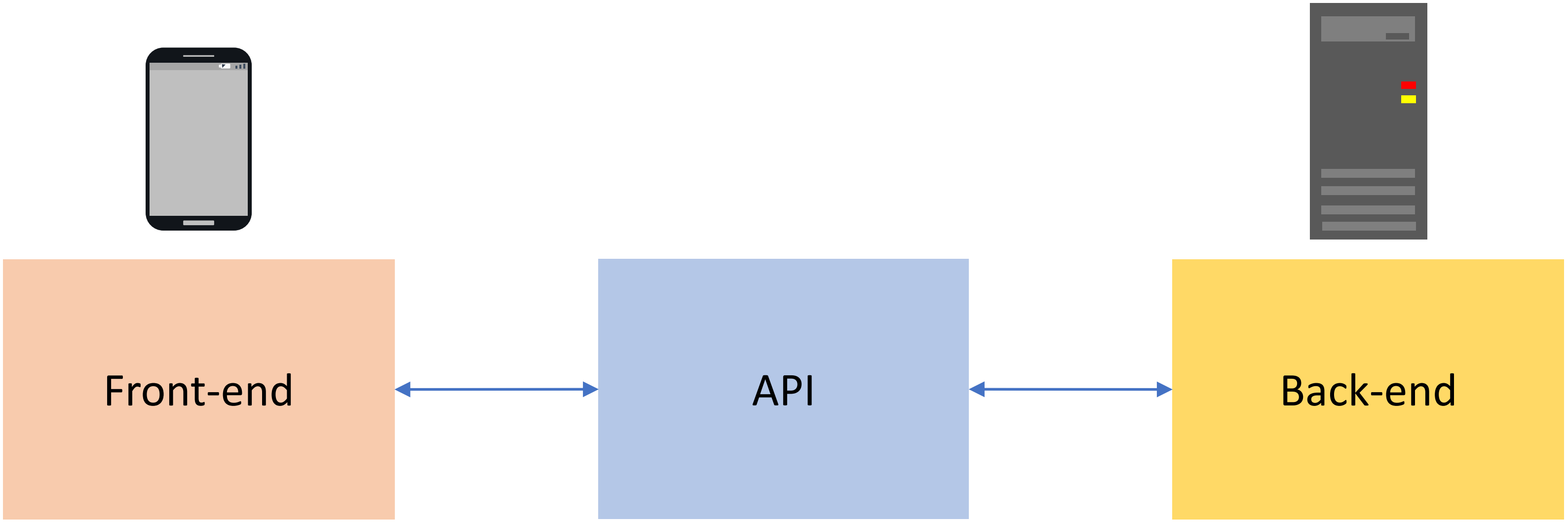
4.2 Network Operation, Background Operation and Mobile Server Options

