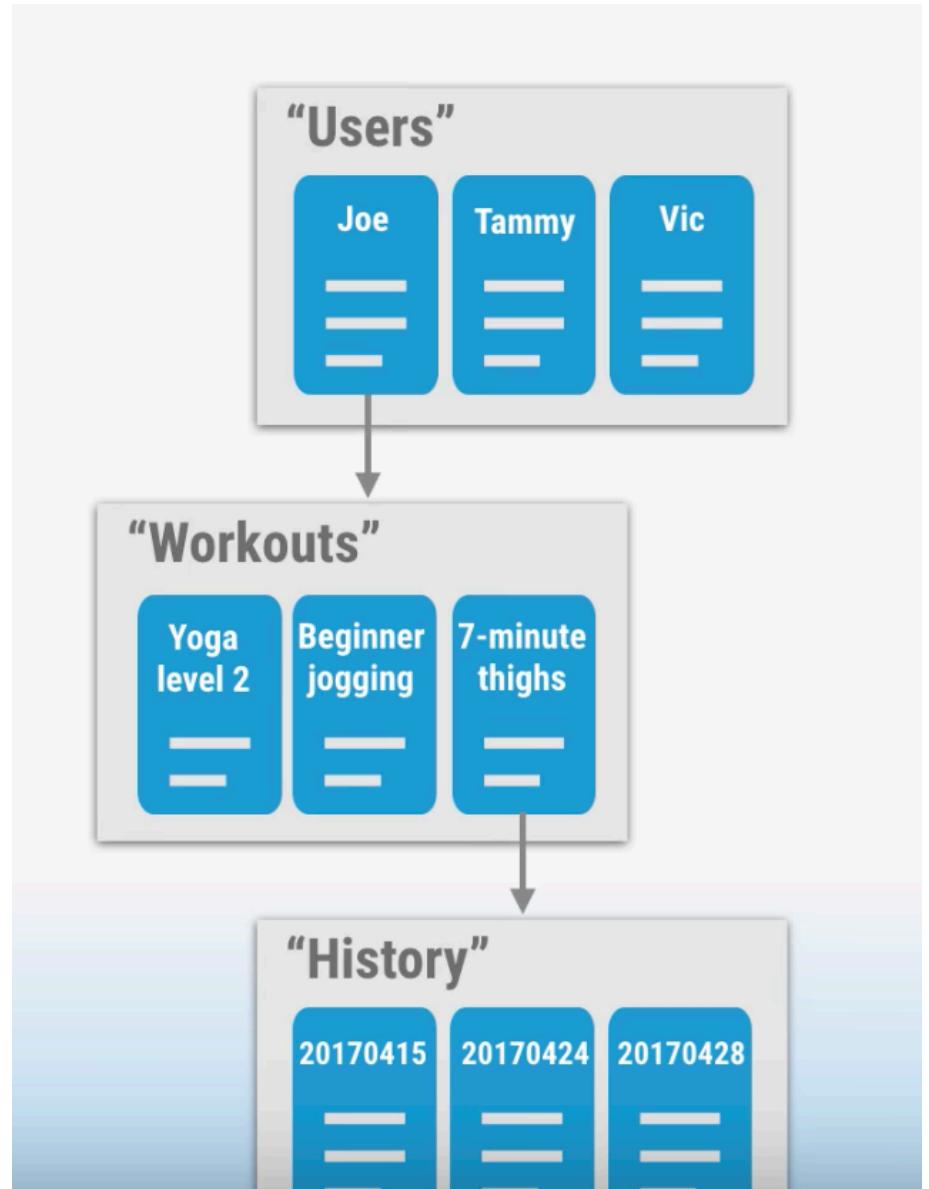
- Document is just key-value pairs

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 your application has to be a collection



USING FIRESTORE

USING FIRESTORE

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

The screenshot shows the Firebase console interface. On the left, there's a sidebar with sections like Project Overview, Develop, Quality, and Analytics. The main area is titled 'Database' and shows tabs for Data, Rules, Indexes, and Usage. A modal window titled 'Create database' is open, divided into two steps: 1. Secure rules for Cloud Firestore (selected) and 2. Set Cloud Firestore location. Step 1 contains instructions and two radio button options: 'Start in locked mode' (selected) and 'Start in test mode'. It also includes a link to learn more about security rules and a code snippet for the 'locked mode' rules. Step 2 is partially visible at the bottom. At the bottom of the modal, there's a note about preventing Cloud Datastore usage, a 'Cancel' button, and a 'Next' button.

Create database

1 Secure rules for Cloud Firestore 2 Set Cloud Firestore location

After you define your data structure, you will need to write rules to secure your data.
[Learn more](#)

Start in **locked mode**
Make your database private by denying all reads and writes

Start in **test mode**
Get set up quickly by allowing all reads and writes to your database

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if false;
    }
  }
}
```

i All third party reads and writes will be denied

Enabling Cloud Firestore will prevent you from using Cloud Datastore with this project, notably from the associated App Engine app

Cancel Next

USING FIRESTORE

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

The screenshot shows the Firebase Project Overview on the left and the Database section on the right.

Project Overview (Left):

- Develop
 - Authentication
 - Database
 - Storage
 - Hosting
 - Functions
 - ML Kit
- Quality
 - Crashlytics
 - Performance
 - Test Lab
 - App Distribution
- Analytics
 - Dashboard
 - Events
 - Conversions
 - Audiences
 - Funnels
 - User Properties
 - Latest Release
 - Extensions

Database (Right):

- chat-cat
- Cloud Firestore
- Data (selected)
- Rules
- Indexes
- Usage

Under the Data tab:

- chat-cat-2f244
- + Start collection

Bottom right: Your database is ready to go. Just add data.

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

The screenshot shows the Firestore console interface. At the top, there's a navigation bar with a home icon, followed by a breadcrumb trail: 'trending > hot'. Below this is a table structure representing a document:

firechat-66f87	trending	hot
+ ADD COLLECTION	+ ADD DOCUMENT	+ ADD COLLECTION
trending >	hot >	+ ADD FIELD group: "art" timestamp: "today" votes: 200

The 'hot' collection is currently selected, indicated by a grey background. The 'ADD FIELD' button is highlighted in blue. The document contains three fields: 'group' with value 'art', 'timestamp' with value 'today', and 'votes' with value 200.

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 ?? ""
    }
}
```

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

REMOVE IT

USING FIRESTORE

App Engine		Quotas	VIEW USAGE HISTORY			
 Dashboard	<p>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? ↗</p> <p>Storage</p>	Resource	Usage today	Daily quota 	Per-minute quota 	Rate limit status 
 Services		Cloud Storage Class B Operations	0 of 0.05	<div style="width: 0%;"><div style="width: 0%;"></div></div> 0%	--	Standard rate
 Versions		Cloud Storage Class A Operations	0 of 0.02	<div style="width: 0%;"><div style="width: 0%;"></div></div> 0%	--	Standard rate
 Instances		Cloud Storage Standard Storage	0.00098 of 5 GB	<div style="width: 0%;"><div style="width: 0%;"></div></div> 0%	--	Standard rate
 Task queues		Cloud Storage Network (Egress) - Americas and EMEA	0.000001 of 1 GB	<div style="width: 0%;"><div style="width: 0%;"></div></div> 0%	--	Standard rate
 Security scans		Firebase Read Operations	0.000023 of 0.05 Million Ops	<div style="width: 0%;"><div style="width: 0%;"></div></div> 0%	--	Standard rate
 Firewall rules		Firebase Small Operations	0.000003 of 0.05 Million Ops	<div style="width: 0%;"><div style="width: 0%;"></div></div> 0%	--	Standard rate
 Quotas		Firebase API Calls	81	--	--	Standard rate
 Blobstore		Data Sent to Firebase API	0.000001 GB	--	--	Standard rate
 Memcache		Firebase 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

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](#).

Cloud Firestore

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
<p>Deep queries with limited sorting and filtering functionality.</p> <ul style="list-style-type: none">• You can only sort or filter on a property, not sort <i>and</i> filter on a property, in a single query.• Queries are deep by default: They always return the entire subtree.	<p>Indexed queries with compound sorting and filtering.</p> <ul style="list-style-type: none">• 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 scalability
 - 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



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



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



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 your 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 your 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

- Label images using Google Vision API
 - Full
 - Subset

Image Labeling

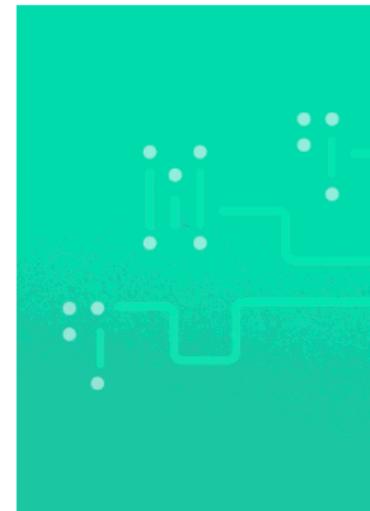
iOS 

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)

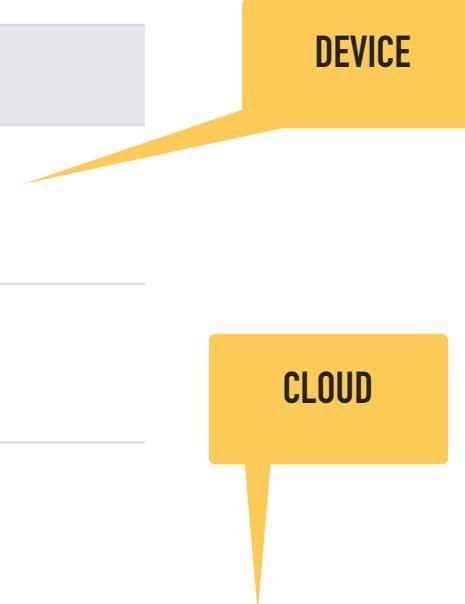
	On-device	Cloud
Pricing	Free	Free for first 1000 uses of this feature per month: see Pricing
Label coverage	400+ labels that cover the most commonly-found concepts in photos. See below.	10,000+ labels in many categories. See below. Also, try the Cloud Vision API demo to see what labels can be found for 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		
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/0bmqjrz
Confidence	0.9205354	Confidence	0.9860726
Description		Description	
Sports		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 > Documentation



SEND FEEDBACK

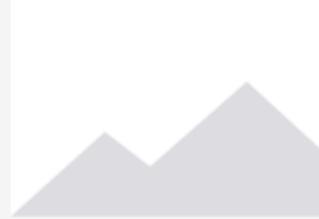
Try it!

Use the application below to return image annotations for your image file. Click the **Show JSON** button to view the raw response.

- Maximum file size is 4MB.
- Your browser must have JavaScript enabled.

Try the API

Upload your image



[HTTPS://CLOUD.GOOGLE.COM/VISION/DOCS/DRAG-AND-DROP?AUTHUSER=0](https://cloud.google.com/vision/docs/drag-and-drop?authuser=0)

- Cloud vision api demo site

ML KIT

IMAGE LABELS

```
pod 'Firebase/MLVision'
```

- Add the pod

```
# 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  
let o  
option  
let l  
  
label  
gu  
  
// 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

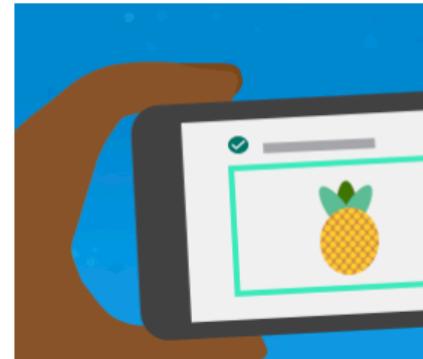
AutoML Vision Edge

iOS 

Create custom image classification models from your own training data with AutoML Vision Edge.

- Train your own models with vision edge

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 train your 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).jpg
| ___well-tempered (2).jpg
```

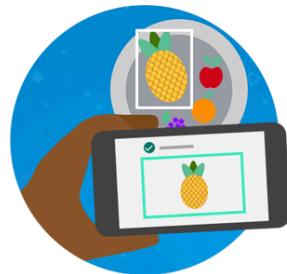
ML KIT

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

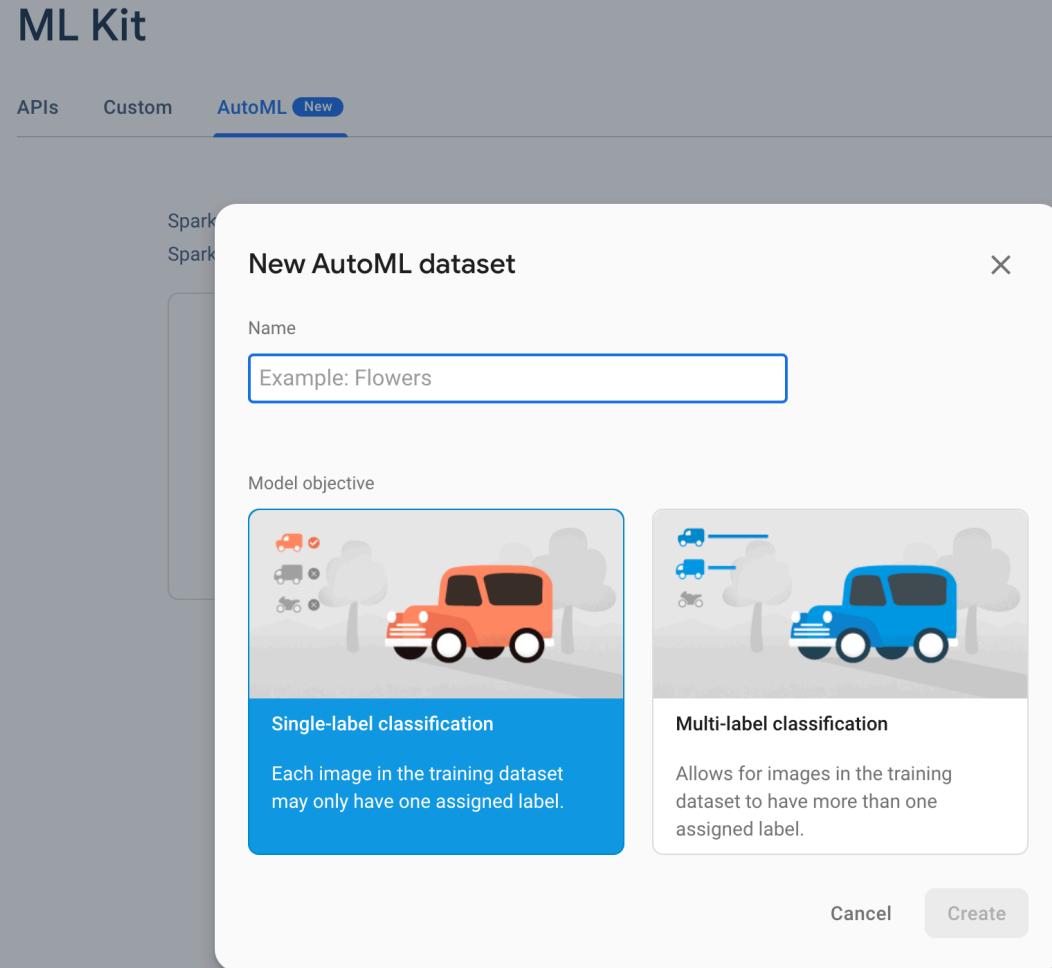
- Processes them to train a model
- 1 free model
 - 1000 images
 - 1 hours of processing

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

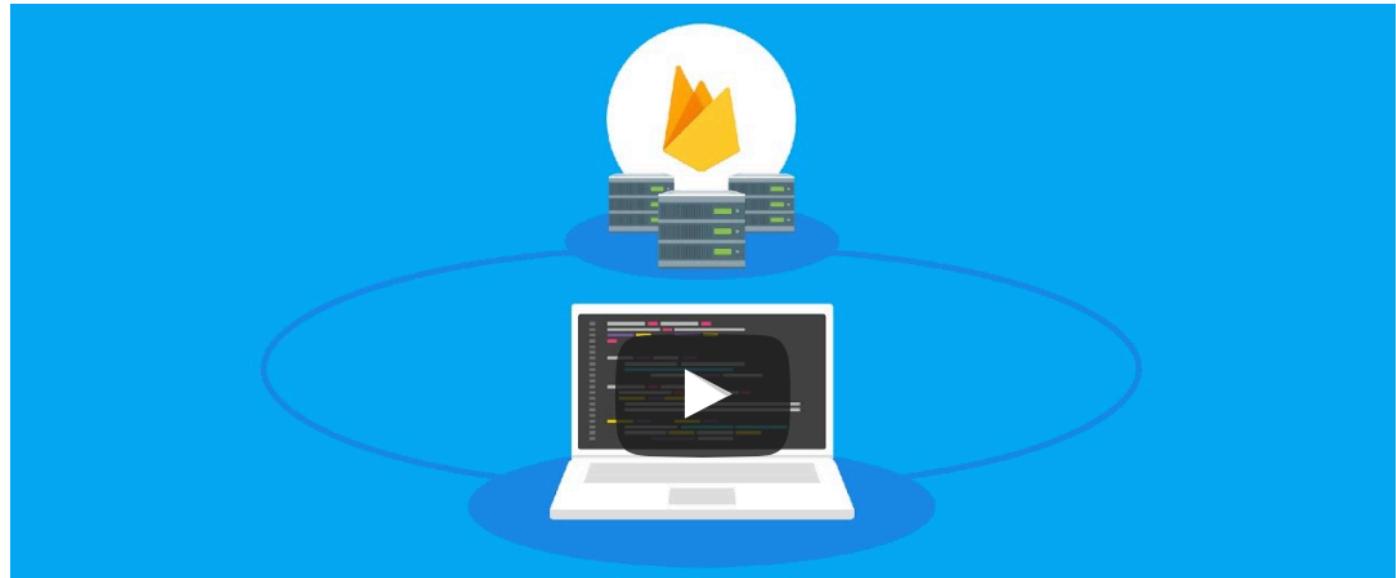
- Enable your model in the console
- Real-time update



FIREBASE CLOUD FUNCTIONS

🔥 FIREBASE CLOUD FUNCTIONS

- <https://youtu.be/vr0Gfvp5v1A>



Cloud Functions
for Firebase

[HTTPS://YOUTU.BE/VR0GFVP5V1A](https://youtu.be/vr0Gfvp5v1A)



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

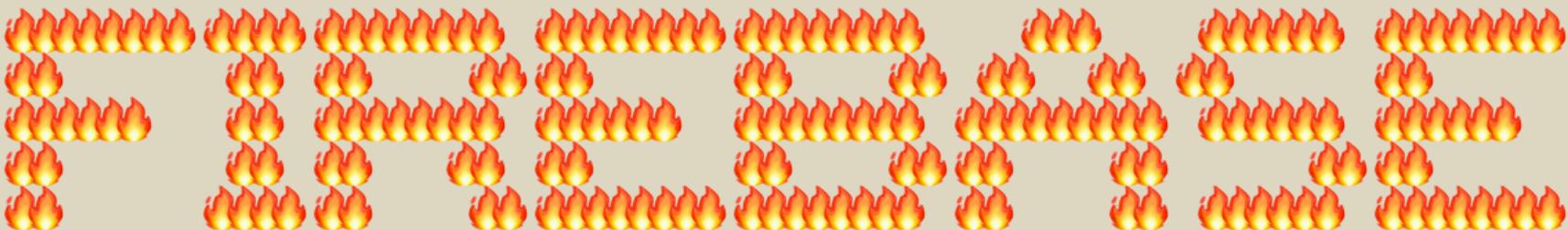
FIREBASE CLOUD FUNCTIONS

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

The screenshot shows the Firebase Cloud Functions setup process. The left sidebar lists various services: Authentication, Database, Storage, Hosting, Functions (which is selected), and ML Kit. The main content area has a title "Functions" with the subtitle "Run your mobile backend code without managing servers". A blue callout box titled "Set up Functions" contains two steps: "1 Install" and "2 Deploy". It instructs users to install Firebase command line tools using npm ([Node.js](#)) and provides the command `$ npm install -g firebase-tools`. It also mentions that users can change npm permissions if it doesn't work. Below this, a note says to run the install command again if they've previously installed tools. At the bottom right of the callout are "Cancel" and "Continue" buttons. To the right of the callout, there are two sections: "How does Functions work?" with a link to "View the docs" and "What can Functions do for me?" with a link to "Learn more". The footer features the Firebase logo and the text "Introducing Cloud Functions for Firebase".

