Backend-as-a-Service

Google Firebase
AWS Mobile Hub
Azure App Service
Motivation

- What kind of code does a web application backend developer typically write?
  - Game site
  - Social media site
  - Messaging app
Backend-as-a-Service

- Turnkey back-end infrastructure service
  - Provide common backend functions to the application
    - Authentication (email/password, OAuth)
    - Real-time database
    - Browser notifications
    - Messaging
    - Advertising
    - Analytics
    - REST API endpoints for app functions
    - DNS and HTTPS certificate management
    - CDN management
    - “All the stuff you need in your app, but don’t want to code yourself”
  - Typically used for mobile applications to allow developer to focus only on application code (e.g. a game)
Initially

- Programmatic interface to backend, real-time database
  - All YCombinator startup companies in 2011 (mobile apps)

- No need to configure or deploy database
  - Database infrastructure hosted by each company
  - Apps given a programmatic or REST-based API for data access
  - Schemaless, key-value data stores with basic querying (e.g. filters)

- App uses persistent connection to send and receive data updates in real-time (via AJAX/REST)
Parse

- Allows those without a CS degree to create a full-featured mobile application in a short amount of time
  - UI/UX designers can implement entire app
  - Parse maintains backend
Issues

- Vendor lock-in
  - Recall Windows SDK discussion
  - App and data must be migrated if developer wants to switch platforms
- Longevity (i.e. supply-chain stability)
  - What happens if startup dies?
  - No business wants to rely on a startup for a core function
Parse

- Acquired by Facebook in 2013

**Facebook to Buy Mobile Startup Parse in Cash-and-Stock Deal**

*By Evelyn M. Rusli*

Updated April 25, 2013 8:27 p.m. ET

Facebook Inc., which is ramping up its mobile business, has agreed to buy startup Parse in a cash-and-stock deal valued at about $85 million, according to people familiar with the matter.
Those relying on Parse platform forced to migrate or learn how to operate the Parse backend
• Acquired in 2014

• But, same issues
  • Will Google keep it cheap?
  • Will Google ever shut it down?
Google Firebase
Firebase

• Rapid development of iOS, Android, and web applications
  • Node.js, Python, Go and Java libraries for server application to call into
  • Native iOS and Android client libraries for mobile-only or hybrid apps
  • Write code to the APIs and then forget about managing backend hardware and software (abstracted away)
Firebase

- Initial features
  - Real-time NoSQL JSON database via a REST API
    - Each data object given a URL
    - Database updates from client consist of a simple HTTP request
    - Updates automatically pushed to iOS, Android, and web UIs viewing data
  - Authentication out-of-the-box with e-mail, Facebook, Twitter, github, Google
    - OAuth2 a nightmare to get right
    - On Firebase, two steps
      - Find and add your developer API key
      - Add one line of code to your auth page
- Hosting infrastructure
  - Static content delivered via CDN
  - Automatic certificate generation for https
Firebase

- Then added
  - Notifications
  - Analytics (how users use your app via event logging)
  - Advertising network integration (AdWords/AdMob)
    - Likely why Google won’t shut this down?
  - Messaging
  - Remote configuration
    - Controlling client apps (e.g. AB testing to see which version makes more money)
Example

- API access to manage user account creating and sign-in

```javascript
firebase.auth().createUserWithEmailAndPassword(email, password).catch(function(error) {
    // Handle Errors here.
    var errorCode = error.code;
    var errorMessage = error.message;
    // ...
});
```

```javascript
firebase.auth().signInWithEmailAndPassword(email, password).catch(function(error) {
    // Handle Errors here.
    var errorCode = error.code;
    var errorMessage = error.message;
    // ...
});
```
Pricing

- Free-tier
  - Up to 100 concurrent users
  - Up to 50 GB of data in BigQuery (for analytics)
  - Up to 1 TB of querying

- Quick-starts
  - [https://github.com/firebase/quickstart-python](https://github.com/firebase/quickstart-python)
  - [https://github.com/firebase/quickstart-js](https://github.com/firebase/quickstart-js)
Firebase Lab #1

- Firebase Web Codelab (39 min)
- Create project in the Google Firebase Console (different from the Google Cloud Console)

https://console.firebase.google.com/

- Call it firebaselab
Enable use of Google authentication

- From console, Develop=>Authentication->Sign-In Method
- Enable the **Google Sign-in Provider** to allow users of your web app to sign in with their Google Accounts, then Save
Enable use of Cloud Storage

- (Optional) if you want to do entire lab
- Storage > Get Started > Got It
Firebase code to add