Objectives

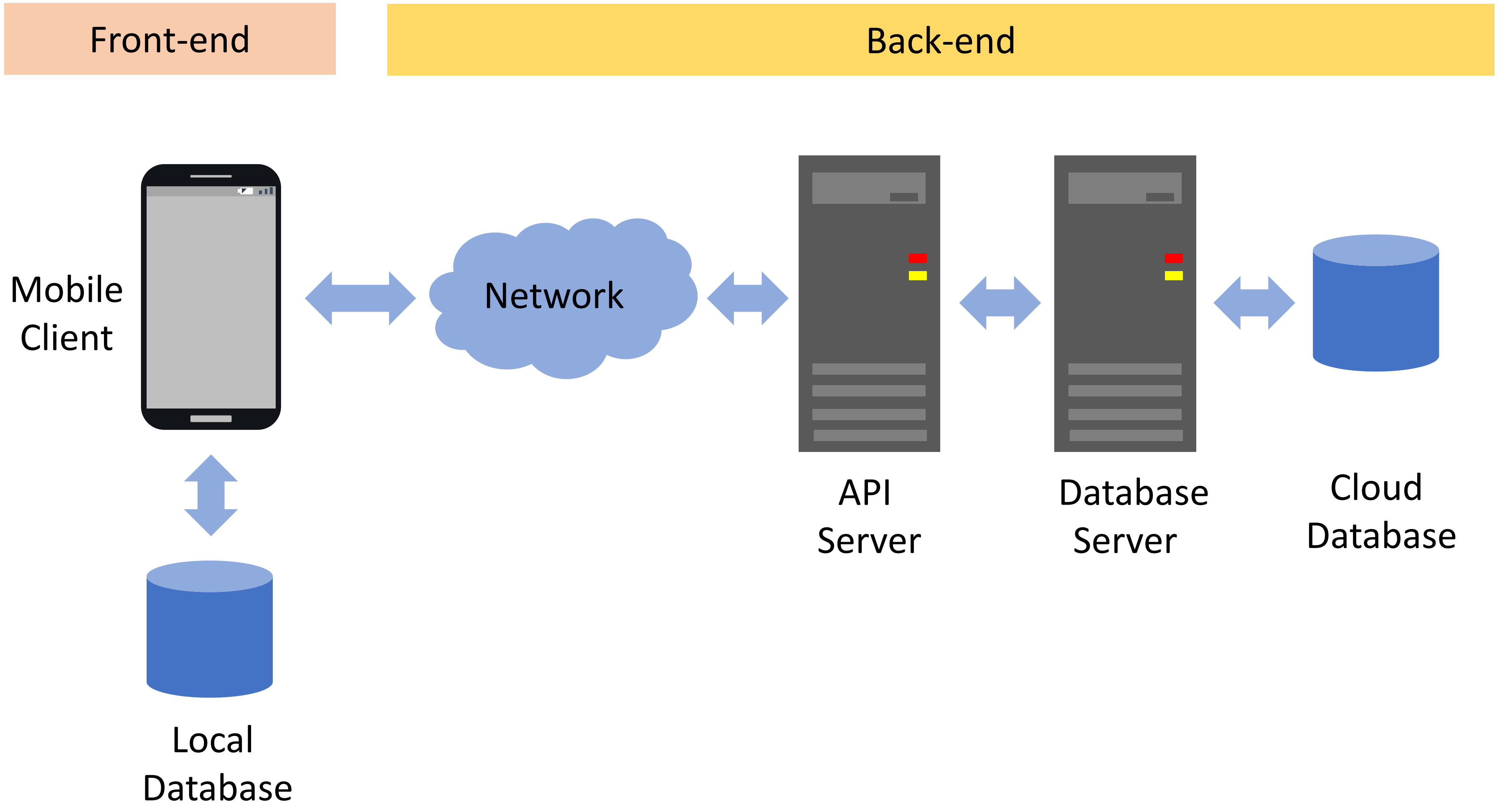
- Understand mobile-to-server communication
- Understand background operations
- Explain network operations
- Explain mobile server options

Mobile-to-server Communication

- Most mobile apps are the front-end interfaces of back-end services
- The two components use Application Programming Interface (API) to communicate with each other



