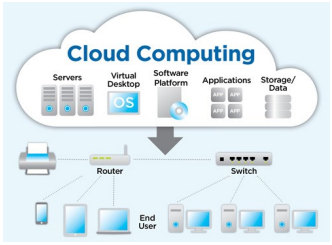
MOBILE COMPUTING

CSE 40814/60814
Spring 2021

Cloud Computing

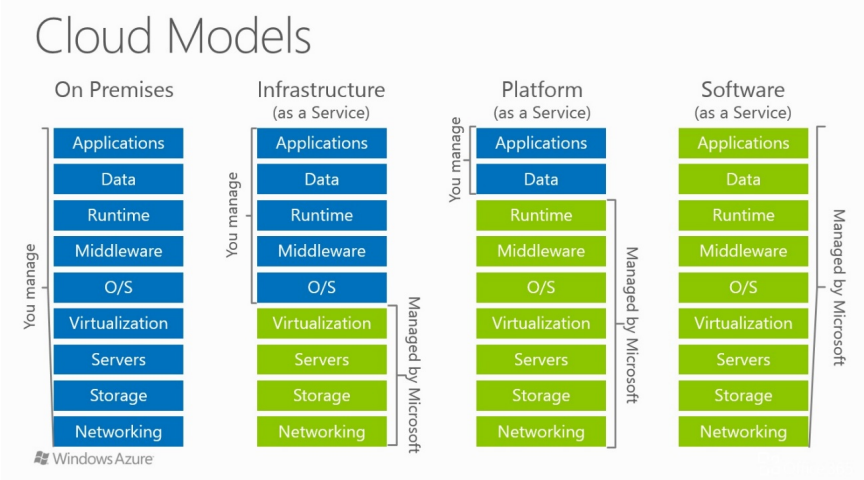


- Delivery of computing **services**
  - servers
  - storage
  - analytics
  - databases
  - networking
  - and much more...
- Another definition: network-based computing taking place over the Internet, while hiding complexity of underlying infrastructure using simple APIs

Cloud Computing

- Collection/group of **integrated and networked hardware, software, and Internet infrastructure** (called a platform)
- Platforms provide **on demand** services that are **always on** and **accessible anytime** and **anywhere**
- **Advantages:** new applications; anytime/anywhere access; homogeneity; virtualization; resilient; cost; sharing; collaboration; maintenance; security; …

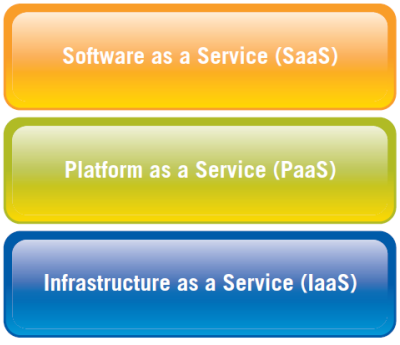
Cloud Models



Definitions

- **Virtualization:** creation of a virtual resource such as a server, desktop, operating system, file, storage, or network
- **Middleware:** software that acts as a bridge between an operating system or database and applications, especially on a network
- **Runtime:** software designed to support the execution of computer programs

IaaS, PaaS, SaaS



Enduser application is delivered as a service. Platform and infrastructure is abstracted, and can deployed and managed with less effort.

Application platform onto which custom applications and services can be deployed. Can be built and deployed more inexpensively, although services need to be supported and managed.

Physical infrastructure is abstracted to provide computing, storage, and networking as a service, avoiding the expense and need for dedicated systems.

- Simple example:
- IaaS: barebones computer
- PaaS: computer + OS (incl. development environment)
- SaaS: complete solution including application(s)

Cloud Example: S3

- Amazon Simple Storage Service (S3)
- Unlimited storage
- Pay for what you use

	S3 Standard	S3 Standard – Infrequent Access	AWS Glacier
STORAGE			
First 50 TB/ month	$0.023 / GB	$0.0125 / GB	$0.004 / GB
Next 450 TB/ month	$0.022 / GB	$0.0125 / GB	$0.004 / GB
Over 500 TB/ month	$0.021 / GB	$0.0125 / GB	$0.004 / GB
REQUESTS			
PUT, COPY, POST, or LIST	$0.005 / 1,000 requests	$0.01 / 1,000 requests	
GET and all other requests	$0.004 / 10,000 requests	$0.01 / 10,000 requests	
Delete requests	Free	Free	Free, but with limits and potential surcharges
Lifecycle Transition Requests into S3 Standard IA		$0.01 / 1,000 requests	
Glacier archive and restore requests			$0.05 / 1,000 requests, see Glacier pricing for more details on retrieval fees