- Click on "Add Firebase to your web app"
Firebase code to add

- Keep tab open so code can be copied into app later

```html
<script src="https://www.gstatic.com/firebasejs/5.2.1/firebase.js"></script>

// Initialize Firebase
var config = {
  apiKey: "[API_KEY]",
  authDomain: "fir-1-b024d.firebaseapp.com",
  databaseURL: "https://fir-1-b024d.firebaseio.com",
  projectId: "fir-1-b024d",
  storageBucket: "fir-1-b024d.appspot.com",
  messagingSenderId: "6753650673"
};
firebase.initializeApp(config);
</script>
```
Setup code

- Goto console.cloud.google.com to find the project created (firebaselab)
- Launch Cloud Shell
  - Clone repository
    ```
    git clone https://github.com/firebase/friendlychat-web
    ```
- Use **npm** to install the Firebase CLI.
  - In Cloud Shell
    ```
    cd friendlychat-web/web-start
    npm -g install firebase-tools
    ```
- To verify that the CLI has been installed correctly, run
  ```
  firebase --version
  ```
Install the Firebase CLI

- Authorize the Firebase CLI to deploy app by running:

  `firebase login --no-localhost`

- Visit URL given and login to your pdx.edu account

- Get authorization code and paste it in to complete login

  Please copy this code, switch to your api:

  4/AADaqkGDYgv5ufFn0S_a3P3_3wV

  Paste authorization code here: 4/AACbxpOJTWywwV87nQO6m6t-465w

  Success! Logged in as wuchang@pdx.edu
  wuchangfeng@cs410c-wuchang-201515:/frien
Setup Firebase for app

- Make sure you are in the `web-start` directory then set up Firebase to use your project
  
  ```
  firebase use --add
  ```

- Use arrow keys to select your Project ID and follow the instructions given.

```
wuchangfeng@cs410c-wuchang-201515:~/friendlychat-web/web-start$ firebase use --add
? Which project do you want to add? cs410c-wuchang-201515
? What alias do you want to use for this project? (e.g. staging) firebase
ebaselab
```

Created alias `firebasebaselab` for cs410c-wuchang-201515.
Now using alias `firebasebaselab` (cs410c-wuchang-201515)
```
Include Firebase code to main page

- At bottom of `index.html` before script inclusion
- Note: Comment out the other two JS files

```html
<!-- Import and configure the Firebase SDK -->
<!-- These scripts are made available when the app is served or deployed on Firebase -->
<script src="https://www.gstatic.com/firebasejs/5.0.4/firebase.js"></script>
<script>
  // Initialize Firebase
  var config = {
    apiKey: "YOUR_API_KEY",
    authDomain: "fir-1-b024d.firebaseapp.com",
    databaseURL: "https://fir-1-b024d.firebaseio.com",
    projectId: "fir-1-b024d",
    storageBucket: "fir-1-b024d.appspot.com",
    messagingSenderId: "6753650673"
  };
  firebase.initializeApp(config);
</script>
<!-- <script src="/__/firebase/4.1.3/firebase.js"></script> -->
<!-- <script src="/__/firebase/init.js"></script> -->
<script src="scripts/main.js"></script>
</body>
</html>
```
Install the Firebase CLI

- Run the app on the development server
  
  ```
  firebase serve
  ```

- Click on link or go to Web Preview, change port to 5000, and preview
Part 1: Firebase Authentication

- In `scripts/main.js` modify `FriendlyChat.prototype.initFirebase` function to set shortcuts to Firebase SDK features and initiate auth (Line 57)

```javascript
// Sets up shortcuts to Firebase features and initiate firebase auth.
FriendlyChat.prototype.initFirebase = function() {
  // Shortcuts to Firebase SDK features.
  this.auth = firebase.auth();
  this.database = firebase.database();
  this.storage = firebase.storage();
  // Initiates Firebase auth and listen to auth state changes.
  this.auth.onAuthStateChanged(this.onAuthStateChanged.bind(this));
};
```
• Change the `FriendlyChat.prototype.signIn` function to set Google as identity (OAuth) provider (Near Line 117)

```javascript
// Signs-in Friendly Chat.
FriendlyChat.prototype.signIn = function() {
    // Sign in using Firebase with popup auth and
    // Google as the identity provider.
    var provider = new firebase.auth.GoogleAuthProvider();
    this.auth.signInWithPopup(provider);
};
```

• Also set the `FriendlyChat.prototype.signOut` function just below