🔥 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 deploy 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`.

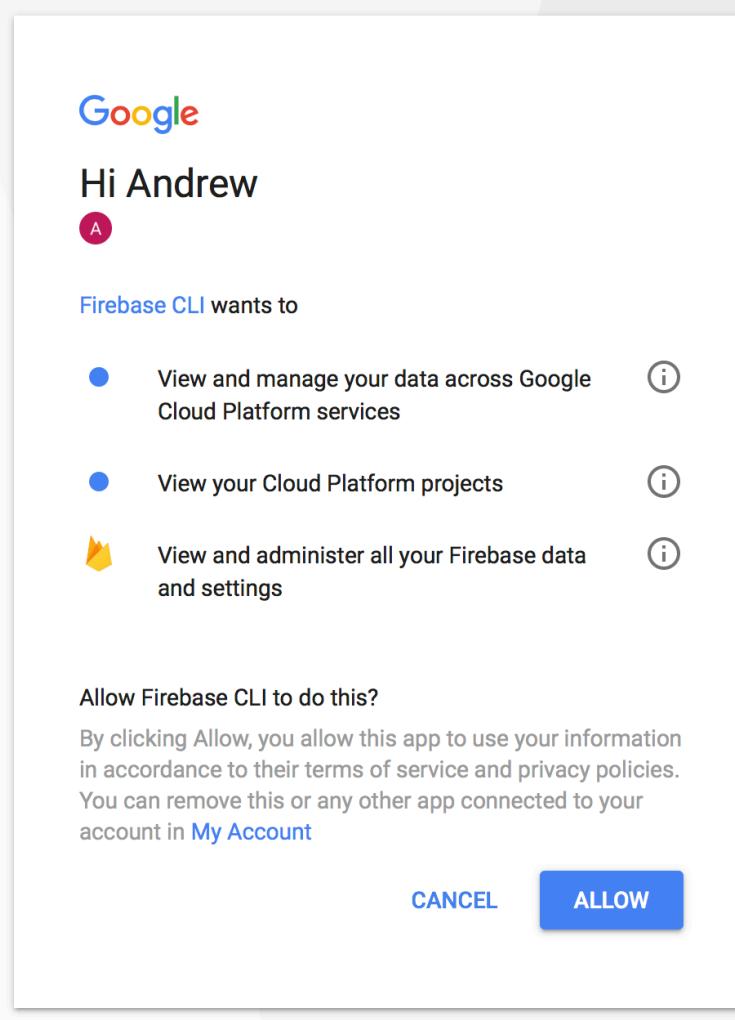
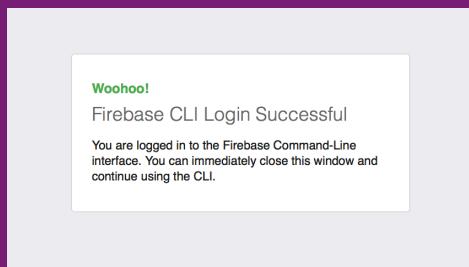
✓ wrote `functions/package.json`

FIREBASE CLOUD FUNCTIONS

```
firebase — node /usr/local/bin/firebase login — 83x24
tabinkowski@Ts-MacBook-Pro ~/Google Drive/g-Teaching/uchicago.cloud/mpcs51033-2017-
autumn/mpcs51033-2017-aut
[515 % firebase login
? Allow Firebase to collect information? No
[ FIREBASE LOGIN
Visit this URL on any dev
https://accounts.google.com/o/oauth2/auth?client\_id=50550455887-1grhgmd47bqnekij5i8b5pr03ho849e6.apps.googleusercontent.com&scope=email%20openid%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloudplatformprojects.readonly%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Ffirebase%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloud-platform&response\_type=code&state=1041200543&redirect\_uri=http%3A%2F%2Flocalhost%3A9005
Waiting for authentication...
```

🔥 FIREBASE CLOUD FUNCTIONS

- Give Firebase permissions



English (United States) ▾

Help

Privacy

Terms



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 not selected any features. Continuing will simply associate this folder 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

```
functions — more package.json — 83×24
tabinkowski@Ts-MacBook-Pro ~/Google Drive/g-Teaching/uchicago.cloud/mpcs51033-2017-
autumn/mpcs51033-2017-autumn-code-samples.firebaseio/functions
[548 % ls
index.js      node_modules  package.json
tabirkowski@Ts-MacBook-Pro ~/Google Drive/g-Teaching/uchicago.cloud/mpcs51033-2017-
autu /mpcs51033-2017-autumn-code-samples.firebaseio/functions
[549 % cat package.json
{
  "CODE": [
    "functions"
  ],
  "description": "Cloud Functions for Firebase",
  "dependencies": {
    "firebase-admin": "^5.4.0",
    "firebase-functions": "^0.7.0"
  },
  "MODULES": [
    "package.json"
  ]
}
package.json (END)
```

The diagram illustrates the structure of a package.json file for Firebase Cloud Functions. It highlights specific sections of the JSON object:

- CODE**: Points to the "functions" key.
- MODULES**: Encloses the "node_modules" and "package.json" entries.
- INFO**: Encloses the "description" and "dependencies" keys.

🔥 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.firebaseio/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.firebaseio/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.firebaseio/functions
[553 % atom index.js
tabinkowski@Ts-MacBook-Pr
autumn/mpcs51033-2017-aut
[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.firebaseio.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.firebaseio/functions
```

FIREBASE CLOUD FUNCTIONS

The screenshot shows a web browser window displaying the Firebase Cloud Functions dashboard. The URL in the address bar is `console.firebaseio.google.com/u/0/project/chat-cat-2f244/functions/list`. The dashboard has a dark blue header with the word "chat-cat" in a dropdown menu. Below the header, there are tabs for "Dashboard", "Health", "Logs", and "Usage". The "Dashboard" tab is selected. A table lists a single function:

Function	Trigger	Region	Runtime	Memory	Timeout
helloWorld	HTTP Request https://us-central1-chat-cat-2f244.cloudfunctions.net/helloW...	us-central1	Node.js	256 MB	60s

The left sidebar of the Firebase console shows various services: Authentication, Database, Storage, Hosting, Functions (which is currently selected), and ML Kit. Below these are Quality sections for Crashlytics, Performance, and Test Lab.

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 Functions?](#) for examples of use cases).
3. Receives a data object that contains a snapshot of the data stored in the specified database location.

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 will listen for events.

Set the event handler

Functions let you handle database events at two levels of specificity; you can listen for specific events like creation, update, or deletion events, or you can listen for any change of any kind to a node.



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深深/path
```

DATABASE TRIGGERS

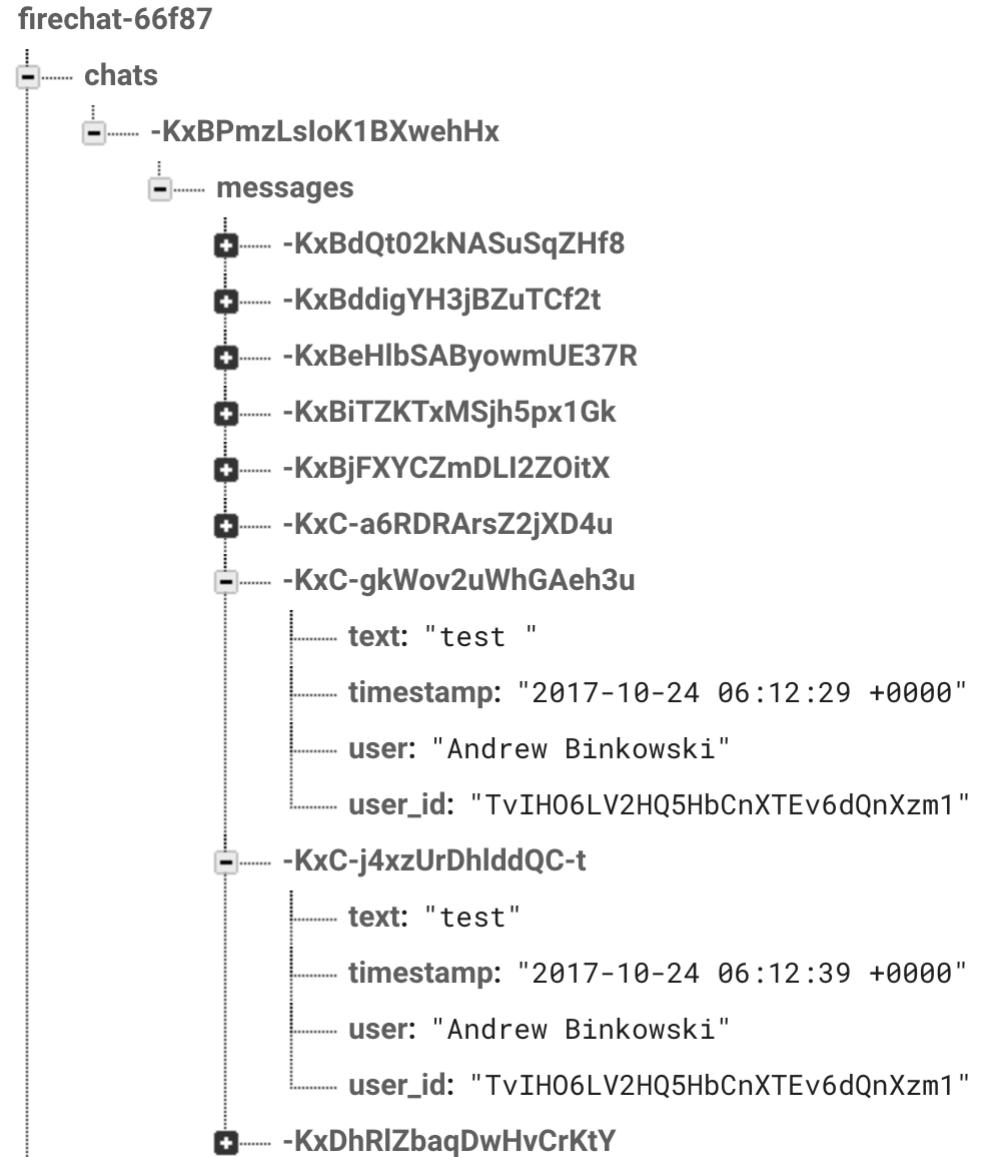
- Use wildcards to match keys
- Path available at `event.params`

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

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

DATABASE TRIGGERS

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

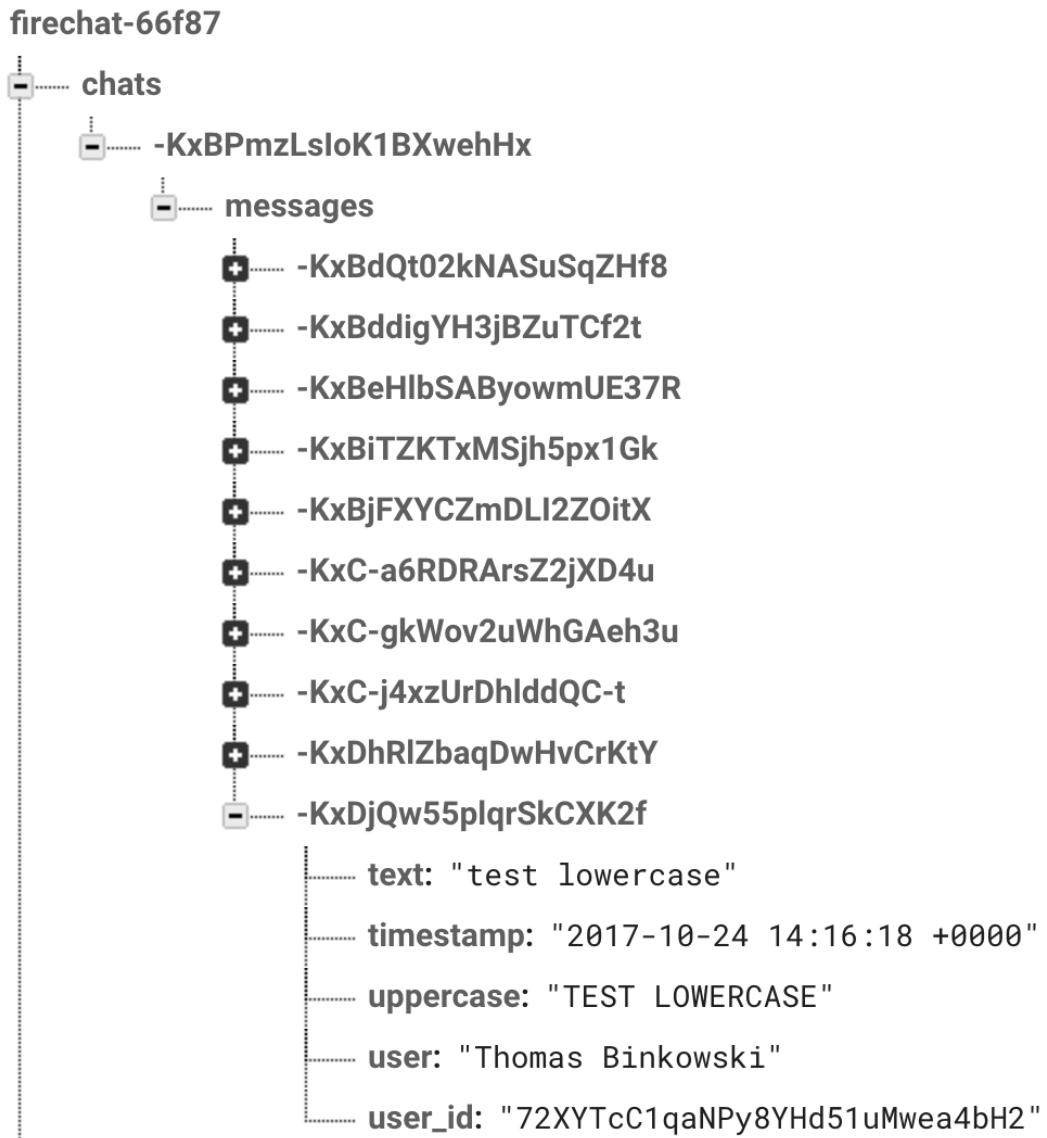


DATABASE TRIGGERS

```
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

 Search logs



All functions ▾

All log levels ▾



Time ↓ Level Function Event message

Oct 24, 2017

9:07:...  makeUp... ► Function execution took 1091 ms, finished with status: ...

9:07:...  makeUp... ▼ Uppercasing -KxBPmzLsIoK1BXwehHx -KxDhRlZbaqDwHvCrKtY
cloud function test

9:07:...  makeUp... ► Billing account not configured. External network is not...

9:07:...  makeUp... Function execution started

1:31:...  addMes... ► Function execution took 621 ms, finished with status co...

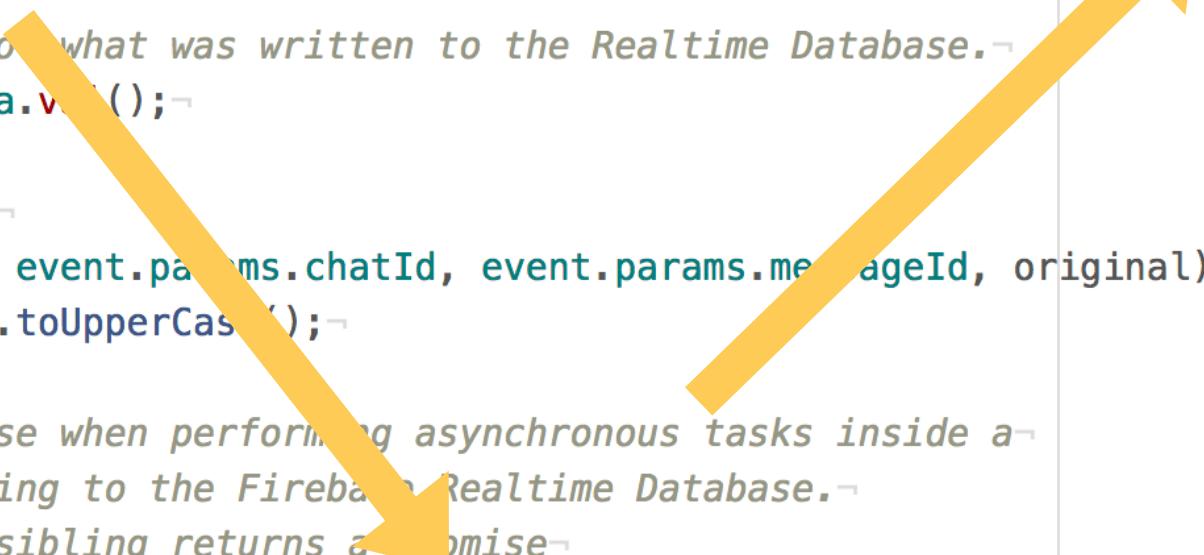
DATABASE TRIGGERS

- Careful that changes do not trigger infinite loop



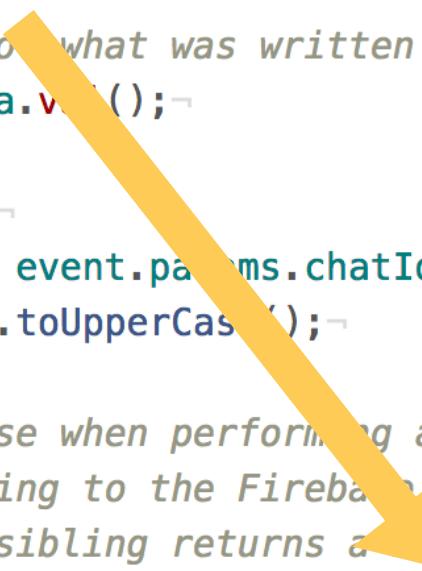
DATABASE TRIGGERS