Mobile System Architecture



Front-end

Back-end

Mobile Client

Network

API Server

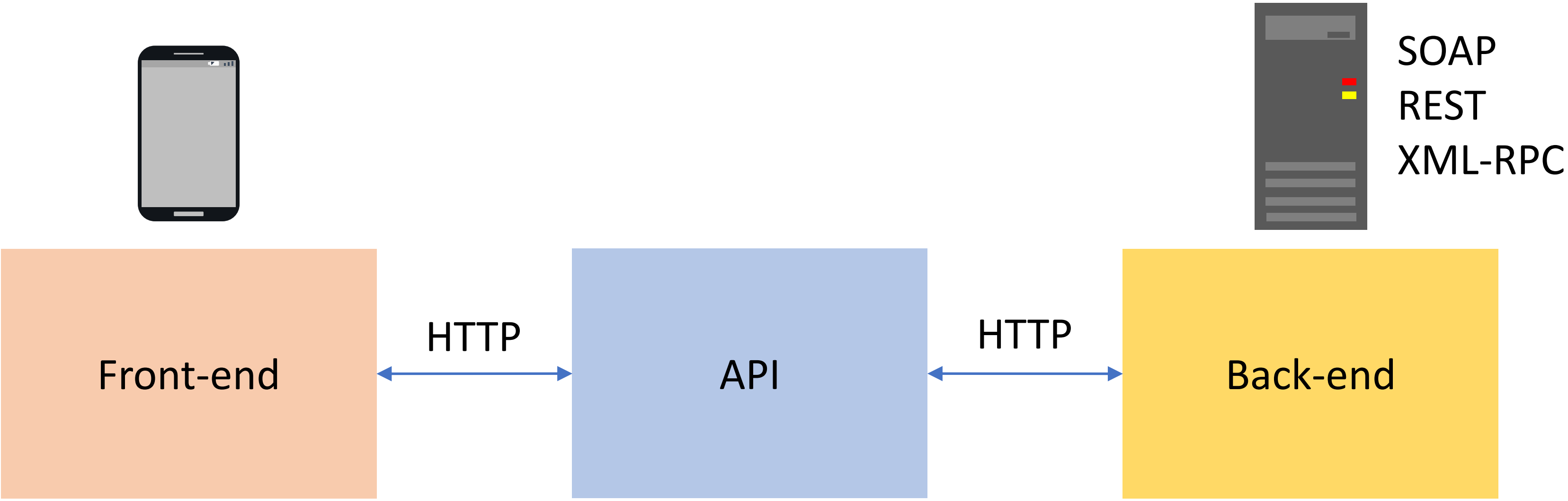
Database Server

Cloud Database

Local Database

Communication Protocol

- HTTP is the most commonly used protocol for communication
- Methods of communication: SOAP, REST, and XML-RPC



Machine-to-machine Communication

- Representational State Transfer (REST)
  - Supports data formats: HTML, XML and JavaScript Object Notation (JSON)
  - Inherits HTTP operations; GET, POST, PUT and DELETE

https://www.restapitutorial.com/lessons/whatisrest.html

Machine-to-machine Communication

- Simple Object Access Protocol (SOAP)
  - Supports data formats: XML
  - Works with application layer protocol. E.g. HTTP, SMTP, TCP, or UDP

https://www.w3.org/TR/2000/NOTE-SOAP-20000508/

REST data formats

- XML

```
<?xml version="1.0" encoding="UTF-8"?>
<authentication-context>
   <username>my_username</username>
   <password>my_password</password>
   <validation-factors>
      <validation-factor>
         <name>remote_address</name>
         <value>127.0.0.1</value>
      </validation-factor>
   </validation-factors>
</authentication-context>
```

- JSON

```
{
   "username" : "my_username",
   "password" : "my_password",
   "validation-factors" : {
      "validationFactors" : [ {
         "name" : "remote_address",
         "value" : "127.0.0.1"
       } ]
    }
}
```

SOAP data formats

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
      <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
        <n:priority>1</n:priority>
        <n:expires>2001-06-22T14:00:00-05:00</n:expires>
      </n:alertcontrol>
  </env:Header>
  <env:Body>
    <m:alert xmlns:m="http://example.org/alert">
      <m:msg>Pick up Mary at school at 2pm</m:msg>
    </m:alert>
  </env:Body>
</env:Envelope>
```

Android JSON Parser

- Most Google API's are available as JSON REST services
- JSON is faster and easier than XML
- Android uses the JSONObject to read JSON streams

https://googleapis.github.io/HowToREST.html#an-api-definition

JSONObject

```
//creating json object
val json_contact:JSONObject = JSONObject(str_response)

//creating json array
var jsonarray_info:JSONArray= json_contact.getJSONArray("info")
var i:Int = 0
var size:Int = jsonarray_info.length()

arrayList_details= ArrayList()

for (i in 0.. size-1) {
    var json_objectdetail:JSONObject=jsonarray_info.getJSONObject(i)
     var user:User= Model()

     user.id=json_objectdetail.getString("id")
     user.name=json_objectdetail.getString("name")
     user.email=json_objectdetail.getString("email")