Cloud Example: EC2

- Amazon Elastic Compute Cloud (EC2)
  - Virtual computing environments (“instances”)
  - Pre-configured templates for instances
  - Launch as many virtual servers as needed (“elastic”)
  - Xen and KVM hypervisor

Other Examples

Mobile Apps & Cloud Computing

- "Information at your fingertips anywhere and anytime"
- **Mobile Cloud Computing (MCC)**
  - **Infrastructure where both the data storage and data processing happen outside of the mobile device**

Mobile Cloud Computing – Why?

- Limited mobile resources
  - Battery, storage, processing, network, …
  - Consider tradeoffs!

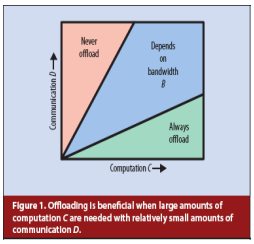


Figure 1. Offloading is beneficial when large amounts of computation C are needed with relatively small amounts of communication D.

- Permanent storage
  - Backup (reliability), long-term storage, anywhere access (availability)

- Data sharing
  - Social media, sensor data, collaboration, producer-consumer, …

Mobile Cloud Computing

- Mobile cloud applications move the computing power and data storage away from the mobile devices and into powerful and centralized computing platforms located in clouds, which are then accessed over the wireless connection based on a **thin native client**

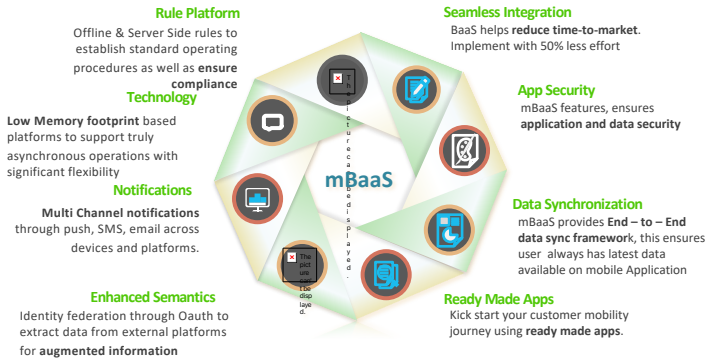
MOBILE CLOUD COMPUTING =
MOBILE COMPUTING + CLOUD COMPUTING

"Backend"

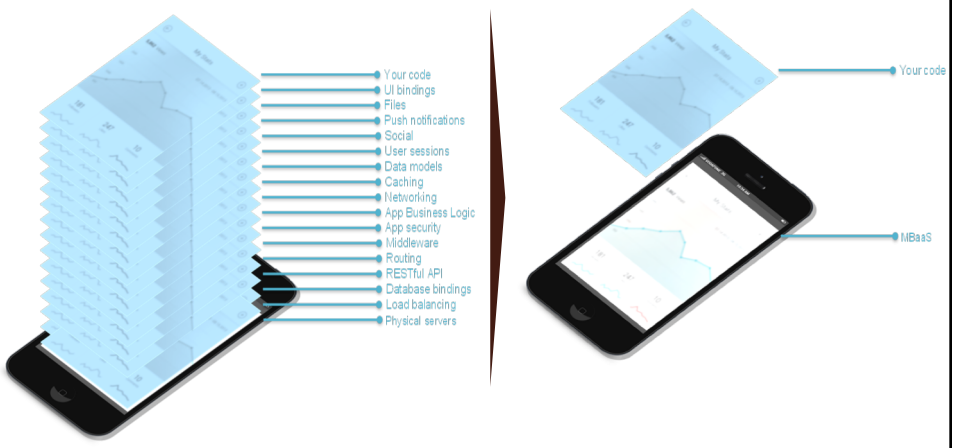
- No clear definition of backend
- Frontend – mobile device user interface
- Backend - data, server, programs the user does not interact with directly
- **Backend as a Service (Baas)**
- Sometimes MBaaS (M for mobile)

Definition of MBaaS

- Provide **web and mobile app** developers with a way to **connec**t their applications to **backend cloud storage and processing** while also providing **common features** such as user management, push notifications, social networking integration, and other features that **mobile users demand** from their apps these days



MBaaS – Focus on Frontend!