```
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.value();  
  ....  
  ....// 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.value();  
  ....  
  ....// 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	Request https://us-central1-chat-cat-2f244.cloudfunctions.net/date	us-central1	Node.j...	256 MB	60s
helloWorld	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

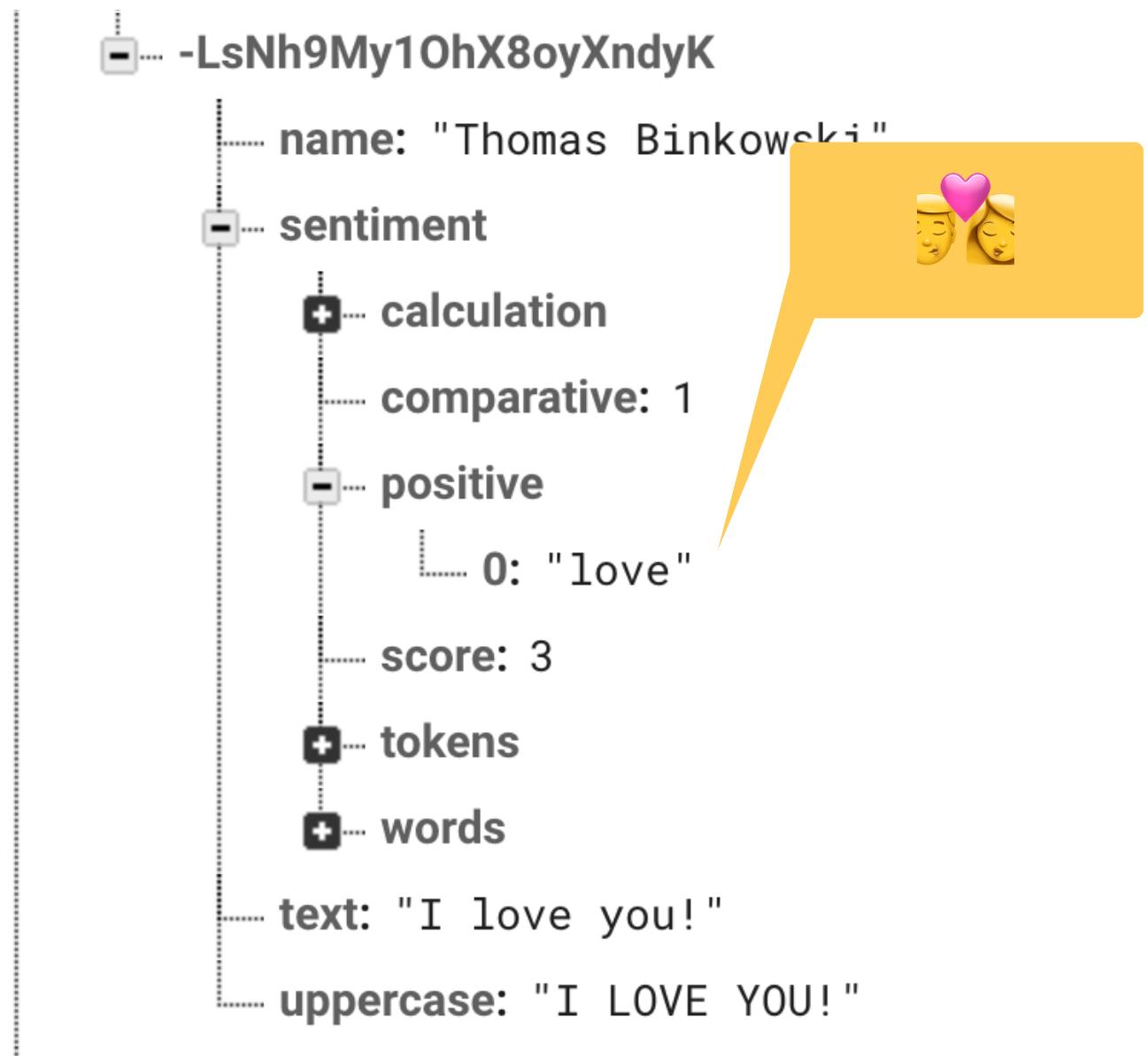
DATABASE TRIGGERS

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 REQUESTS 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

CURL [HTTPS://US-CENTRAL1-CHAT-CAT-2F244.CLOUDFUNCTIONS.NET/SIMPLEHTTP](https://us-central1-chat-cat-2f244.cloudfunctions.net/simplehttp)

```
% curl https://us-central1-chat-cat-2f244.cloudfunctions.net/simplehttp
essages": {"-LsKLkJ-4UFrmpqwE5l9": {"name": "Thomas Binkowski", "text": "This is
e."}, "-LsKM6K14uNQZY9vUvdt": {"name": "Thomas Binkowski", "text": "I am a test."}, "-LsLgkPmDIP_k": {"name": "Thomas Binkowski", "text": "Super important, -LsLgkPmDIP_k"}, "-LsLgkPmDIP_k": {"name": "Thomas Binkowski", "text": "try this !!!"}, "-LsKe0iN_Hlo_g": {"name": "Thomas Binkowski", "text": "what"}, "-LsKe8Eov4vRwBmjuVTX": {"name": "Thomas Binkowski", "text": "This is a test."}, "uppercase": "THIS IS A TEST."}, "-LsKffdsUF9g91J2xAtT": {"name": "Thomas Binkowski", "sentiment": {"calculation": [{"hate": -3}], "comparative": -1, "negative": true}, "score": -3, "tokens": ["i", "hate", "cats"], "words": ["hate"]}, "text": "I hate cats."}, "-LsKg4uP9aTCkwTd56Gk": {"name": "Thomas Binkowski", "sentiment": {"comparative": 0, "score": 0, "tokens": ["does", "this", "still", "work"]}, "text": "Does this still work."}, "uppercase": "DOES THIS STILL WORK"}, "-LsKgrKHSn9znbrSAk": {"name": "Andrew Binkowski", "sentiment": {"comparative": 0, "score": 0, "tokens": ["what", "you", "doing", "tonight"]}, "text": "What are you doing tonight?"}, "uppercase": "WHAT ARE YOU DOING TONIGHT?"}, "-LsKgu33A7140g8qj_ar": {"name": "Thomas Binkowski", "sentiment": {"comparative": 0, "score": 0, "tokens": ["nothing"]}, "text": "Nothing"}, "uppercase": "NOTHING"}, "-LsNZzX7v3z3T11vYT48": {"name": "Thomas Binkowski", "sentiment": {"comparative": 0, "score": 0, "tokens": ["this", "is", "a", "post", "about", "cats"]}, "text": "This is about cats."}, "uppercase": "THIS IS A POST ABOUT CATS!"}, "-LsN_BMgWnyNcH8wamnB": {"name": "Thomas Binkowski", "sentiment": {"comparative": 0, "score": 0, "tokens": ["hi"]}, "text": "Hi Cat!"}, "uppercase": "HI CAT!"}, "-LsNh9My1OhX8oyXndyK": {"name": "Thomas Binkowski", "sentiment": {"calculation": [{"love": 3}], "comparative": 1, "positive": true}, "score": 3, "tokens": ["i", "love", "you"], "words": ["love"]}, "text": "I love you!"}, "uppercase": "I LOVE YOU!"}, "-LsN_BMgWnyNcH8wamnB": {"name": "Andrew Binkowski", "lastMessage": 1572370076540}, "Binkowski": {"lastMessage": 157237627} % curl https://us-central1-627 % curl https://us-central1-chat-cat-2f244.cloudfunctions.net627 % curl https://us-central1-chat-
```

GET REQUESTS RETURN JSON DATA

DATABASE TRIGGERS

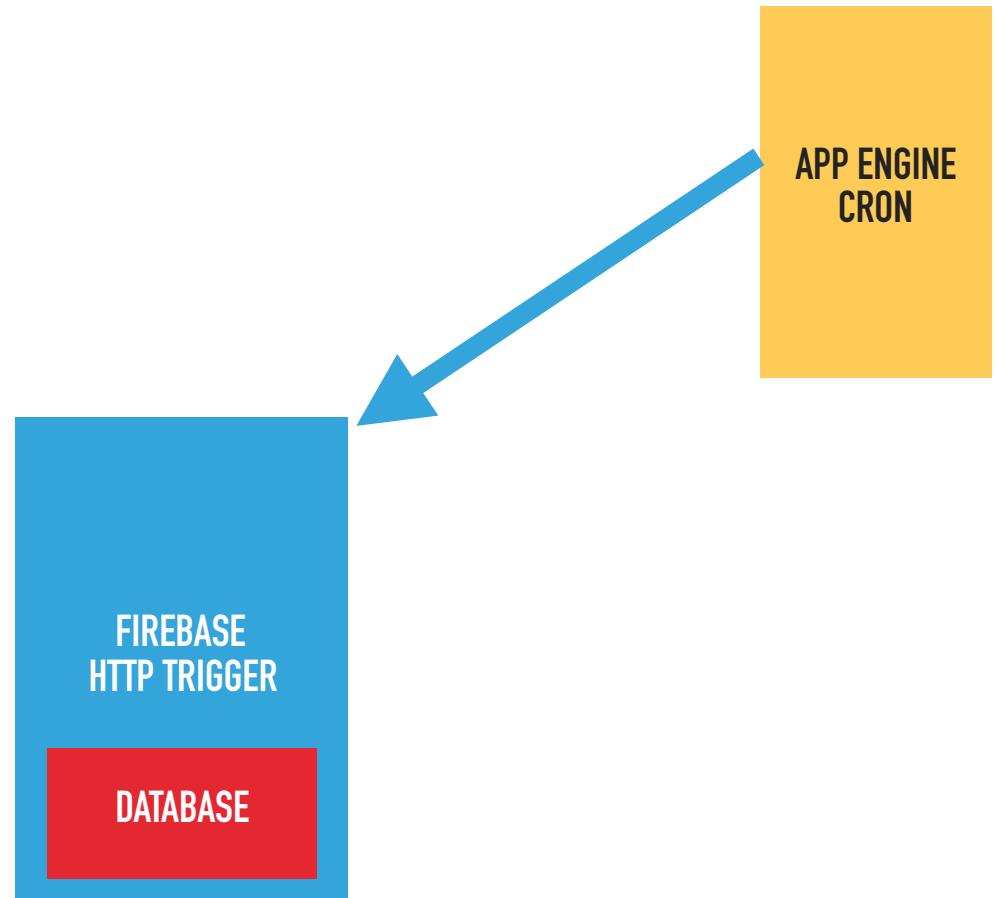
- Authentication trigger
- Welcome email via gmail API

```
// 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, 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('Children', data.numChildren)
    admin.database().ref('messages').child('total').set(data.numChildren)
    res.status(200).send('updated from cron')
  });
});
```

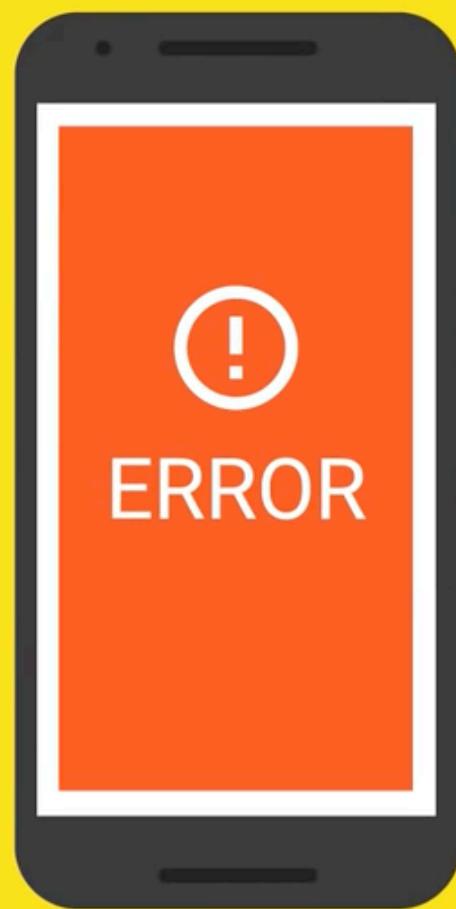
BREAK TIME



FIREBASE CRASHLYTICS CRASH REPORTING

FIREBASE CRASH REPORTING

- [#](https://www.youtube.com/watch?list=PLIK7zZEyLmOF_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

- Authentication
- Database
- Storage
- Hosting
- Functions
- ML Kit

- Quality
- Crashlytics
- Performance
- Test Lab
- App Distribution

- Analytics
- Dashboard



The most powerful, yet lightest weight crash reporting solution

[Learn more](#)

1 Are you a Fabric user migrating a Crashlytics app? [?](#)

No, set up a new Firebase app

Choose this if you're setting up Crashlytics for a new app

Yes, migrate my Fabric app to Firebase

Choose this if you are an existing Fabric user and want to migrate a Fabric Crashlytics app. Crash data will appear in both Fabric and Firebase dashboards.

[Next](#)