    arrayList_details.add(user)
}
```

https://www.javatpoint.com/kotlin-android-json-parsing-using-url

Main vs Background Thread

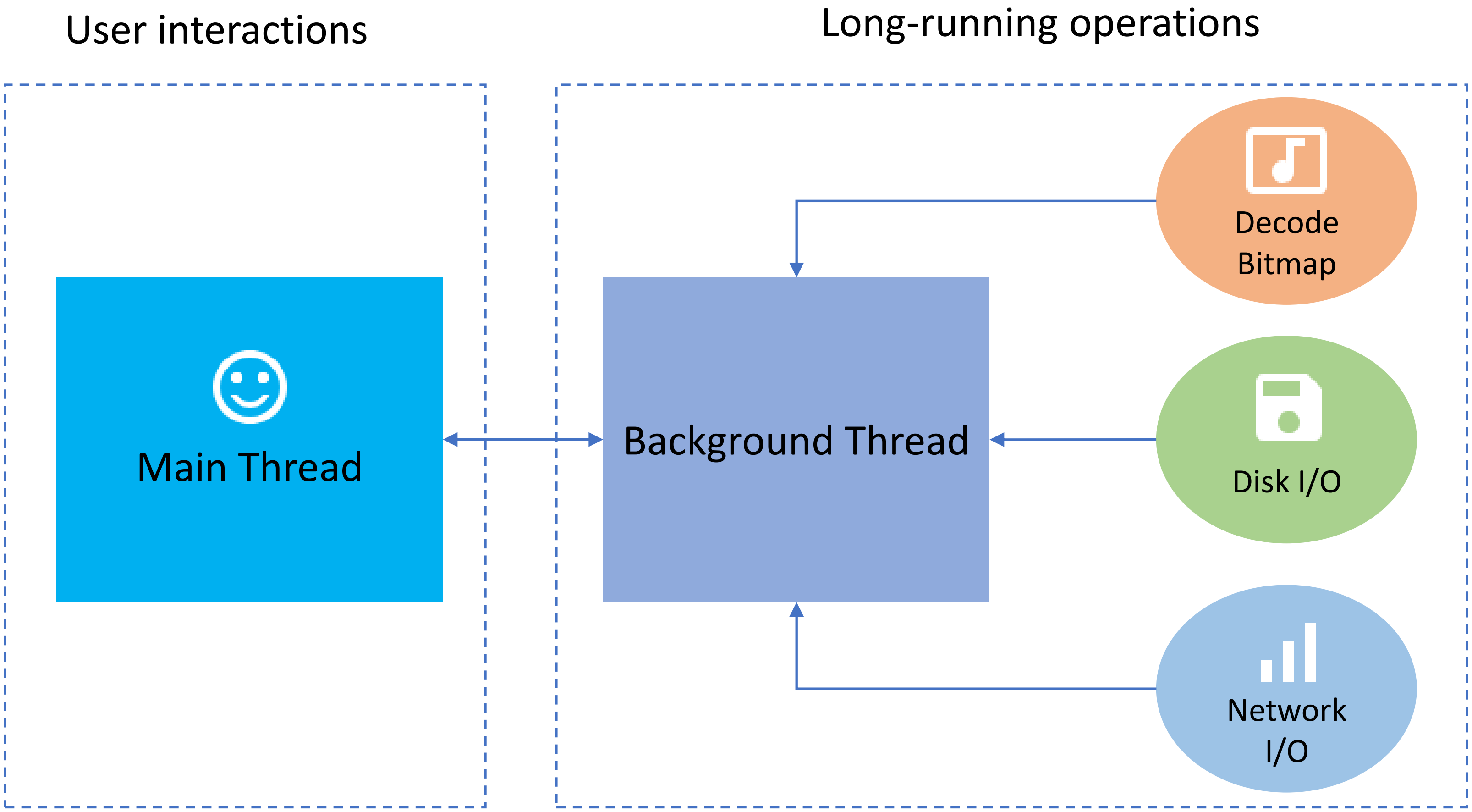
- UI and network operations should be implemented on different thread

- Android throws a NetworkOnMainThreadException if you perform network operations on UI

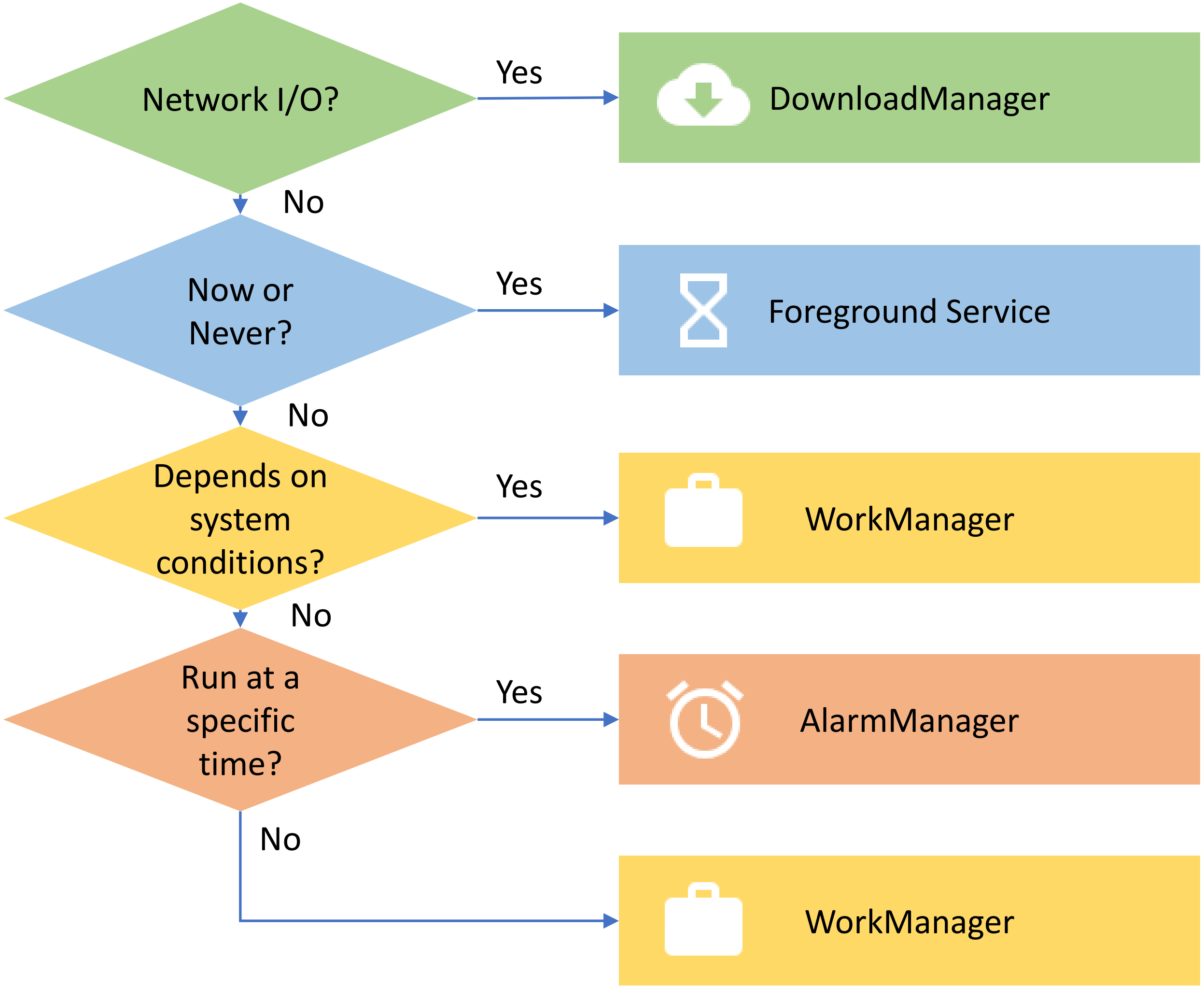
- Two types of thread:
  - Main - UI
  - Background

https://developer.android.com/guide/background/

Main vs Background Thread



Background solutions



Background solutions

Class	Description
DownloadManager	Perform long-running HTTP downloads
Foreground services	User-initiated work that need to run immediately and must execute to completion
AlarmManager	To do the job at the time you specify
WorkManager	Runs deferrable background work when the work's conditions (like network availability and power) are satisfied

Transferring Data via Network

- Java: use AsyncTask or IntentService for real-time data transfer
- Kotlin: use WorkManager - coroutine

Note: Use Sync Adapters for transfer of data regularly and efficiently, but not instantaneously.

https://developer.android.com/training/basics/network-ops

Connecting to the Network

- Permission:

```
<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

- Network communication best practices:
  - Minimize the amount of sensitive data that you transmit over the network
  - Send all network traffic over Secure Socket Layer (SSL)

https://developer.android.com/training/articles/security-ssl.html

Determine Status of Internet Connection

- Use the Connectivity Manager to query the active network and determine if it has Internet connectivity

Kotlin

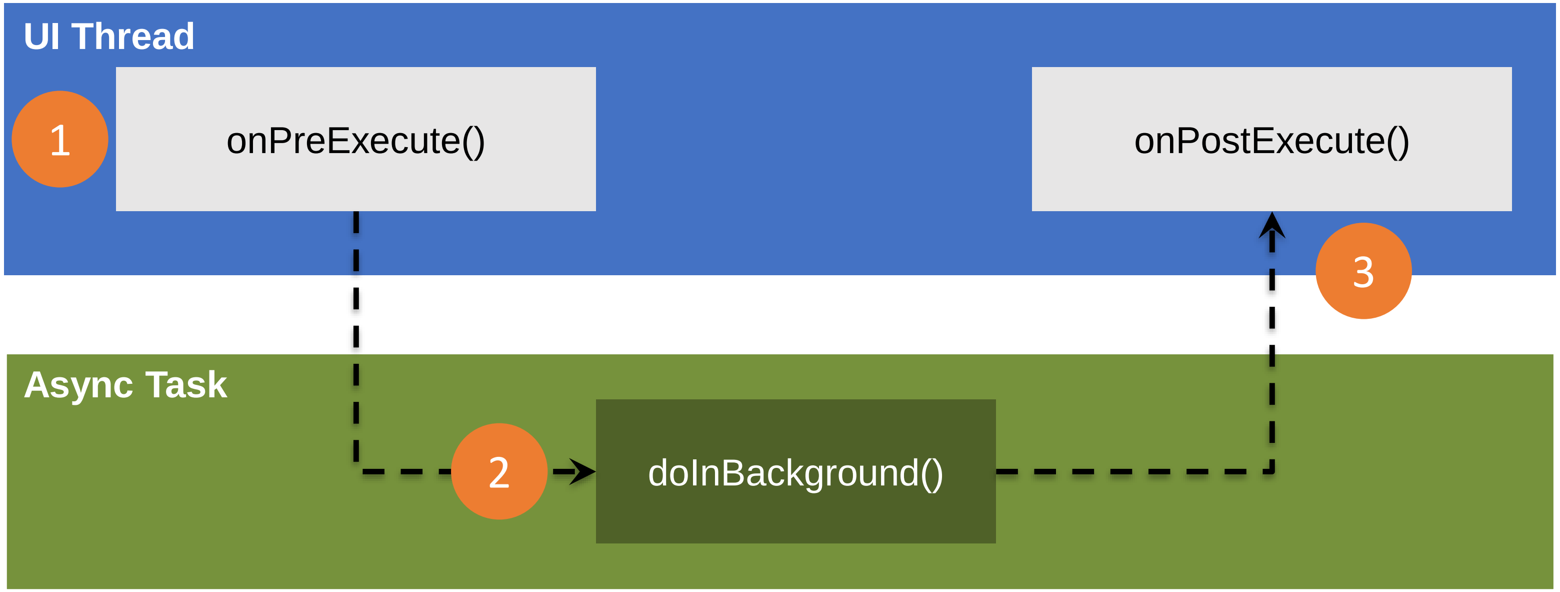
```kotlin
val cm = context.getSystemService(Context.CONNECTIVITY_SERVICE) as ConnectivityManager

val activeNetwork: NetworkInfo? = cm.activeNetworkInfo

val isConnected: Boolean = activeNetwork?.isConnectedOrConnecting == true

// Determine the type of Internet connection currently available
val isWiFi: Boolean = activeNetwork?.type == ConnectivityManager.TYPE_WIFI
```