```javascript
// Signs-out of Friendly Chat.
FriendlyChat.prototype.signOut = function() {
    // Sign out of Firebase.
    this.auth.signOut();
};
```
• **FriendlyChat.prototype.onAuthStateChanged** function triggers when the auth state changes.
  • Change the two lines with a TODO to read the user's profile pic and name from OAuth provider (Google)

```javascript
// Triggers when the auth state change for instance when the user
// signs-in or signs-out.
FriendlyChat.prototype.onAuthStateChange = function(user) {
  if (user) { // User is signed in!
    // Get profile pic and user's name from the Firebase user object.
    var profilePicUrl = user.photoURL; // Only change these two lines!
    var userName = user.displayName;    // Only change these two lines!
    ...
  }
}
```
To detect if the user is signed-in add these few lines to the top of the
`FriendlyChat.prototype.checkSignedInWithMessage` function where the `TODO` is located:

```javascript
// Returns true if user is signed-in. Otherwise false and
// displays a message.
FriendlyChat.prototype.checkSignedInWithMessage = function() {
  // Return true if the user is signed in Firebase
  if (this.auth.currentUser) {
    return true;
  }
  ...
```
If you want to test with the development server (i.e. via `firebase serve`), you will need to authorize its appspot domain it is served from.

- Firebase=>Authentication=>Sign-in Method=>Authorized Domains

The domain used on a `firebase deploy` is enabled by default (`$PROJECT_ID.firebaseapp.com`)
• Ensure that third-party cookies are enabled
• In Chrome=>Settings=>Advanced=>Privacy and Security=>Content Settings=>Cookies=>Block Third Party Cookies
Test Signing-In to the App

- Update app

  ```javascript
  firebase serve
  ```

- Click on link or go to Web Preview and change to 5000

- Sign-In with Google

- Show that the Google profile pic and name of the user is displayed
Part 2: Implement reading messages

- Create a JSON file called initial_messages.json on your local machine
- Paste the FriendlyChat messages below into it (for import)
- Note that the messages are also included in the initial_messages.json file at repository root

```json
{
    "messages": {
        "-K2ib4H77rj0LYewF7dP": {
            "text": "Hello",
            "name": "anonymous"
        },
        "-K2ib5JHRbbL0NrztUfO": {
            "text": "How are you",
            "name": "anonymous"
        },
        "-K2ib62mjHh34CAUbide": {
            "text": "I am fine",
            "name": "anonymous"
        }
    }
}
```
Import messages into database

- Firebase=>Database
  - Overflow drop-down on the far right, select *Import JSON*.
  - Browse to `initial_messages.json` file created.
  - Select it then click *Import*
    - This will replace any data currently in your database.
fir-1-b024d

messages

- K2ib4H77rj0LYewF7dP
  - name: "anonymous"  
    - text: "Hello"

- K2ib5JHRbbL0NrzTUfO
  - name: "anonymous"
    - text: "How are you"

- K2ib62mjHh34CAUbide
  - name: "anonymous"
    - text: "I am fine"
Alternative way of setting messages

- In web-start directory in Cloud Shell, you can import the initial_messages.json file at repository root directly via

  ```bash
  firebase database:set / ../initial_messages.json
  ```

- Try editing the file and running the command to see the database messages set
Synchronizing Messages

- Modify `FriendlyChat.prototype.loadMessages` function in `main.js`
  - Synchronize messages on the app across clients
  - Add listeners that trigger when changes are made to data
    - Listeners update UI element for showing messages.
  - Only display the last 12 messages of the chat for fast load

```javascript
// Loads chat messages history and listens for upcoming ones.
FriendlyChat.prototype.loadMessages = function() {
    // Reference to the /messages/ database path.
    this.messagesRef = this.database.ref('messages');
    // Make sure we remove all previous listeners.
    this.messagesRef.off();
    // Loads the last 12 messages and listen for new ones.
    var setMessage = function(data) {
        var val = data.val();
        this.displayMessage(data.key, val.name, val.text, val.photoUrl, val.imageUrl); }.bind(this);
    this.messagesRef.limitToLast(12).on('child_added', setMessage);
    this.messagesRef.limitToLast(12).on('child_changed', setMessage);
};
```
Test message import

- Run firebase serve again to update your app
- Sample messages imported earlier should be displayed in the Friendly-Chat UI
Test real-time database updates

- Go back to Firebase Database web UI to view messages in database
- We will manually add a message and it will update the UI in real-time automatically
  - Click + to add child to messages
- Click on + to add a child to the initial Name:Value entry
Test real-time database updates

- Then, in under the parent, click + again to add a second child (Do not embed second child under the first)

- Set the name of the Key in first row as well as the Name:Value entry for the name of the user and the text they sent. As soon as you "Add", switch quickly back to running app to see update
Hello
anonymous

How are you
anonymous

I am fine
anonymous

Real-time update
Wu From Console

Message...
Part 3: Implement message sending

- Update `FriendlyChat.prototype.saveMessage`