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
```

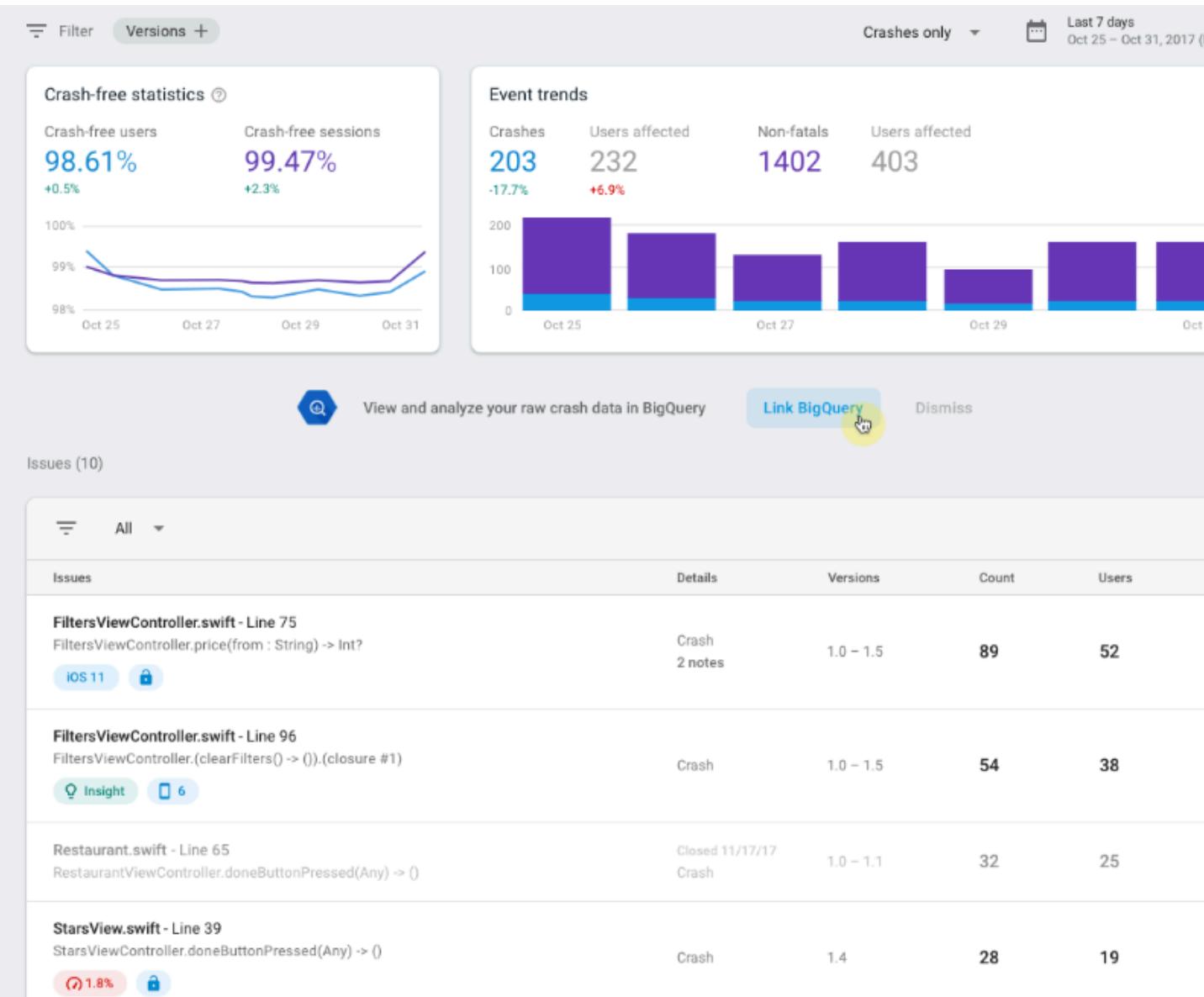
FIREBASE CRASH REPORTING

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

- Add additional information to crash messages
- Should be anonymous

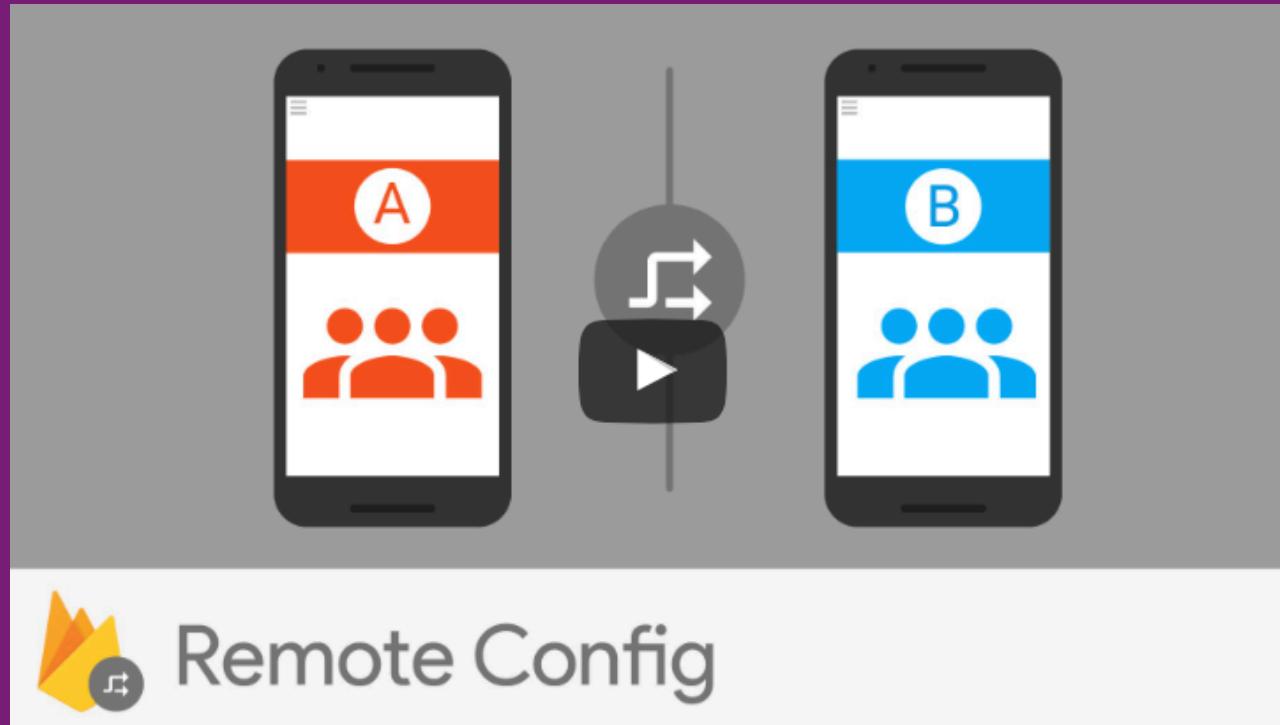
FIREBASE CRASH REPORTING

- Logs crash data
- Deobfuscation (human readable)



FIREBASE REMOTE CONFIG

REMOTE CONFIG



- https://youtu.be/_CXXVFPO6f0?list=PLI-K7zZEsvYLmOF_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 Server

Your App

Which value is fetched from the server?

1) Highest priority conditional value (if **true** for a given app instance)

2) Server-side default value (if present)



Which value does my app get?

1) Value fetched from the server (if activated)

2) In-app default value (set using **setDefaults**)

3) Static initialized value

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 Server

Your App

Which value is fetched from the server?

1) Highest priority conditional value (if **true** for a given app instance)

2) Server-side default value (if present)



Which value does my app get?

1) Value fetched from the server (if activated)

2) In-app default value (set using **setDefaults**)

3) Static initialized value

REMOTE CONFIG

- Use cases
- Staged rollout

Edit condition

Use conditions to provide different parameter values if a condition is met

Name Color

New search rollout group

Applies if...

User in random percentile <= 10 % DEF and

✖ Cancel Save condition

Parameter key <small>②</small>	Value for New search rollout group	Add value for condition <small>▼</small>
new_search_feature	true {} ✖	{} ✖
Add description	Default value	
	false {}	{}
✖	Cancel Update	
new_search_feature	 ↗ New search rollout group true	
	Default value false	

REMOTE CONFIG

The screenshot shows a list of remote configurations for a key named "promo_splash". The configurations are organized into five categories: "Android_English", "Android_French", "iOS_English", "iOS_French", and a "Default value". Each category is represented by a colored button (purple for Android, orange for iOS) followed by the configuration name and its corresponding URL.

Category	Configuration Name	URL
Android	Android_English	https://mywebsite.com/splash_android_en.jpg
Android	Android_French	https://mywebsite.com/splash_android_fr.jpg
iOS	iOS_English	https://mywebsite.com/splash_ios_en.jpg
iOS	iOS_French	https://mywebsite.com/splash_ios_fr.jpg
General	Default value	https://mywebsite.com/splash.jpg

- Use cases
 - Platform/language specific assets

REMOTE CONFIG

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

Edit condition

Use conditions to provide different parameter values if a condition is met

Name

Color

Post midnight 7 Feb



Applies if...

Date/Time



After

2/7/2019

12:01 AM

Los Angeles



new_promo_campaign_live

↳ Post midnight 7 Feb

true

Default value

false

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 Remote Config keys or parameter values. It is possible to accidentally expose sensitive data 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 Remote Config to store sensitive information that would require a user's explicit authorization. This includes things like payment information or location data.
- Don't store configuration values that could compromise the security or privacy of your users. For example, don't store sensitive information like API keys or user passwords.
- Don't attempt to circumvent the requirements of your app's target platform using Remote Config. This includes trying to run code that would normally be restricted by platform rules.

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

The screenshot shows the Firebase Remote Config interface. On the left, there's a sidebar with various analytics and growth features. The main area is titled "Remote Config" with a subtitle: "Customize and experiment with app behavior using server-side configuration parameters & feature flags". A central dialog box is titled "Add a parameter". It has fields for "Parameter key" (with an example value "holiday_promo_enabled") and "Default value" (set to "Other empty values"). There are also "Add value for condition" and "Add description" buttons.

- Events
- Conversions
- Audiences
- Funnels
- User Properties
- Latest Release
- Retention
- StreamView
- DebugView

Grow

- Predictions
- A/B Testing
- Cloud Messaging
- In-App Messaging

Remote Config

Customize and experiment with app behavior using server-side configuration parameters & feature flags

Add a parameter

Parameter key ?

Example: holiday_promo_enabled

Add value for condition ▼

Default value

Other empty values ▼ { }

Add description

REMOTE CONFIG

Remote Config

Publish changesDiscard all

Parameters

↗ Remote Config now works on Web! [Learn More](#)

Add parameter

dark_mode_beta	True
----------------	------

holiday_promo_enabled	True

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 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 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()

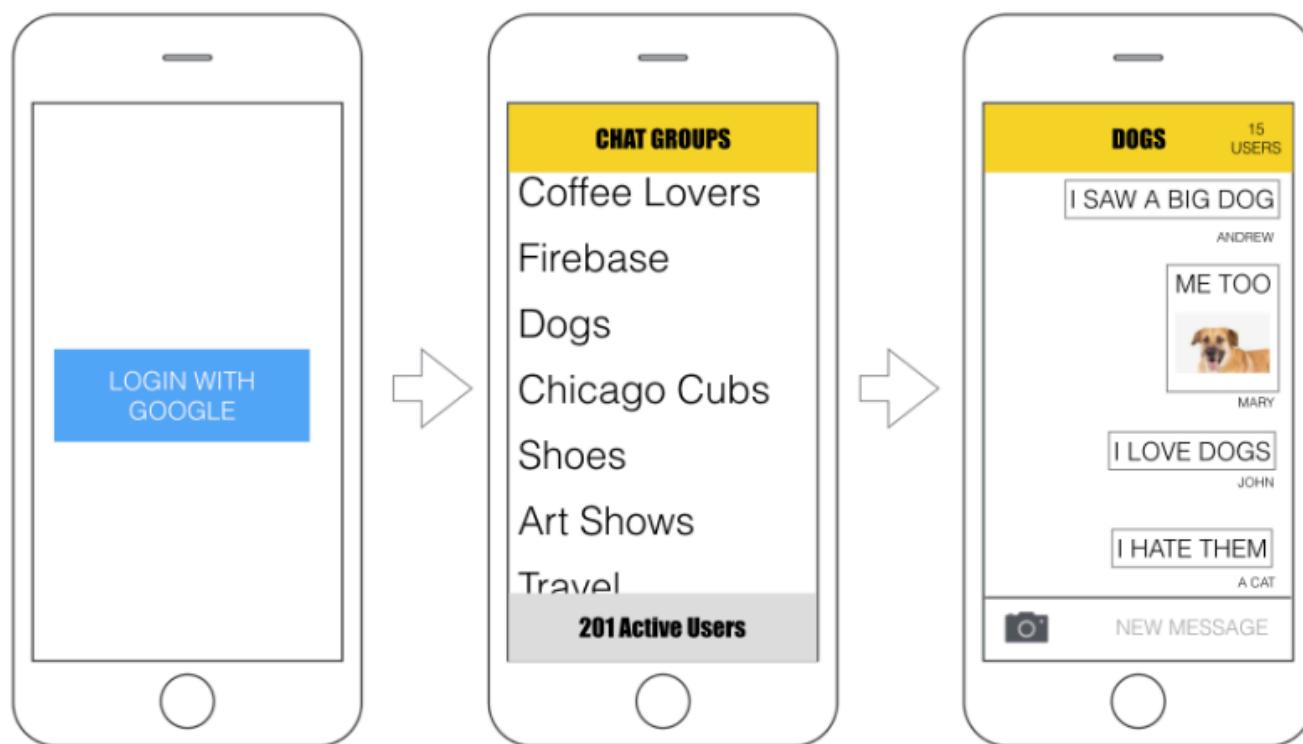
            } else {
                print("Config not fetched")
                print("Error \(error!.localizedDescription)")
            }
            print("Joke of the day: \(self.remoteConfig["joke_of_the_day"].stringValue!)")
        }
    }

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

FETCH CLOUD VALUES

ASSIGNMENT 5

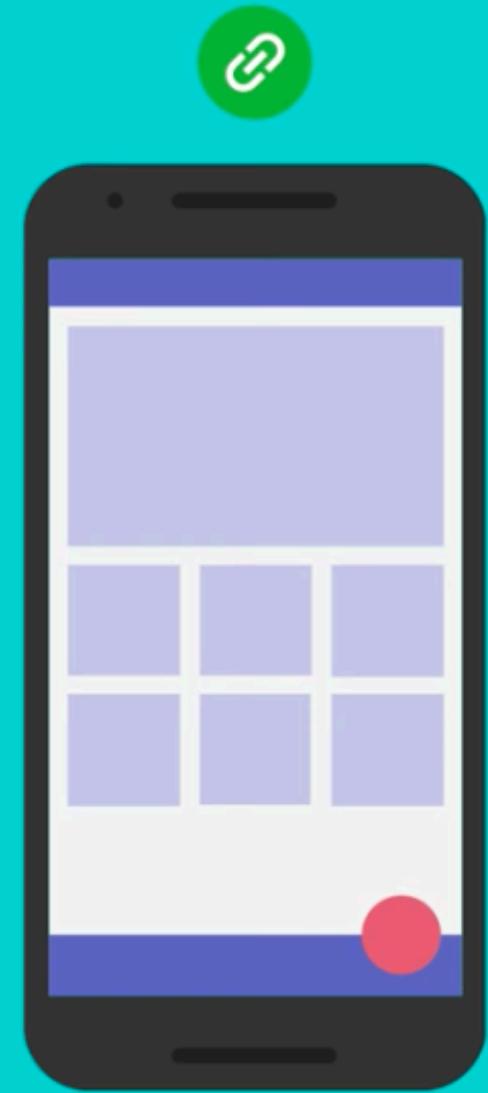
ASSIGNMENT 5



FIREBASE DYNAMIC LINKS

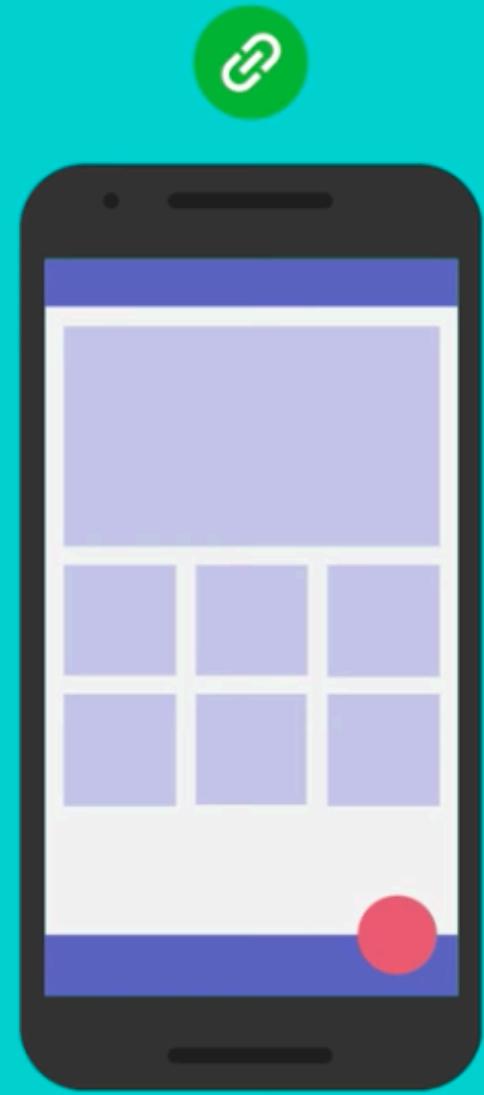
DYNAMIC LINKS

- Firebase dynamic links
 - https://youtu.be/LvY1JMcrPF8?list=PLIK7zZEsvYLmOF_07IayrTntevxtbUxDL



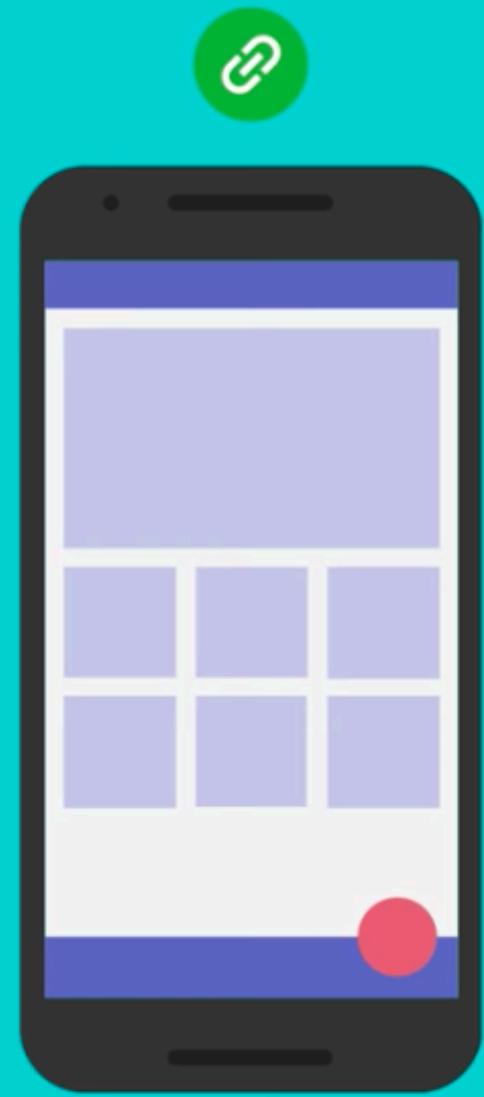
DYNAMIC LINKS

- 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

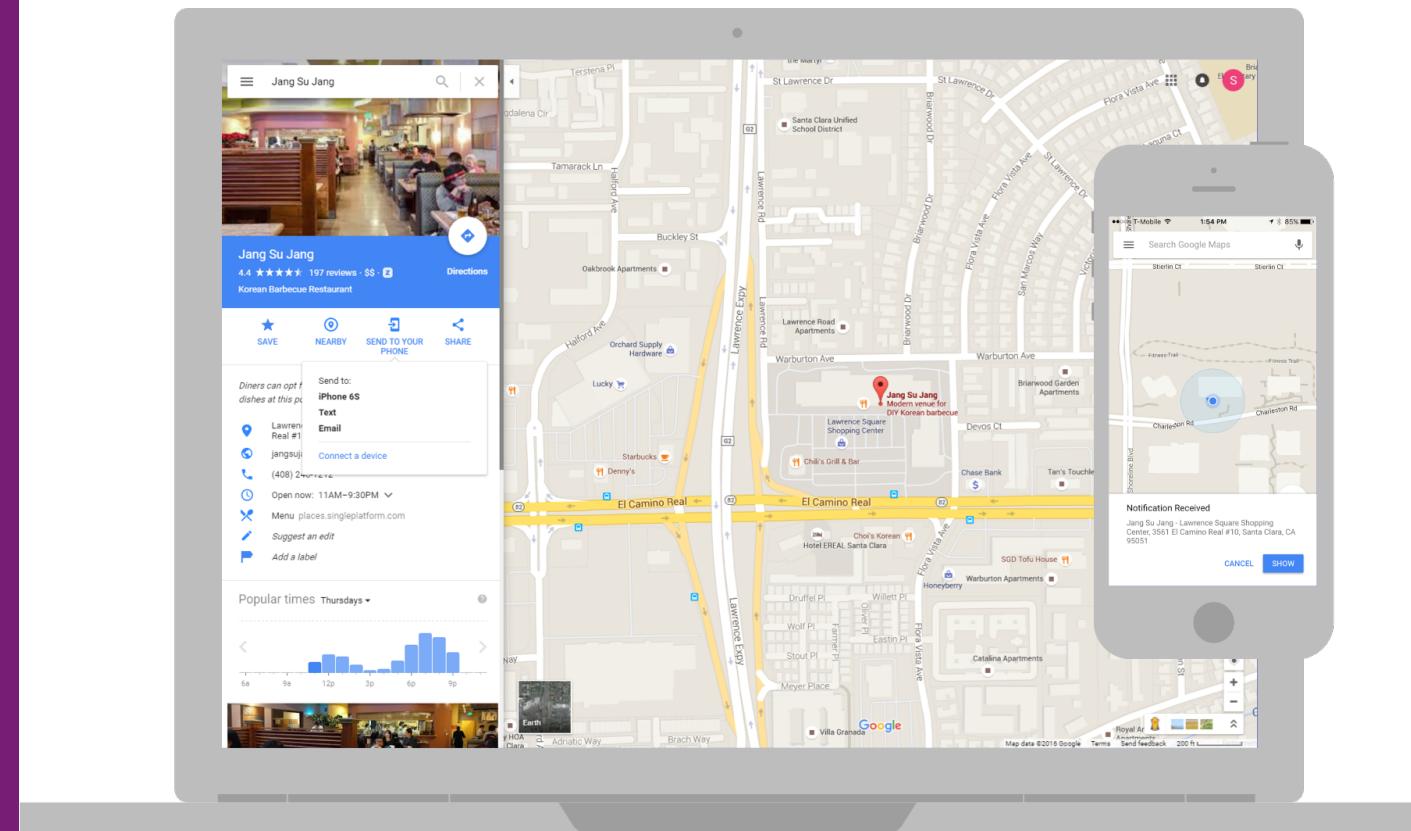
- Converting web (or desktop) users to app users



DYNAMIC LINKS

USE CASES

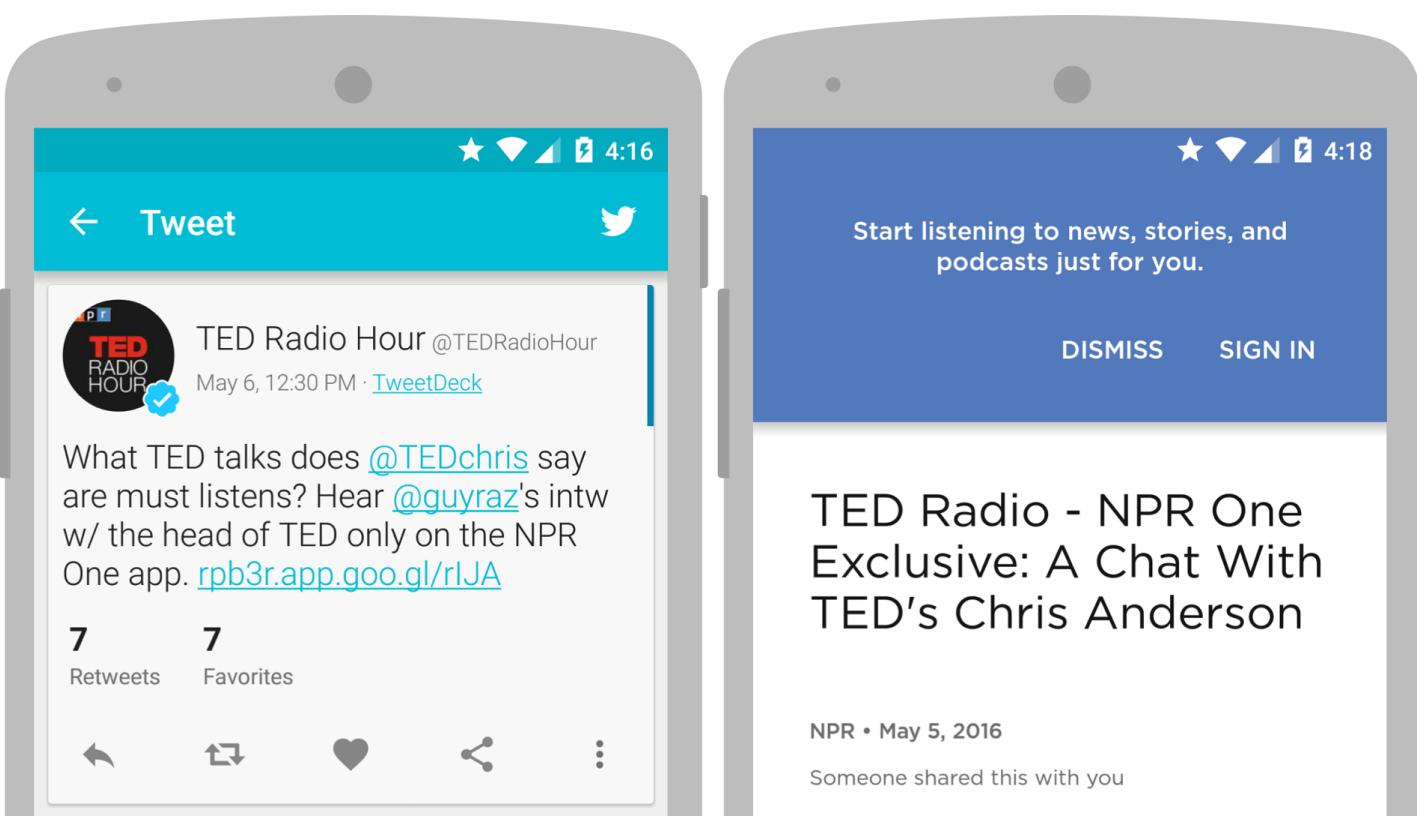
- Don't loose place



DYNAMIC LINKS

USE CASES

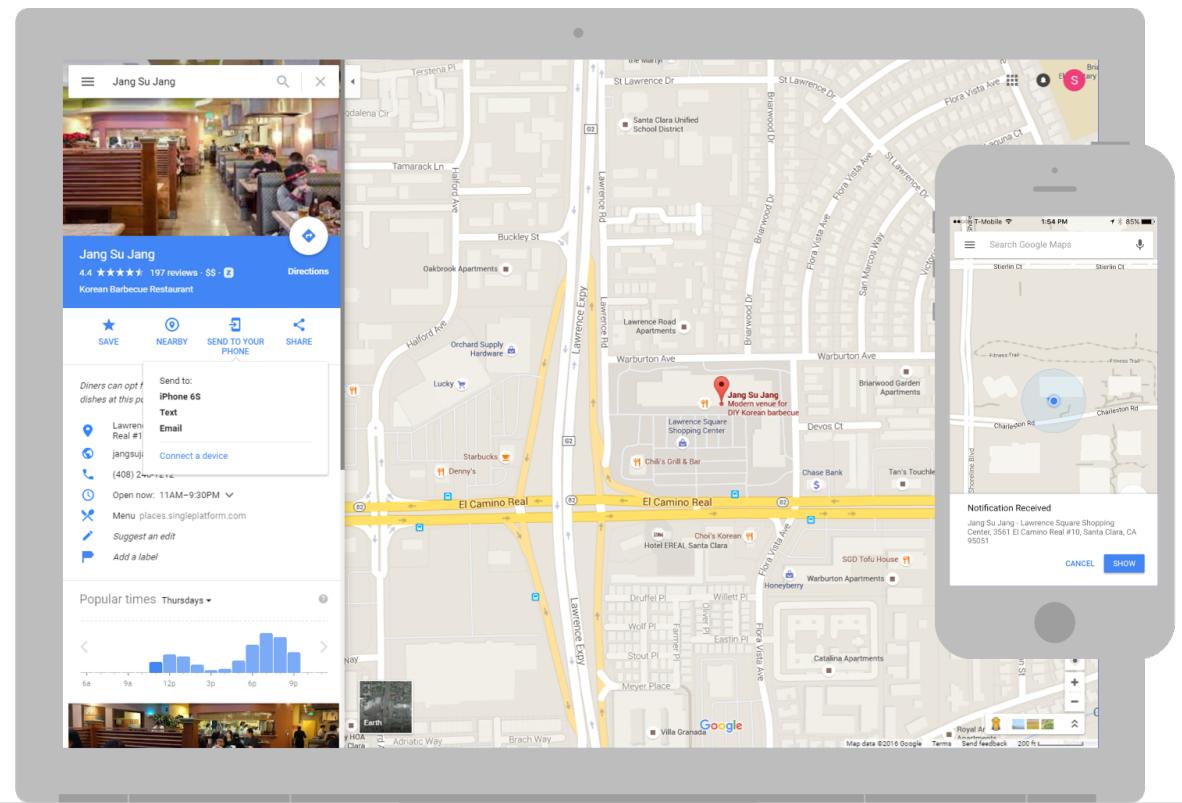
- Send links through social media
 - Direct content links
 - Track engagement



DYNAMIC LINKS

USE CASES

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



DYNAMIC LINKS

FIREBASE

WHERE WE WERE

- Open this page in our app!

APP

DYNAMIC LINKS

- <https://abc123.app.goo.gl/?link=https://example.com/invitation?gameid%3D1234%26referrer%3D555&apn=com.example.android&ibi=com.example.ios&isi=12345>

FIREBASE

ADDITIONAL DATA

APP

DYNAMIC LINKS

URL SHORTENER

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

DYNAMIC LINKS

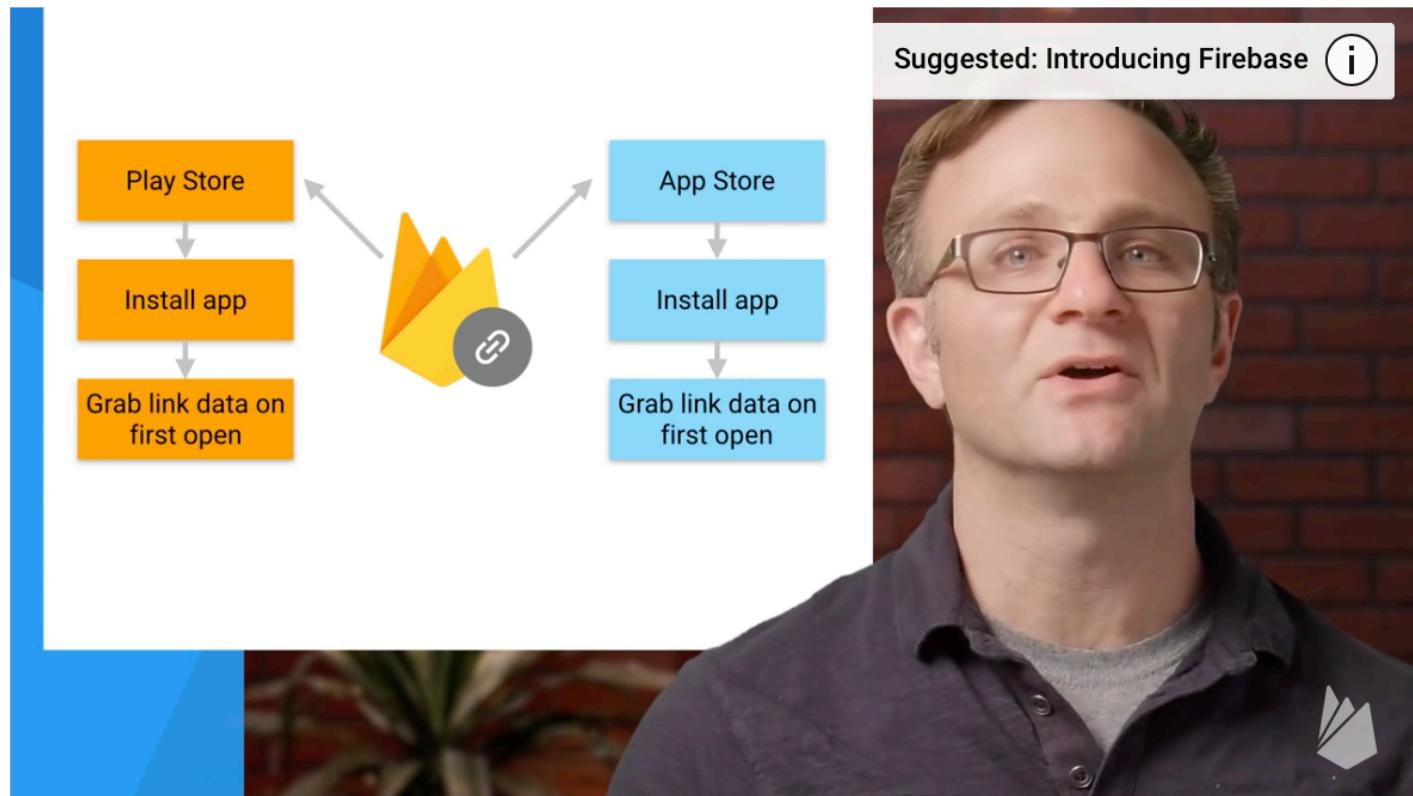
- Implementing deep links

- 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

`https://www.google.com`

Safari

`mailto:joe@gmail.com`

Mail

`tel:415-555-1212`

Phone

`com.example.myapp://stuffhere`

Your app!

SETTING UP DYNAMICS LINKS

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

The screenshot shows the 'Dynamic Links' section of the YouApp console. At the top, there is a dropdown menu labeled 'chat-cat' and a 'Go to' button. Below the title 'Dynamic Links', the URL 'https://chatcat.page.link' is displayed with a dropdown arrow. To the right of the URL are filters for 'Last 30 days' (Sep 29 – Oct 29) and a 'New Dynamic Link' button. A table header with columns 'Link name', 'URL', 'Created ↓', 'Clicks', 'First-opens', and 'Re-opens' follows. A message in the center of the table states 'You have no dynamic links under this URL prefix'.

SETTING UP DYNAMICS LINKS

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

Dynamic Links

Create

1 Set up your short URL link

Customize your short link URL to make it more professional and contextual. [Learn More ↗](#)

URL prefix

`https://chatcat.page.link` / coffee_lovers

Link preview

`https://chatcat.page.link/coffee_lovers`

[Next](#)

2 Set up your Dynamic Link

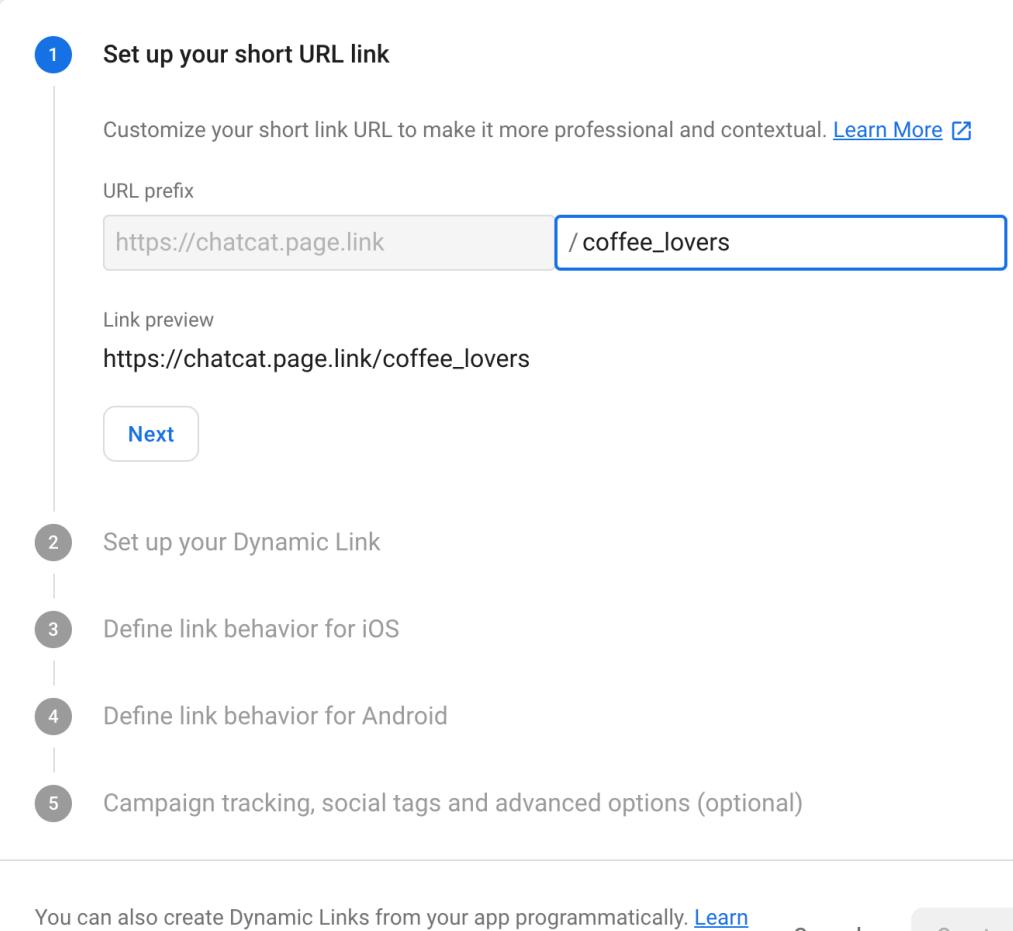
3 Define link behavior for iOS

4 Define link behavior for Android

5 Campaign tracking, social tags and advanced options (optional)

You can also create Dynamic Links from your app programmatically. [Learn more ↗](#)

[Cancel](#) [Create](#)



SETTING UP DYNAMICS LINKS

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

Dynamic Links
Create

1 Set up your short URL link

2 **Set up your Dynamic Link**

A Dynamic Link is a deep link into your app that works whether or not your app is installed. On desktop it will go to the deep link url. [Learn More ↗](#)

Deep link URL [?](#)
http://chatcat.page.link

Dynamic Link name [?](#)
Join Coffee Lovers

Previous **Next**

3 Define link behavior for iOS

4 Define link behavior for Android

5 Campaign tracking, social tags and advanced options (optional)

You can also create Dynamic Links from your app programmatically. [Learn](#)

SETTING UP DYNAMICS LINKS

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

The screenshot shows the Firebase Dynamic Links 'Create' page. The URL in the browser is `console.firebaseio.google.com/u/0/project/chat-cat-2f244/durablelinks/creation/https%3A%2F%2Fchatcat.page.link`. The left sidebar lists various Firebase services like Firestore, Functions, and Authentication. The main area is titled 'Create' and shows a step-by-step process:

- Set up your short URL link (Completed)
- Set up your Dynamic Link (Completed)
Join Coffee Lovers, Deep link URL: `http://chatcat.page.link`
- Define link behavior for iOS
 Open the deep link URL in a browser
 Open the deep link in your iOS App
iOS `mobi.uchicago.chat-cat`
Please add your App Store ID and Team ID [here](#)
- Define link behavior for Android
- Campaign tracking, social tags and advanced options (optional)

At the bottom, it says: You can also create Dynamic Links from your app programmatically. [Learn more](#) [Cancel](#) [Create](#)

SETTING UP DYNAMICS LINKS

- Project settings
- Uniquely identifies the developer

The screenshot shows the 'our apps' section of a configuration interface. On the left, under 'iOS apps', there is a card for 'Chat Cat' (mobi.uchicago.chat-cat). On the right, there is a summary panel for this app.

Download the latest config file [GoogleService-Info.plist](#)

This file contains configuration details such as keys and identifiers, for the services you just enabled.

App ID ②
1:323794022426:ios:3eb4527f6a34687c4ee1a3

App nickname
Chat Cat [edit](#)

Bundle ID
mobi.uchicago.chat-cat

App Store ID ②
123456789 [edit](#)

Team ID ②
723KYE3F9E [edit](#)

[Remove this app](#)

SETTING UP DYNAMICS LINKS

- Apple developer > Membership
- Team ID

Andrew Binkowski
University of Chicago (Department of Computer Science) ▾



Membership Details

Your team's membership information and legal agreements.

Membership Information

Program Type	iOS Developer University Program
Team Name	University of Chicago (Department of Computer Science)
Team ID	723KYE3F9E
Entity Type	Education / University
Phone	1-773-7026614
Address	1100 East 58th Street Chicago, Illinois 60637 United States

Device Reset Date

Team Agent	Andrew Binkowski
Your Role	Agent

[Need to edit this information?](#)

SETTING UP DYNAMICS LINKS

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

The screenshot shows the Firebase console interface. On the left, a sidebar lists various services: Overview, Analytics, ELOP, Authentication, Database, Storage, Hosting, Functions, Test Lab, Crash Reporting, Performance, Notifications, Remote Config, Dynamic Links, and AdMob. The 'Dynamic Links' option is highlighted. The main area displays 'Your apps' and an 'iOS apps' section. An app named 'mobi.uchicago.firechat' is selected. To its right, there's a button to 'Download the latest config file' (GoogleService-Info.plist). Below this, configuration fields include 'App ID' (1:487781121319:ios:78425b1380795a79), 'App nickname' (Add a nickname), 'Bundle ID' (mobi.uchicago.firechat), 'App Store ID' (284910350), 'App ID Prefix' (723KYE3F9E), 'Reporting currency' ((USD \$) US Dollar), and 'Time zone' ((GMT-05:00) GMT-05:00). A 'DELETE THIS APP' button is at the bottom right. A callout box points to the 'App Store ID' field with the text: 'You can find your App Store ID in your app's URL. In the example below, 123456789 is the App Store ID. https://itunes.apple.com/us/app/yourapp/id123456789'.

SETTING UP DYNAMICS LINKS

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

IF THE USER DOESN'T HAVE THE APP INSTALLED, ONE REDIRECT OPTION IS TO GO TO APP STORE

The screenshot shows the Firebase console interface. On the left, a sidebar lists various services: Overview, Analytics, ELOP, Authentication, Database, Storage, Hosting, Functions, Test Lab, Crash Reporting, Performance, Notifications, Remote Config, Dynamic Links, AdMob, and AdSense. The 'Dynamic Links' section is highlighted with a yellow arrow pointing towards the main content area. The main content area displays 'Your apps' under 'iOS apps'. An iOS icon followed by 'mobi.uchicago.firechat' is selected. To the right, there's a section titled 'Download the latest config file' with a download button labeled 'GoogleService-Info.plist'. Below this are fields for 'App ID' (1:487781121319:ios:78425b1380795a79), 'App nickname' (Add a nickname), 'Bundle ID' (mobi.uchicago.firechat), 'App Store ID' (284910350), and 'App ID Prefix' (723KYE3F9E). A callout box over the 'App Store ID' field contains the text: 'You can find your App Store ID in your app's URL. In the example below, 123456789 is the App Store ID. https://itunes.apple.com/us/app/yourapp/id123456789'. There are also fields for 'Reporting currency' (USD \$) and 'Time zone' (GMT-05:00). At the bottom right is a 'DELETE THIS APP' button.