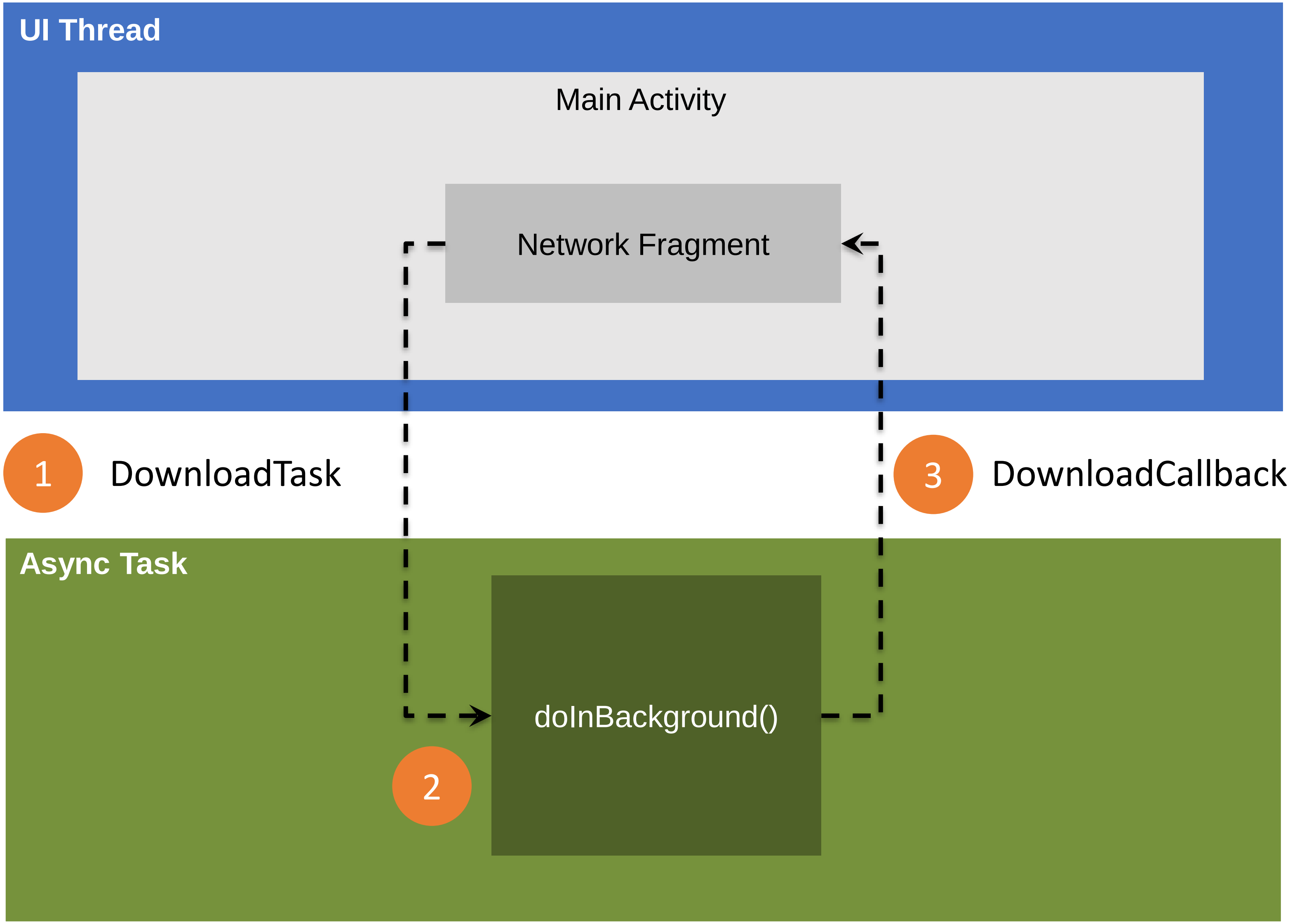
Network Connection

- Network operations can involve unpredictable delays
- Perform network operations on a separate thread from the UI
- Use Async Task to perform network operation

Async Task

- It performs background operations and publish results on UI thread




UI Thread
Main Activity
Network Fragment
1
DownloadTask
3
DownloadCallback
Async Task
doInBackground()
2

AsyncTask

- It must be a private subclass.

- Override at least one method:
  - –doInBackground(Params…)

- Often will override onPostExecute(Result)

- Suitable for short operations

Convert InputStream to a String

- InputSteam is a readable source of bytes

- You can decode/convert it into a target data type

```
val inputStream: InputStream? = null
...

val bitmap: Bitmap = BitmapFactory.decodeStream(inputStream)

// Download an image file and display it
findViewById<ImageView>(R.id.image_view)?.apply {
    setImageBitmap(bitmap)
}
```

Questions?