```javascript
// Saves a new message on the Firebase DB.
FriendlyChat.prototype.saveMessage = function(e) {
  e.preventDefault();

  // Check that the user entered a message and is signed in.
  if (this.messageInput.value && this.checkSignedInWithMessage()) {
    var currentUser = this.auth.currentUser;
    // Add a new message entry to the Firebase Database.
    this.messagesRef.push({
      name: currentUser.displayName,
      text: this.messageInput.value,
      photoUrl: currentUser.photoURL || '/images/profile_placeholder.png'
    }).then(function() {
      // Clear message text field and SEND button state.
      FriendlyChat.resetMaterialTextfield(this.messageInput);
      this.toggleButton();
    }.bind(this)).catch(function(error) {
      console.error('Error writing message to Firebase Database', error);
    });
  }
};
```
Test message sending

- Update your app
  `firebase serve`

- Sign-in to Google if necessary

- Click on Message box, type a message and click Send
Test message sending

- Message will be inserted into real-time database
- UI will automatically update with message and the account profile picture
- Note
  - One can mock up an iOS or Android client version to interoperate (see the two other codelabs)
Firebase Lab #1

- Skip steps 9, 10, 11, and 12 (unless you want to try it)
- Deploy app
  ```
  firebase deploy
  ```
- Send URL to partner or instructor so they can add messages via their Google account
- Show a screenshot of messages sent by multiple users
- [https://codelabs.developers.google.com/codelabs/firebase-web](https://codelabs.developers.google.com/codelabs/firebase-web) (39 min)
Extra
Implement Sending Images

• We will add code that:

  • Creates a "placeholder" chat message with a temporary loading image into the chat feed.

  • Upload the file to Cloud Storage to the path: /<uid>/<postId>/<file_name>

  • Update the chat message with the newly uploaded file's Cloud Storage URI in lieu of the temporary loading image.
Implement Sending Images

- In `FriendlyChat.prototype.saveImageMessage` find block that checks if the user is signed-in.

// Saves message containing image URI in Firebase via Cloud Storage

```javascript
FriendlyChat.prototype.saveImageMessage = function(event) {

  // Replace the code within block that checks if the user is signed-in
  if (this.checkSignedInWithMessage()) {
    // TODO(DEVELOPER): Upload image to Firebase storage and add message
  }
```

- Replace inner block where the `TODO` is located with
Implement Sending Images

```javascript
if (this.checkSignedInWithMessage()) {
    // Add message with loading icon, update with shared image when done
    var currentUser = this.auth.currentUser;
    this.messagesRef.push({
        name: currentUser.displayName,
        imageUrl: FriendlyChat.LOADING_IMAGE_URL,
        photoUrl: currentUser.photoURL || `/images/profile_placeholder.png`
    }).then(function(data) {
        // Upload the image to Cloud Storage.
        var filePath = currentUser.uid + '/' + data.key + '/' + file.name;
        return this.storage.ref(filePath).put(file).then(function(snapshot) {
            // Get file's Storage URI and update chat message placeholder.
            var fullPath = snapshot.metadata.fullPath;
            return data.update({imageUrl:
                this.storage.ref(fullPath).toString()})
            }.bind(this));
    }.bind(this)).catch(function(error) {
        console.error('Error uploading a file to Cloud Storage:', error);
    });
}
```
Displaying Images

- Chat messages with images include Cloud Storage references where they are located
- Done in the format below

  gs://<bucket>/<uid>/<postId>/<file_name>
Displaying Images

• Modify `FriendlyChat.prototype.setImageUrl`

```javascript
// Sets URL of given img element with location in Cloud Storage.
FriendlyChat.prototype.setImageUrl = function(imageUri, imgElement) {

};
```

• to query Cloud Storage based on URI via this code

```javascript
// If the image is a Cloud Storage URI we fetch the URL.
if (imageUri.startsWith('gs://')) {
    imgElement.src = FriendlyChat.LOADING_IMAGE_URL;
    // Display a loading image first.
    this.storage.refFromURL(imageUri).getMetadata().then(function(metadata) {
        imgElement.src = metadata.downloadURLs[0];
    });
} else {
    imgElement.src = imageUri;
}
```
Displaying Images

- Restart app again via firebase serve, then add an image