SETTING UP DYNAMICS LINKS

- Dynamic links domain
- Hosts redirects based on client

The screenshot shows the Firebase Dynamic Links 'Create' wizard. The left sidebar lists 'Analytics' (Dashboard, Events, Conversions, Audiences, Funnels, User Properties, Latest Release, Retention, StreamView, DebugView) and 'Grow' (Predictions, A/B Testing, Cloud Messaging, In-App Messaging, Remote Config, Dynamic Links, AdMob). The 'Dynamic Links' option under Grow is highlighted. The main area shows step 3: 'Define link behavior for iOS'. It includes a list of steps: 1. Set up your short URL link (checked), 2. Set up your Dynamic Link (checked), and 3. Define link behavior for iOS (checked). Under step 3, there are two options: 'Open the deep link URL in a browser' (radio button) and 'Open the deep link in your iOS App' (radio button, selected). A dropdown menu shows 'iOS mobi.uchicago.chat-cat'. Below this, it says 'If your app is not installed, send the user to' with options: 'App Store page for your app' (radio button) and 'Custom URL'. Advanced Settings (optional) include: 'Open a different app if on iPad' (checkbox), 'Add App Store campaign or affiliate parameters' (checkbox), and 'Override the default custom URL scheme' (checkbox). Navigation buttons 'Previous' and 'Next' are at the bottom.

chat-cat ▾ Dynamic Links > Create

Go to docs

chat-cat

Dynamic Links > Create

1 Set up your short URL link

2 Set up your Dynamic Link
Join Coffee Lovers, Deep link URL: http://chatcat.page.link

3 Define link behavior for iOS

Open the deep link URL in a browser

Open the deep link in your iOS App

iOS mobi.uchicago.chat-cat

If your app is not installed, send the user to

App Store page for your app

Custom URL

Advanced Settings (optional)

Open a different app if on iPad

Add App Store campaign or affiliate parameters

Override the default custom URL scheme

Previous

Next

4 Define link behavior for Android

5 Campaign tracking, social tags and advanced options (optional)

SETTING UP DYNAMICS LINKS

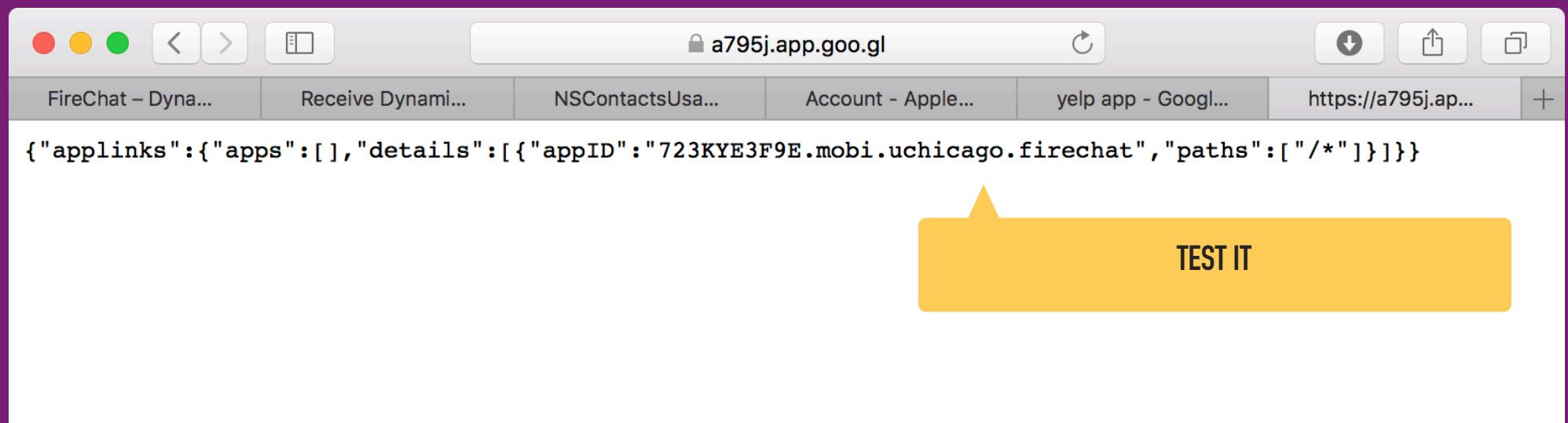
- Dynamic links domain
- Hosts redirects based on client

The screenshot shows the Firebase console interface for creating a dynamic link. On the left, there's a sidebar with navigation links: Test Lab, App Distribution, Analytics (with Dashboard selected), Grow (with Predictions, A/B Testing, Cloud Messaging, In-App Messaging, Remote Config, Dynamic Links selected, and AdMob), and Extensions. The main area is titled "Dynamic Links Create". It shows a step-by-step process:

- Set up your short URL link
- Set up your Dynamic Link
Join Coffee Lovers, Deep link URL: <http://chatcat.page.link>
- Define link behavior for iOS
Link directly in `mobi.uchicago.chat-cat`
- Define link behavior for Android
 - Open the deep link URL in a browser
 - Open the deep link in your Android App

Below the steps, there's a note about optional campaign tracking, social tags, and advanced options. At the bottom, there's a message about programmatically creating links, a "Create" button, and standard UI buttons for "Previous", "Next", "Cancel", and "more".

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 handle Universal Links
- Firebase is hosting our associated domain

The screenshot shows the Xcode Capabilities tab for a project named "FireChat". The "Associated Domains" section is expanded, showing the domain "applinks:a795j.app.goo.gl" entered into the "Domains" field. A note below the field says: "Steps: ✓ Add the Associated Domains entitlement to your entitlements file ✓ Add the Associated Domains feature to your App ID." Other entitlements listed include Apple Pay, In-App Purchase, Maps, Personal VPN, Network Extensions, Background Modes, Inter-App Audio, Keychain Sharing, App Groups, Data Protection, HomeKit, and HealthKit.

Entitlement	Status
Apple Pay	OFF
In-App Purchase	OFF
Maps	OFF
Personal VPN	OFF
Network Extensions	OFF
Background Modes	OFF
Inter-App Audio	OFF
Keychain Sharing	OFF
Associated Domains	ON
App Groups	OFF
Data Protection	OFF
HomeKit	OFF
HealthKit	OFF

SETTING UP DYNAMICS LINKS

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

The screenshot shows the Xcode Capabilities tab for a project named "FireChat". The "Associated Domains" section is expanded, showing a text input field with the value "applinks:a795j.app.goo.gl". A yellow callout bubble with the text "THIS REQUIRES COORDINATE WITH APPLE DEVELOPER PORTAL" points to this input field.

Capability	Status
Apple Pay	OFF
In-App Purchase	OFF
Maps	OFF
Personal VPN	OFF
Network Extensions	OFF
Background Modes	OFF
Inter-App Audio	OFF
Keychain Sharing	OFF
Associated Domains	ON

Domains: `applinks:a795j.app.goo.gl`

Steps:

- ✓ Add the Associated Domains entitlement to your entitlements file
- ✓ Add the Associated Domains feature to your App ID.

THIS REQUIRES COORDINATE WITH APPLE DEVELOPER PORTAL

SETTING UP DYNAMICS LINKS

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

The screenshot shows the Xcode Identity & Capabilities editor. On the left, there's a sidebar with categories: Keys, Identifiers, Devices, and Provisioning Profiles. Under Identifiers, 'App IDs' is selected, showing two entries: 'XC mobi uchicago -016-ShareExtension Fa...' and 'XC mobi uchicago -016-ShareExtension To...'. Under Devices, 'All' is selected, listing Apple TV, Apple Watch, iPad, iPhone, and iPod Touch. Under Provisioning Profiles, both 'All' and 'Development' are listed.

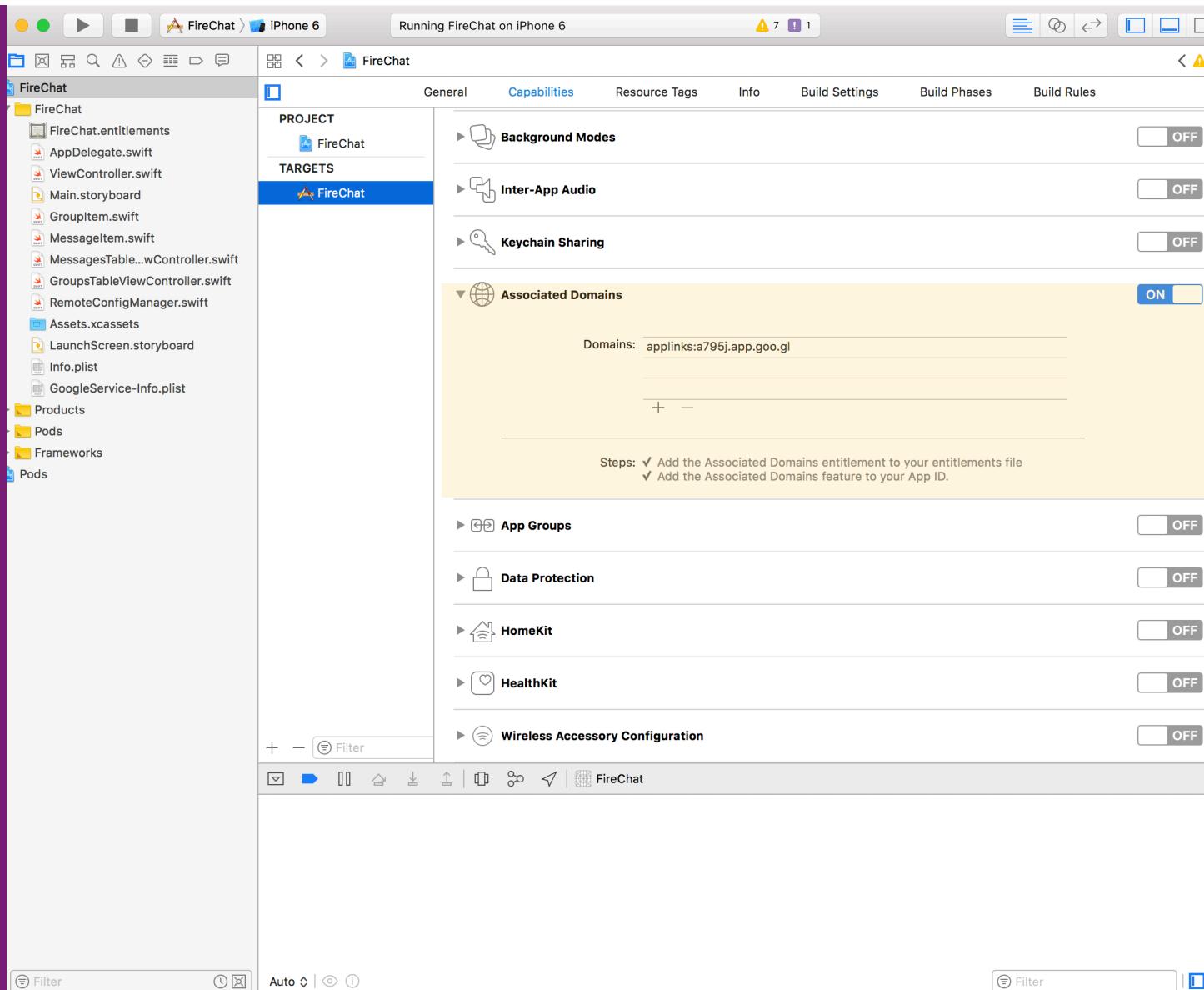
The main pane displays the 'XC mobi uchicago FireChat' configuration. It includes fields for Name (XC mobi uchicago FireChat), Prefix (723KYE3F9E), and ID (mobi.uchicago.FireChat). Below this is a table of application services:

Service	Development	Distribution
App Groups	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Associated Domains	<input checked="" type="radio"/> Enabled	<input checked="" type="radio"/> Enabled
Data Protection	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Game Center	<input checked="" type="radio"/> Enabled	<input checked="" type="radio"/> Enabled
HealthKit	<input type="radio"/> Disabled	<input type="radio"/> Disabled
HomeKit	<input type="radio"/> Disabled	<input type="radio"/> Disabled
iCloud	<input type="radio"/> Disabled	<input type="radio"/> Disabled
In-App Purchase	<input checked="" type="radio"/> Enabled	<input checked="" type="radio"/> Enabled
Inter-App Audio	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Personal VPN	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Push Notifications	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Wallet	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Wireless Accessory Configuration	<input type="radio"/> Disabled	<input type="radio"/> Disabled

At the bottom, there's an 'Edit' button and another row of entries: 'XC mobi uchicago instaWatchItOnTv' and 'mobi.uchicago.instaWatchItOnTv'.

SETTING UP DYNAMICS LINKS

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



SETTING UP DYNAMICS LINKS

Domains

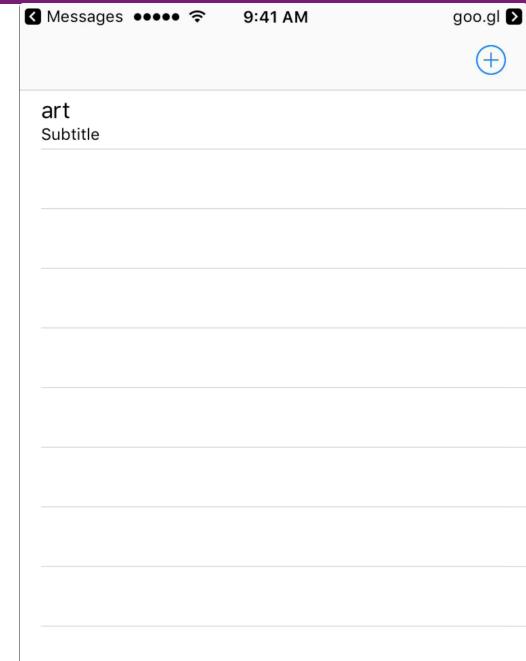
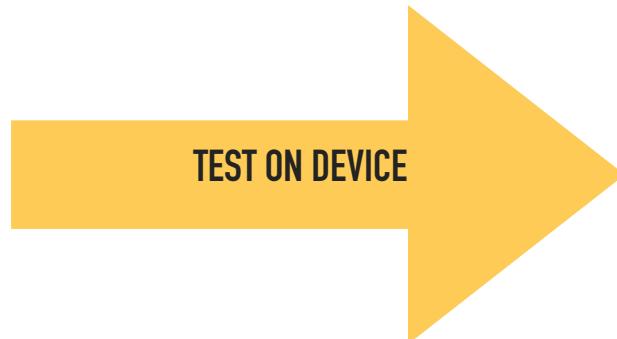
Domains: `applinks:a795j.app.goo.gl`

+

Steps: ✓ Add the Associated Domains entitlement to your entitlements file
✓ Add the Associated Domains feature to your App ID.

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

SETTING UP DYNAMICS LINKS



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 handle Universal Links
- Firebase is hosting our associated domain

https://a795j.app.goo.gl/ ⓘ



Dynamic Links are URLs that reduce friction and get users to relevant screens of your app whether or not it is installed

ⓘ [Learn more](#)

NEW DYNAMIC LINK

CREATING A DYNAMIC LINK

FIREBASE DYNAMIC LINKS

IOS UNIVERSAL LINKS

ANDROID LINKS

CREATING A DYNAMIC LINK

- Set up Dynamic link in the console

Dynamic Links

← Create Dynamic Link ?

1 Set up your Dynamic Link

A Dynamic Link is a deep link into your app that works whether or not your app is installed. On desktop it will go to the deep link url. [Learn More](#)

Deep link URL ?

Example: `https://yourapp.com/welcome`

Dynamic Link name ?

Arts Chat

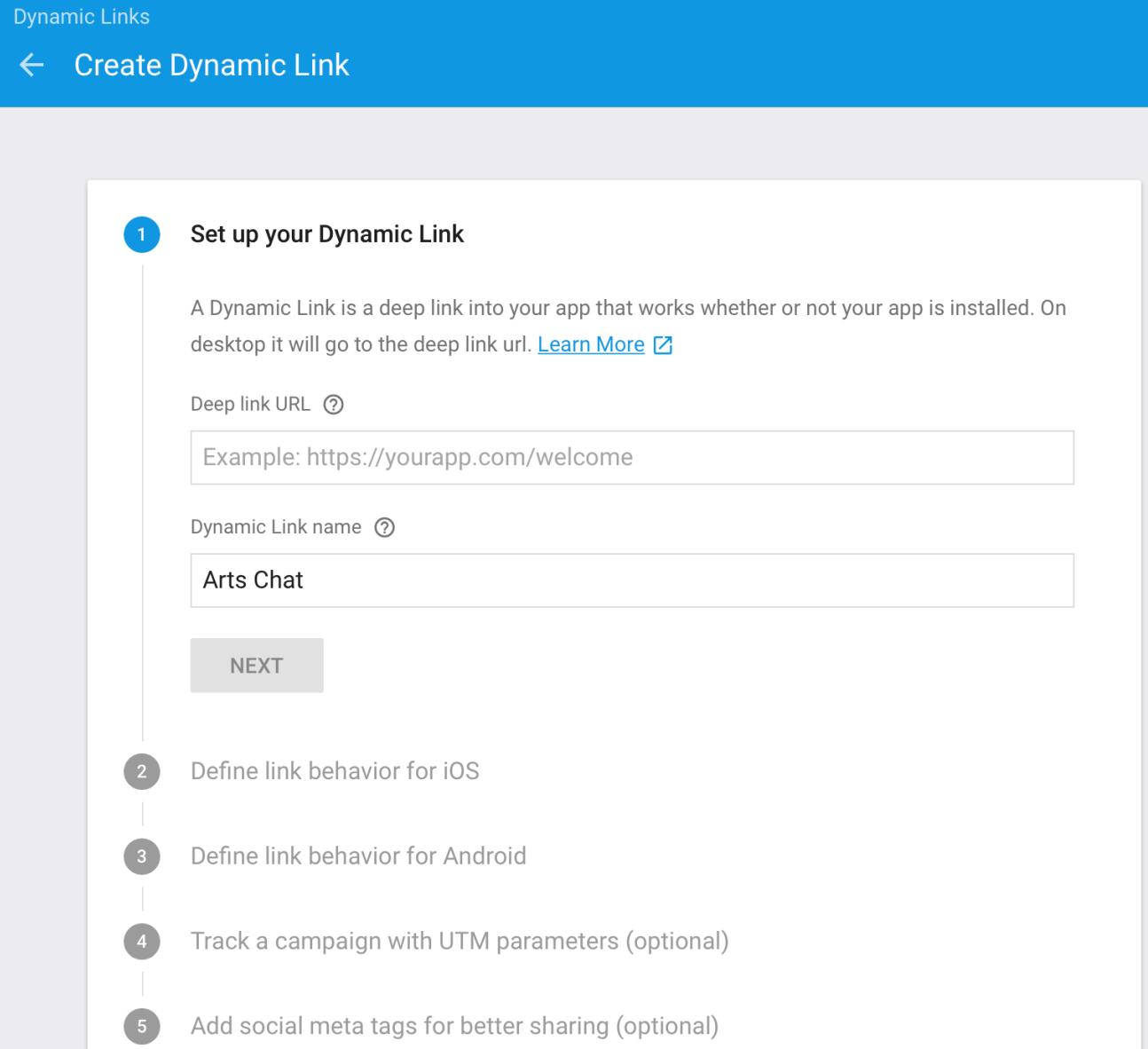
NEXT

2 Define link behavior for iOS

3 Define link behavior for Android

4 Track a campaign with UTM parameters (optional)

5 Add social meta tags for better sharing (optional)



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

The screenshot shows the Firebase console interface. On the left, there's a sidebar with various services: Overview, Analytics, Authentication, Database, Storage, Hosting, Functions, Test Lab, Crash Reporting, Notifications, Remote Config, Dynamic Links (which is highlighted in blue), and AdMob. The main area is titled "Dynamic Links" and "Create Dynamic Link". It has a step-by-step guide:

- Set up your Dynamic Link**

A Dynamic Link is a deep link into your app that works whether or not your app is installed. On desktop it will go to the deep link url. [Learn More ↗](#)

Deep link URL [?](#)

Dynamic Link name [?](#)

NEXT
- Define link behavior for iOS
- Define link behavior for Android
- Track a campaign with UTM parameters (optional)
- Add social meta tags for better sharing (optional)

You can also create Dynamic Links from your app programmatically. [Learn more ↗](#)

CANCEL CREATE DYNAMIC LINK

CREATING A DYNAMIC LINK

- Define the behavior of the link in the selected application

1 Set up your Dynamic Link
Art Group, Deep link URL: <http://uchicago.mobi/groups/art>

2 Define link behavior for iOS

Open the deep link URL in a browser
 Open the deep link in your iOS App

ios mobi.uchicago.firechat ▾

If your app is not installed, send the user to

App Store page for your app
 Deep link URL ⓘ
 Custom URL ⓘ

Advanced Settings (optional)

Open a different app if on iPad
 Add App Store campaign or affiliate parameters
 Use a custom scheme when universal links aren't supported ⓘ

[PREVIOUS](#) [NEXT](#)

3 Define link behavior for Android
Open the deep link URL in a browser

CREATING A DYNAMIC LINK

- Analytics tags
- Mirrors Google Analytics campaign tracking

1 Set up your Dynamic Link
Art Group, Deep link URL: <http://uchicago.mobi/groups/art>

2 Define link behavior for iOS
Link directly in `mobi.uchicago.firechat`

3 Define link behavior for Android
Open the deep link URL in a browser

4 Track a campaign with UTM parameters (optional)

Track the traffic sources and campaigns that send users to your app

Campaign source (`utm_source`) [?](#) Campaign medium (`utm_medium`) [?](#)

Example: Google Example: cpc

Campaign name (`utm_campaign`) [?](#)

Example: Spring sale

[PREVIOUS](#) [NEXT](#)

5 Add social meta tags for better sharing (optional)

You can also create Dynamic Links from your app programmatically. [Learn more](#) ↗

[CANCEL](#) [CREATE DYNAMIC LINK](#)

CREATING A DYNAMIC LINK

- Social media tags and data

- ✓ Set up your Dynamic Link
Art Group, Deep link URL: <http://uchicago.mobi/groups/art>
- ✓ Define link behavior for iOS
Link directly in `mobi.uchicago.firechat`
- ✓ Define link behavior for Android
Open the deep link URL in a browser
- 4 Track a campaign with UTM parameters (optional)
- 5 Add social meta tags for better sharing (optional)

These tags create a preview of your link when shared on social media. These values will override existing social meta tags for your destination link.

Preview title (st) [?](#)

Example: Holiday Promotion

Preview image URL (si) [?](#)

Example: <https://yoururl.com/ima>

Preview description (sd) [?](#)

Example: Get 25% off when you spend \$50 or more!

CREATING A DYNAMIC LINK

Dynamic Links ?

<https://a795j.app.goo.gl/> ? NEW DYNAMIC LINK

Link name	Created ↑	URL	⌚ Clicks (30 days)
Art Group	Apr ...	https://a795j.app.goo.gl/...	0

- 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>

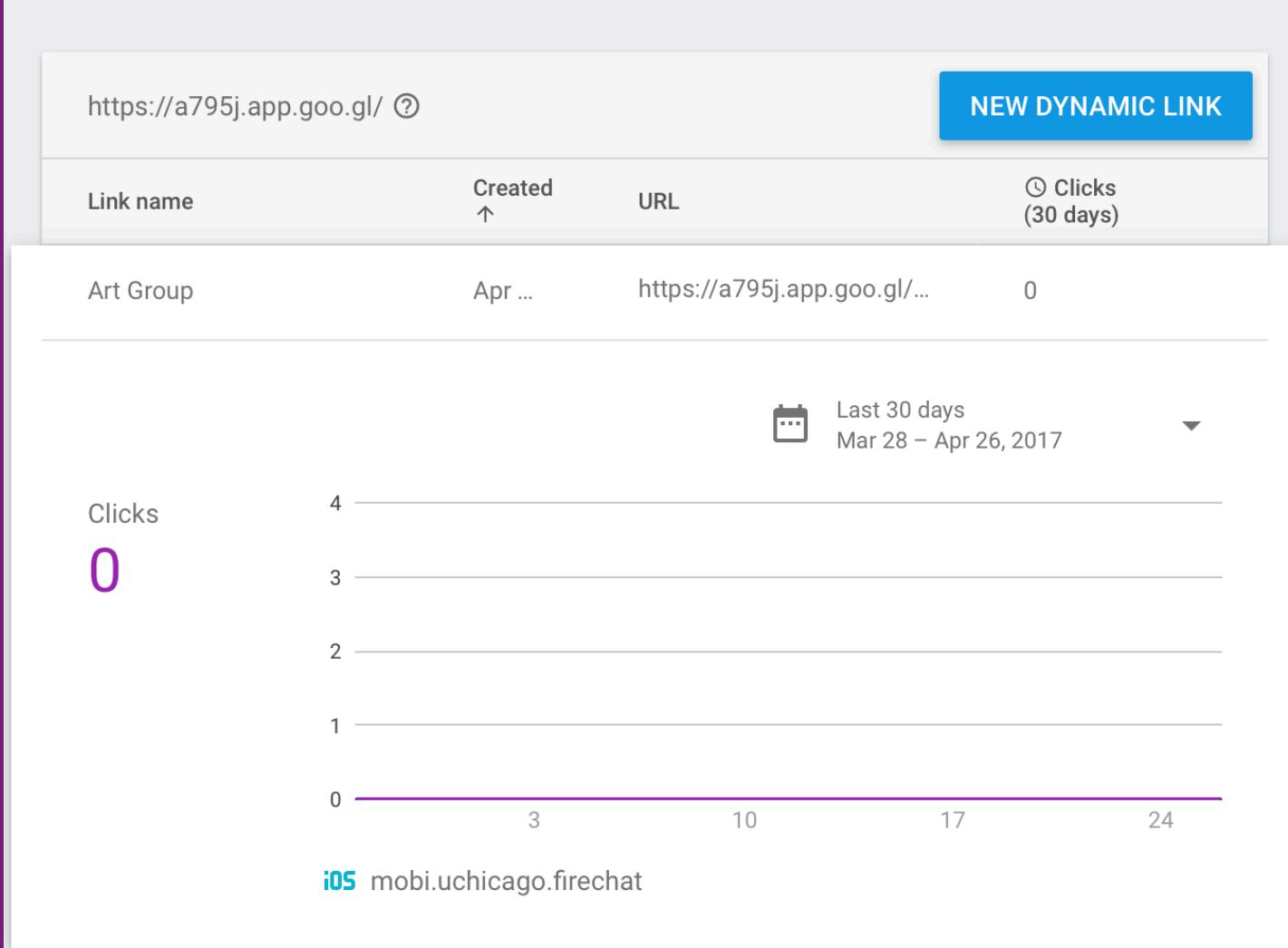
Short Dynamic Link

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

SHORTENED LINK

CREATING A DYNAMIC LINK

- Analytics for specific link



HANDLING DYNAMIC LINKS

HANDLING DYNAMIC LINKS

`https://abcde.app.goo.gl/tyPW`



Dynamic Links library



`FIRDynamicLink
object`

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'

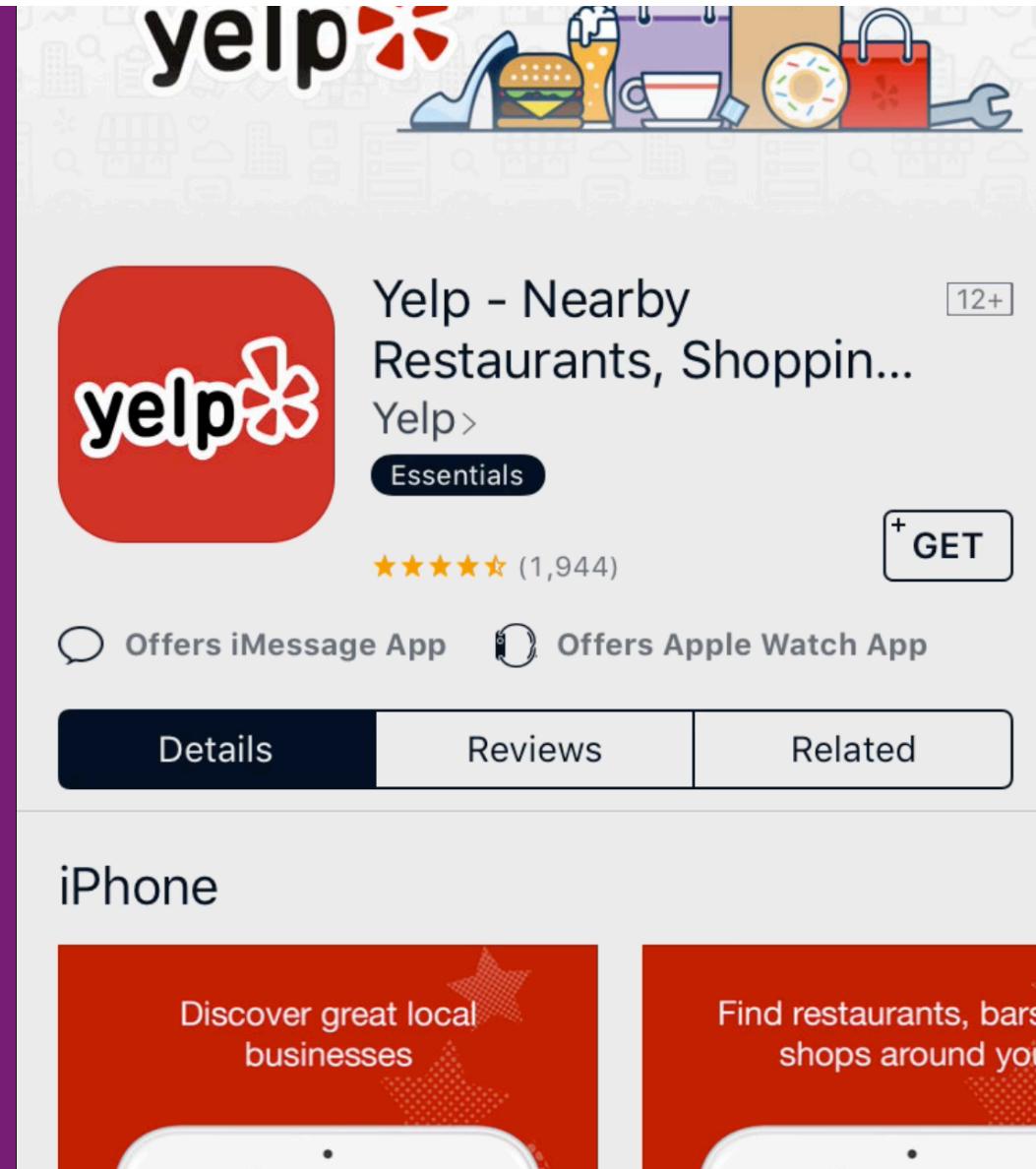
  pod 'Firebase/DynamicLinks'

  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.handleIncomingLink(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.handleIncomingLink(dynamicLink!)
        print("深深的 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 its components
func handleIncomingLink(_ 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