Advantages for Developer

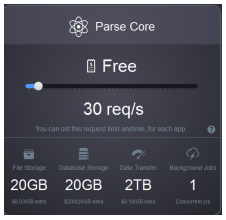
- **Efficiency gains**: reducing the development cost, development time and maintenance cost
- **Faster Time To Market:** reduce obstacles from idea to production and operations overhead.
- **Optimized for Mobile and Tablets:** optimization of data and network for mobile apps, and lower fragmentation problems across multiple platforms and devices.
- **Secure & Scalable:** bundled infrastructure that deals with scalability, security, performance and other operational headaches.
- **Handle App Growth & Maintenance:** brings common and essential 3rd party API resources into a single stack, preventing developers from having to go gather them separately.

MBaaS Providers

kinvey
FEEDHENRY
amazon web services
anypresence
appcelerator
Google Cloud Platform Live
built.io
apigee
APIOMAT Backend as a Service
back{4}app
heroku
Sencha
TruJunction
Firebase

Example: back4app (parse.com)

- Founded in 2011
- Acquired by Facebook in 2013
- More than 500,000 apps in 2014
- Shut down January 2017
- Made code open-source
- Many replacements, including self-hosted
- Back4app.com



Free

High Limits for Devs.

1 GB Database Storage **5 GB** File Storage **10** Requests/second
50 K Requests/month **1** Cloud Code Job

SIGN UP NOW

Back4app

- www.back4app.com
- iOS: http://docs.back4app.com/docs/ios/
- Android: http://docs.back4app.com/docs/android/
- Cloud Code: http://docs.back4app.com/docs/integrations/

Cloud Code (main.js)

```
Parse.Cloud.define("notify", function (request, response) {
        Parse.Push.send({
                channels: request.params.channels,
                data: {
                        alert: request.params.message
                }
        }, {
                success: function ({
                        response.success('Success! ' +
                        request.params.message);
                },
                error: function (error) {
                        response.error('Error! ' +
                        error.message);
                },
                useMasterKey: true,
        });
});
```

Cloud Code (main.js)

```
Parse.Cloud.define("notify2", function (request, response) {
        var pushQuery = new Parse.Query(Parse.Installation);
        pushQuery.equalTo('user', request.params.receiver);
        Parse.Push.send({
                where: pushQuery,
                data: {
                        alert: request.params.message
                }
        }, {
                success: function () {
                        response.success('Success! ' +
                                request.params.message);
                },
                error: function (error) {
                        response.error('Error! ' + error.message);
                },
                useMasterKey: true,
        });
});
```

iOS Code

```
[PFCloud callFunctionInBackground:@"notify"
withParameters:@{
                @"channels": @[username],
                @"message": message
        }
        block:^(NSString *success, NSError *error)  {
                        if (!error)  {

NSLog(@"Success (%@)!", success);
                        } else {

NSLog(@"Fail!");
                                                        }
                }
];
```

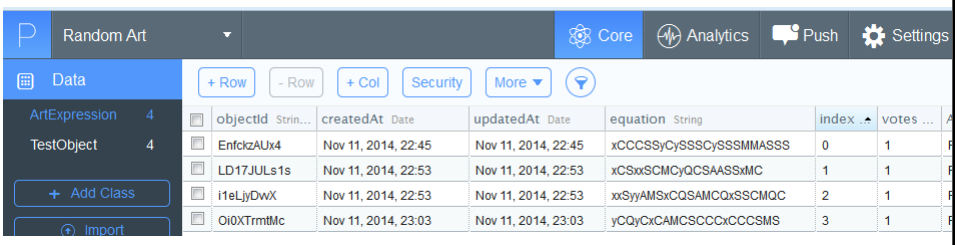
iOS Code

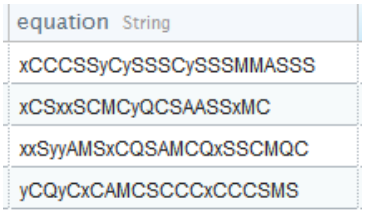
```
[PFCloud callFunctionInBackground:@"notify2"
withParameters:@{
                    @"receiver": someUser.objectId,
                    @"message": message
          }
          block:^(NSString *success, NSError *error) {
                              if (!error) {

NSLog(@”Success (%@)!”, success);
                              } else {
                                 NSLog(@”Fail!”);
                                                    }
                    }
];
```