```
matchConfidence = weak  
"20% off coupon"
```

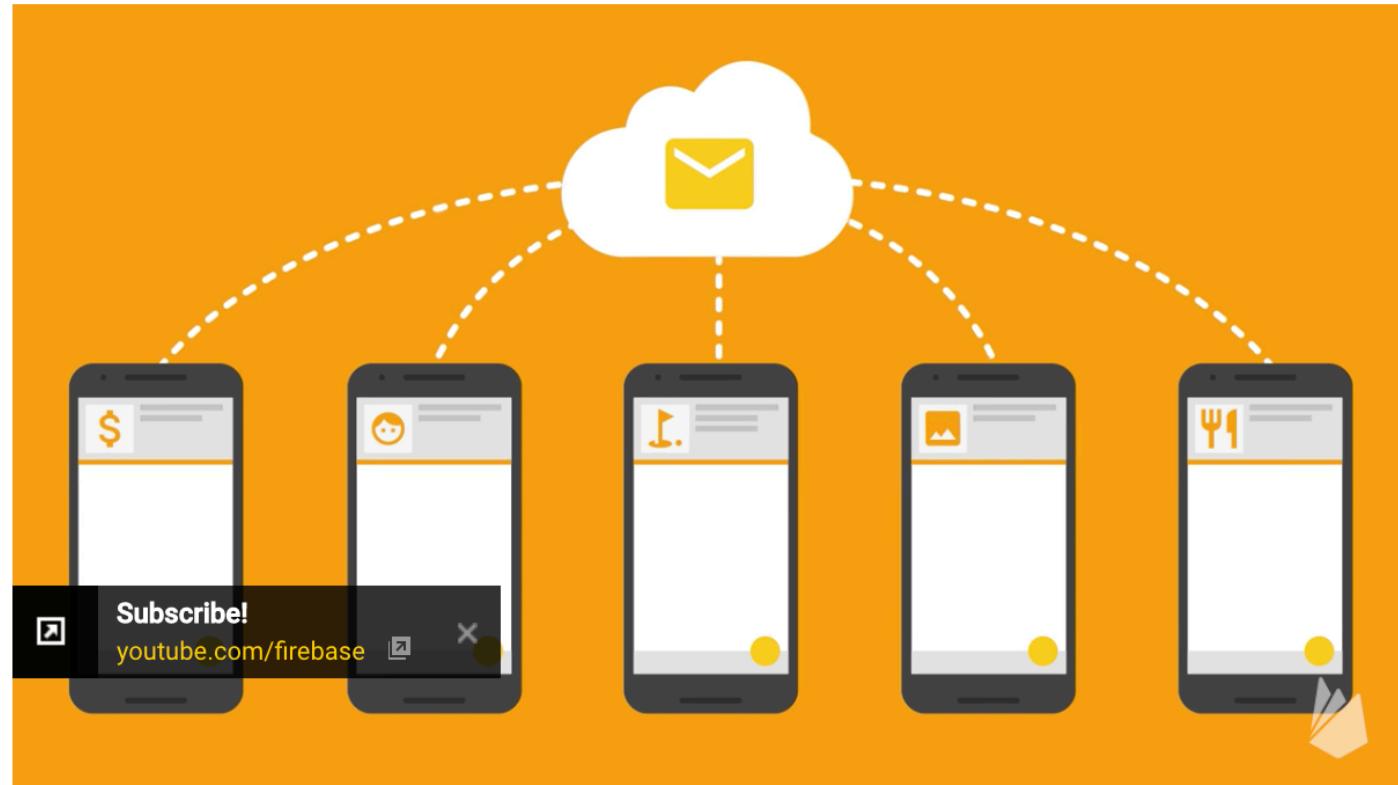
Probably okay