Configure Mobile App

- iOS: http://docs.back4app.com/docs/ios/quickstart/
- Android: http://docs.back4app.com/docs/android/how-to-build-an-android-app-on-back4app/

Parse/Back4app Dashboard





More Parse/Back4app

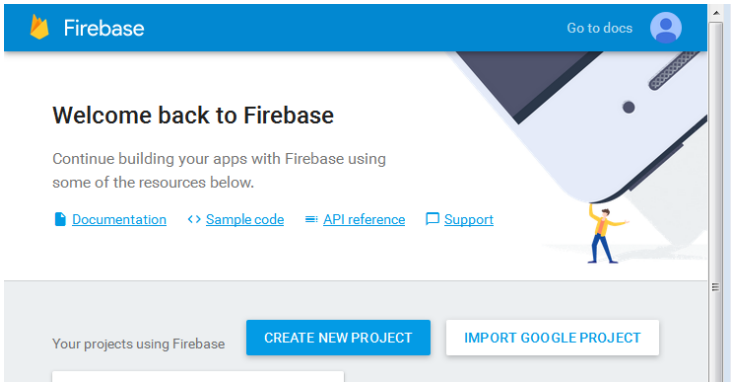
- Includes capability to do local data store
  - Save objects on device, save to cloud later
  - Abstracts away a lot of the details
  - Store/fetch objects in background if possible:
    - Make sure to keep app consistent
    - Provide progress feedback for user
- Parse objects are meant to be "small"
  - Less than 128Kb
  - Parse files for large pieces of data

Firebase

- Yet another Backend as a Service (BaaS):
  - **Firebase** is a real-time cloud data service. Firebase database is stored as JSON and synchronized in real time to every connected client. When you build cross-platform apps with our Android, iOS, and JavaScript SDKs, all of your clients share one Firebase database and **automatically receive updates with the newest data**
- Designed for web and mobile
- Founded in 2011
- Initial product was backend so websites could easily host chat as part of site
- Discovered developers were sending non chat data (such as game state) via the tool

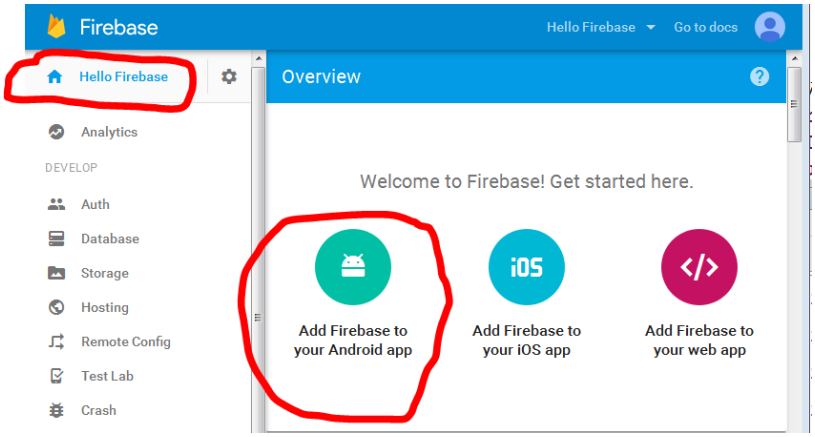
Firebase

- Create Firebase project in console
- Just needs name and country

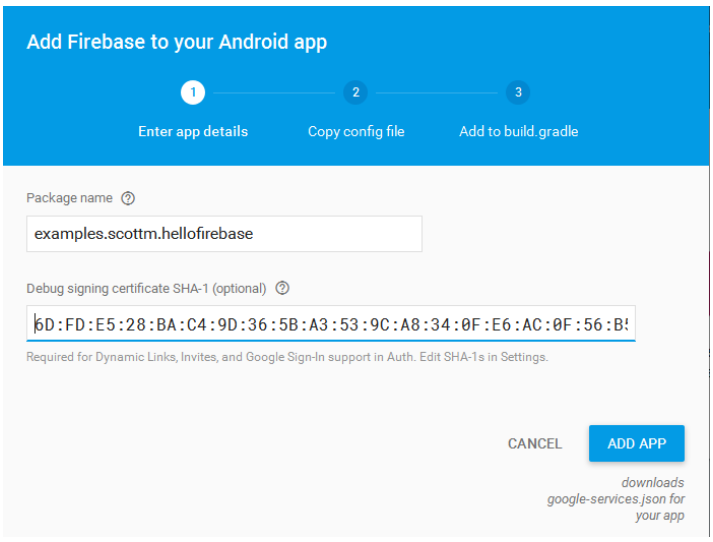


Firebase

- After creating project, overview page:

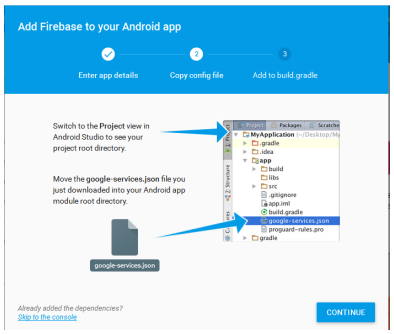


Firebase



Firebase

- After providing package name and SHA-1 fingerprint ...
- Firebase generates a JSON file named google-services.json specific for this project
  - multiple projects / apps -> repeat steps
- Download and add file to project
- Update gradle file



Firebase

- Documentation:
- https://firebase.google.com/docs/?authuser=0

- Capabilities:
  - User authorization
  - Database storage
  - Storage for larger files
  - Cloud messaging
  - Push notifications
  - Analytics
  - Hosting of web content
  - …

	Firebase	Parse Server
General Purpose	Fast real time updates (Real time BaaS)	Open source
Hosting	Google hosting. Free up to 100 simultaneous connections .	Self hosting and Parse hosting providers. No limits. Supports Local testing and developing
Custom Code	Custom code not supported	Custom code totally supported(Cloud Code)
Database	Supports model observer scheme. Now introduced Firebase Storage to upload and download files securely	Has huge relationship based databases
Push	Support Push notifications. Firebase Remote Config to customize apps	Support Push notifications for Android, iOS. Also it is possible to send Push Notifications campaigns.
Setup	Easy setup	Quick setup on Parse Easy step by step set up guide available for migrating from Parse to Parse Server
Storage	Stores data as JSON and data backup can be uploaded to Amazon S3 bucket or Google cloud storage	No restricted time limits and No file storage restrictions. Control over backup, restore, database indexes.
Provider	Developed by Google	Developed by Facebook
Ideal for	Suitable for time applications	Suitable for building general purpose applications

Alternatives

- https://github.com/relatedcode/ParseAlternatives

Choosing a MBaaS

- Ease of Use (Parse, Back4app)
- Automated updates (Firebase)
- Analytics (User data, Crashes)
- Authentication (including social media integration)
- App/Database management
- Push notifications
- Cloud code, background jobs
- System emails (password reset, verification)
- Variety of APIs (iOS, Android, REST)

Firebase & Flutter

- https://firebase.flutter.dev/docs/overview/
- https://codelabs.developers.google.com/codelabs/flutter-firebase#0
- https://firebase.flutter.dev/docs/firestore/usage/

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

- **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.

```
child: StreamBuilder<QuerySnapshot>(
    stream: FirebaseFirestore.instance
        .collection("users")
        .where("facility",
            isEqualTo: widget.userDetail.data()["facility"])
        .snapshots(),
    builder: (context, snapshot) {
      return !snapshot.hasData
          ? Center(child: CircularProgressIndicator())
          : ListView.builder(
              itemCount: snapshot.data.docs.length,
              scrollDirection: Axis.vertical,
              itemBuilder: (context, index) {
                DocumentSnapshot data = snapshot.data.docs[index];
                isAdmin = data.data()["admin"];
                return Card(
                    child: Container(
                        decoration:
                            new BoxDecoration(color: Colors.white),
                        child: ListTile(
                            title: Text(
                                data.data()["firstName"] +
                                    " " +
                                    data.data()["lastName"] +
                                    "\n(" +
                                    data.data()["email"] +
                                    ")",
                                style: TextStyle(
                                    fontSize: 14.0,
                                    fontWeight: FontWeight.bold)), // TextStyle // Text
                            subtitle: Text(data.data()["role"],
                                style: TextStyle(
                                    fontSize: 14.0,
                                    fontWeight: FontWeight.bold)), // TextStyle // Text
```