```
matchConfidence = weak  
"Shared by Joan Smith!"
```

FIREBASE CLOUD MESSAGING

FIREBASE CLOUD MESSAGING

- https://youtu.be/sioEY4tWmLI?list=PLI-K7zZEesYLmOF_07layrTntevxtbUxDL



SETTING UP FOR APNS

SETTING UP APNS

- Messaging uses native APNS services to deliver messages
 - Default

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

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)

Programming Guide

Communicating with APNs

The APNs provider API lets you send remote notification requests to APNs. APNs forwards your 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 request from your app to your provider, that contains a JSON payload and a device token. APNs forwards the request to your provider, that contains a JSON payload and a device token. APNs forwards your notification 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 private keys. A provider authentication token describes connections using tokens.

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

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

- 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 [provider identifier](#)
- The *issued at* (*iat*) registered claim key, whose value indicates the time 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 encode the token using the `base64url` encoding scheme. You must use the Elliptic Curve Digital Signature Algorithm (ECDSA) with the P-256 curve and the SHA-256 hash function to sign the token. Specify the value `ES256` in the algorithm header key (*alg*). For information on how to generate a provider authentication token, see ["Configure push notifications"](#) in Xcode Help.

FIREBASE CLOUD MESSAGING

Finished running FireChat on T. Andrew Binkowski's iPhone 6

Identity and Type

- Name: FireChat
- Location: Relative to Group
- Full Path: /Users/tabinkowski/Google Drive/g-Teaching/uchicago.cloud/mpcs51033-2017-autumn/mpcs51033-2017-autumn-code-samples.firebaseio/FireChat2/FireChat.xcodeproj

Project Document

- Project Format: Xcode 3.2-compatible
- Organization: T. Andrew Binkowski
- Class Prefix:

View Controller - A controller that manages a view.

Storyboard Reference - Provides a placeholder for a view controller in an external storyboard.

Navigation Controller - A controller that manages navigation through a hierarchy of views.

General Capabilities Resource Tags Info Build Settings Build Phases Build Rules

PROJECT Targets

iCloud

Push Notifications ON

Game Center

Wallet

Siri

Apple Pay

In-App Purchase

Maps

Personal VPN

FIREBASE CLOUD MESSAGING

Name	Type	Expires
Alice Chang (Alice's MacBook Air)	iOS Development	Jun 07, 2018
Andrew Binkowski (T.'s MacBook Pro)	iOS Development	Mar 20, 2018

Pending

Development

Production

Keys

All

Identifiers

App IDs

Pass Type IDs

iCloud Containers

App Groups

Devices

All

Apple TV

Apple Watch

FIREBASE CLOUD MESSAGING

iOS, tvOS, watchOS ▾

 Certificates

- All
- Pending
- Development
- Production

 Keys

- All

 Identifiers

- App IDs
- Pass Type IDs
- iCloud Containers
- App Groups

Add iOS Certificate

Select Type Request Generate Download

 What type of certificate do you need?

Development

- iOS App Development**
Sign development versions of your iOS app.
- Apple Push Notification service SSL (Sandbox)**
Establish connectivity between your notification server and the Apple Push Notification service sandbox environment to deliver remote notifications to your app. A separate certificate is required for each app you develop.

FIREBASE CLOUD MESSAGING

- Pending
- Development
- Production

 **Keys**

- All

 **Identifiers**

- App IDs
- Pass Type IDs
- iCloud Containers
- App Groups

 **Devices**

- All
- Apple TV
- Apple Watch
- iPad



Which App ID would you like to use?

All App IDs that you want to enable for remote notifications require their own Apple Push Notification service SSL certificate. The App ID-specific SSL certificate allows your server to connect to the Apple Push Notification service. Note that only explicit App IDs with a specific Bundle Identifier can be used to create an Apple Push Notification service SSL certificate.

Select an App ID for your Apple Push Notification service SSL Certificate (Sandbox)

App ID:

FIREBASE CLOUD MESSAGING

- Pending
- Development
- Production

Keys

- All

Identifiers

- App IDs
- Pass Type IDs
- iCloud Containers
- App Groups

Devices

- All
- Apple TV
- Apple Watch
- iPad



About Creating a Certificate Signing Request (CSR)

To manually generate a Certificate, you need a Certificate Signing Request (CSR) file from your Mac. To create a CSR file, follow the instructions below to create one using Keychain Access.

Create a CSR file.

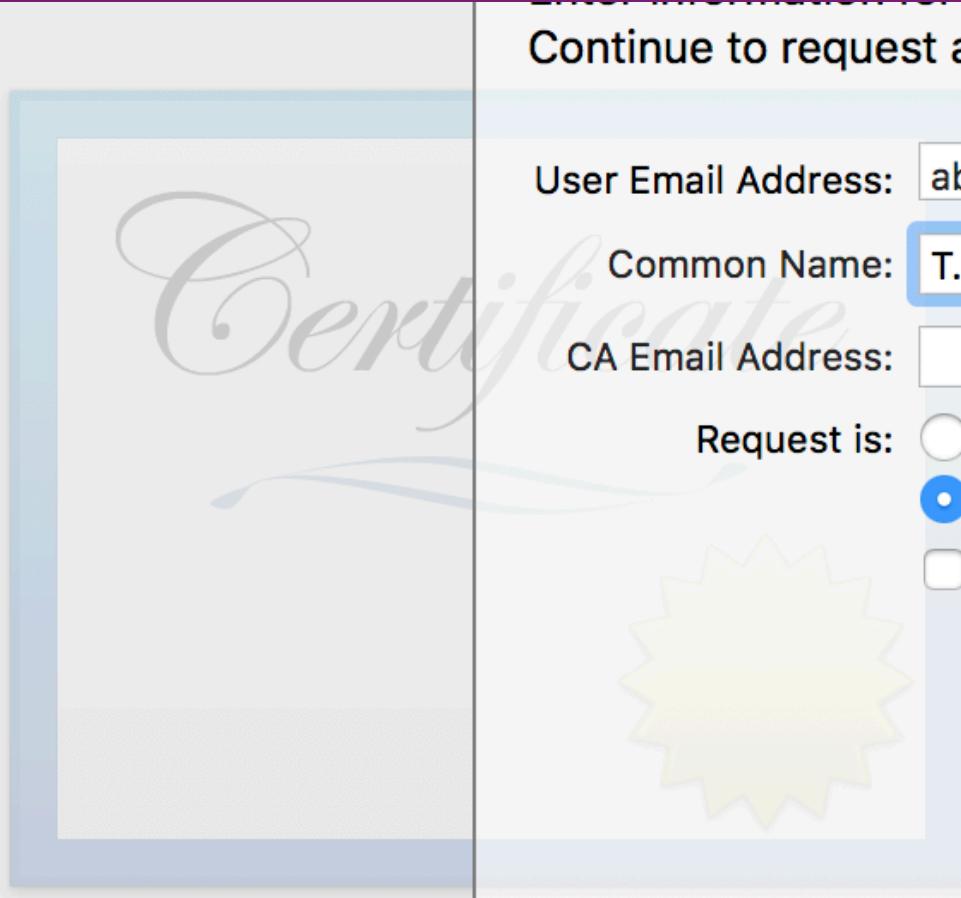
In the Applications folder on your Mac, open the Utilities folder and launch Keychain Access.

Within the Keychain Access drop down menu, select Keychain Access > Certificate Assistant > Request a Certificate from a Certificate Authority.

- In the Certificate Information window, enter the following information:
 - In the User Email Address field, enter your email address.
 - In the Common Name field, create a name for your private key (e.g., John Doe Dev Key).
 - The CA Email Address field should be left empty.
 - In the "Request is" group, select the "Saved to disk" option.
- Click Continue within Keychain Access to complete the CSR generating process.

FIREBASE CLOUD MESSAGING

Continue to request a certificate from the CA.



User Email Address:

Common Name:

CA Email Address:

Request is:

Emailed to the CA

Saved to disk

Let me specify key pair information

FIREBASE CLOUD MESSAGING

- Pending
- Development
- Production

Keys

- All

Identifiers

- App IDs
- Pass Type IDs
- iCloud Containers
- App Groups

Devices

- All
- Apple TV
- Apple Watch
- iPad



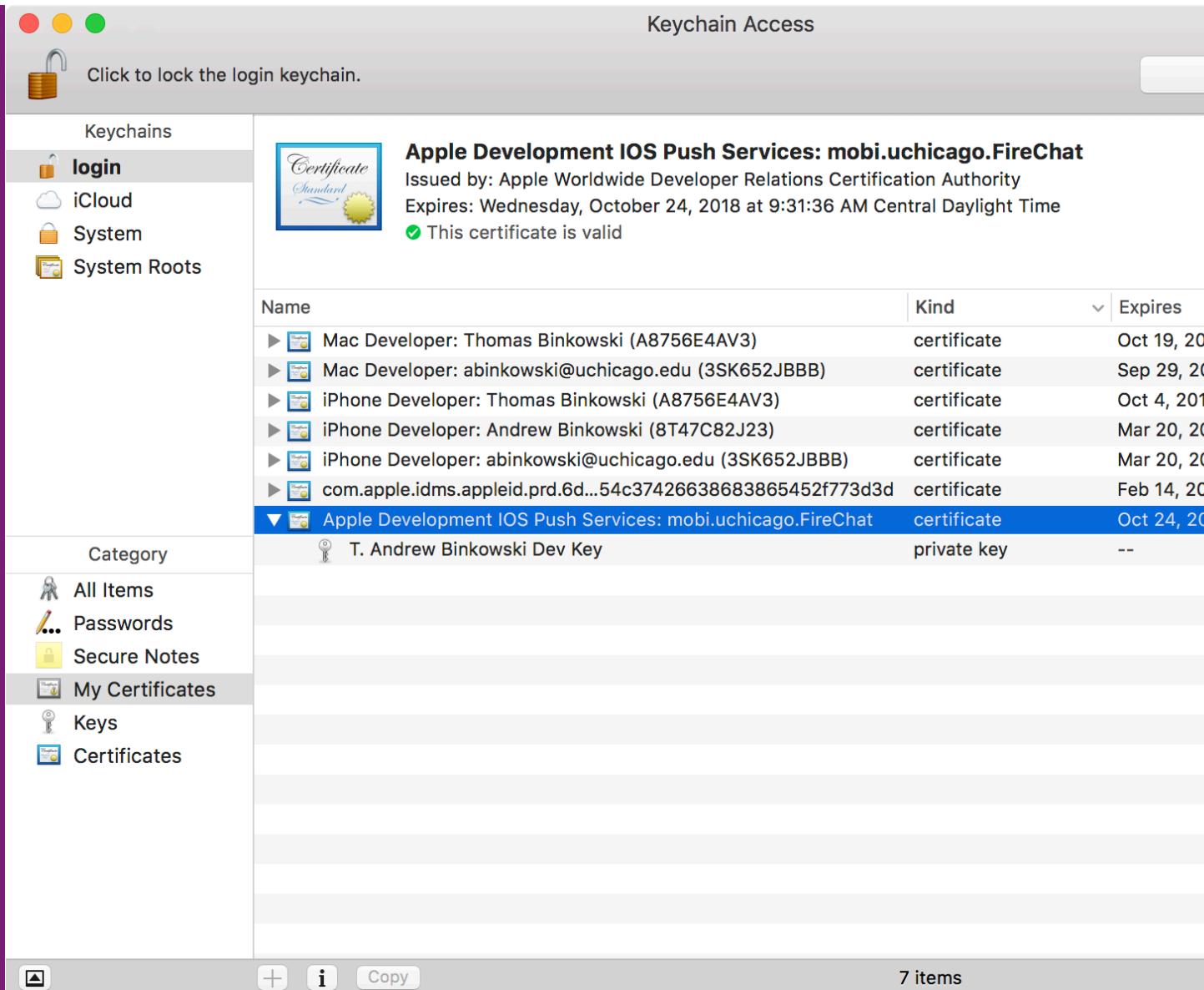
Generate your certificate.

When your CSR file is created, a public and private key pair is automatically generated. Your private key is stored on your computer. On a Mac, it is stored in the login Keychain by default and can be viewed in the Keychain Access app under the "Keys" category. Your requested certificate is the public half of your key pair.

Upload CSR file.
Select .certSigningRequest file saved on your Mac.

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

Storage

Hosting

Functions

Test Lab

Crash Reporting

Performance

GROW

Notifications

Remote Config

Dynamic Links

EARN

iOS apps



mobi.uchicago.firechat

Not configured for Cloud ...

Firebase Cloud Messaging can use either an APNs authentication key or APNs certificate to connect with APNs

APNs Authentication Key



Configuration with auth keys is recommended as they are the more current method for sending notifications to iOS

File

Key ID

App ID
prefix

No APNs auth key

UPLOAD

FIREBASE CLOUD MESSAGING

 Database

 Storage

 Hosting

 Functions

 Test Lab

 Crash Reporting

 Performance

GROW

 Notifications

 Remote Config

 Dynamic Links



Manage notification campaigns and send messages to engage the right users at the right moment

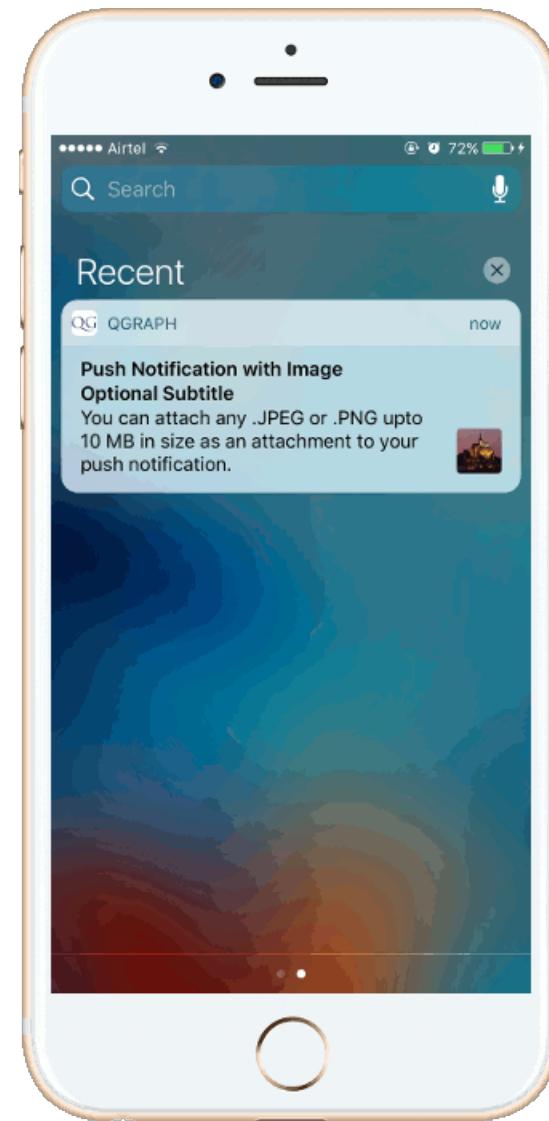
 [Learn more](#)  [View the docs](#)

 SEND YOUR FIRST MESSAGE

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

```
//  
// 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

The screenshot shows the Firebase Notifications dashboard under the 'Notifications' tab. The left sidebar lists various services: Overview, Analytics, Authentication, Database, Storage, Hosting, Functions, Test Lab, Crash Reporting, Performance, GROW, Notifications (which is selected and highlighted in blue), Remote Config, and Dynamic Links. The main area displays a table of notifications with the following data:

Message	Status	Delivery date	Platform
This is a test with the new Firebase ...	✓ Completed	Oct 24, 2017 10:54 A...	iOS
This is pretty cool.	✓ Completed	Oct 24, 2017 10:52 A...	iOS
teste	✓ Completed	Oct 24, 2017 10:51 A...	iOS
9	✓ Completed	Oct 24, 2017 10:50 A...	iOS
8	✓ Completed	Oct 24, 2017 10:50 A...	iOS
6	✓ Completed	Oct 24, 2017 10:48 A...	iOS
6	✓ Completed	Oct 24, 2017 10:47 A...	iOS

SEND FIREBASE NOTIFICATIONS

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

Message text

Message label (optional) ?

Delivery date ?

Send Now ▾

Target

User segment Topic Single device

Target user if...

App AND

TARGET ANOTHER APP

Conversion events ?

▼

FIREBASE CLOUD MESSAGING

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

```
func swizzled_application(_ application: UIApplication,  
didRegisterForRemoteNotificationsWithDeviceToken deviceToken: Data) {  
  
    // Receive the APNs deviceToken  
    // Send it off to FCM  
    // Get back an FCM deviceToken  
}
```

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