1. Explain the relationship of the THREE main components of a mobile solution: mobile app, API, and back-end

2. Why a mobile app should not implement network related tasks on the UI thread?

3. Explain the use of Async Task in network operation.

Find out more

- Perform network operation

  https://developer.android.com/training/basics/network-ops

- Perform network operation using Cronet

  https://developer.android.com/guide/topics/connectivity/cronet

Transmit Data Using Volley

- Volley – HTTP library

- Populates data to UI (main thread), i.e. display search results

- Not suitable for large download or streaming operations

- Supports: string, image and JSON

Benefits of Volley

- Automatic scheduling of network requests

- Multiple concurrent network connections

- Support for request prioritization

- Support cancellation of request

Include Volley

- Add the following dependency to your app's build.gradle file

```
dependencies {
    ...
    implementation 'com.android.volley:volley:1.1.1'
}
```

- Git Repository

  https://github.com/google/volley

Make a Standard Request

- Volley support the following requests:

Request	Description
StringRequest	Retrieves a raw string in response
JsonObjectRequest	Retrieves a JSONObject response
JsonArrayRequest	Retrieves a JSONArray response

StringRequest

```
val textView = findViewById<TextView>(R.id.text)
// ...

// Instantiate the RequestQueue.
val queue = Volley.newRequestQueue(this)
val url = "http://www.google.com"


// Request a string response from the provided URL.
val stringRequest = StringRequest(Request.Method.GET, url,
        Response.Listener<String> { response ->
            // Display the first 500 characters of the response string.
            textView.text = "Response is: ${response.substring(0, 500)}"
        },
        Response.ErrorListener { textView.text = "That didn't work!" })


// Add the request to the RequestQueue.
queue.add(stringRequest)
```

Cancel Request

```
val TAG = "MyTag"
val stringRequest: StringRequest // Assume this exists.
val requestQueue: RequestQueue? // Assume this exists.
...

// Set the tag on the request.
stringRequest.tag = TAG

// Add the request to the RequestQueue.
requestQueue?.add(stringRequest)

...

protected fun onStop() {
    super.onStop()
    requestQueue?.cancelAll(TAG)
}
```

RequestQueue

- A basic RequestQueue needs a network + a cache

- Use the BasicNetwork and DiskBasedCache

```kotlin
// Instantiate the cache
val cache = DiskBasedCache(cacheDir, 1024 * 1024) // 1MB cap

// Set up the network to use HttpURLConnection as the HTTP client.
val network = BasicNetwork(HurlStack())
```

RequestQueue

```
// Instantiate the RequestQueue with the cache and network. Start the queue.
val requestQueue = RequestQueue(cache, network).apply {
    start()
}

val url = "http://www.example.com"

// Formulate the request and handle the response.
val stringRequest = StringRequest(Request.Method.GET, url,
         Response.Listener<String> { response ->
            // Do something with the response
        },
        Response.ErrorListener { error ->
            // Handle error
            textView.text = "ERROR: %s".format(error.toString())
        })

// Add the request to the RequestQueue.
requestQueue.add(stringRequest)
// ...
```

Use a singleton pattern

- If your app makes constant use of the network, setup a single instance of RequestQueue

- Implement a singleton class that encapsulate RequestQueue and Volley functions.

```kotlin
class MySingleton constructor(context: Context) {
     // A static object
    companion object {
        //Writes to this field are immediately made visible to other thread
        @Volatile
        private var INSTANCE: MySingleton? = null

        fun getInstance(context: Context) = INSTANCE ?: synchronized(this) {
                INSTANCE ?: MySingleton(context).also {
                    INSTANCE = it
                }
        }
    }

    val imageLoader: ImageLoader by lazy {
        ImageLoader(requestQueue,
                object : ImageLoader.ImageCache {
                    private val cache = LruCache<String, Bitmap>(20)

                    override fun getBitmap(url: String): Bitmap {
                        return cache.get(url)
                    }

                    override fun putBitmap(url: String, bitmap: Bitmap) {
                        cache.put(url, bitmap)
                    }

                })
    }
    ...
```

```
class MySingleton constructor(context: Context) {

    ...

    val requestQueue: RequestQueue by lazy {
        // applicationContext is key, it keeps you from leaking the
        // Activity or BroadcastReceiver if someone passes one in.
        Volley.newRequestQueue(context.applicationContext)
    }

    fun <T> addToRequestQueue(req: Request<T>) {
        requestQueue.add(req)
    }

} // End of class
```

Examples of performing RequestQueue operations using the singleton class:

```
// Get a RequestQueue
val queue = MySingleton.getInstance(this.applicationContext).requestQueue
...
// Add a request (in this example, called stringRequest) to your RequestQueue.
MySingleton.getInstance(this).addToRequestQueue(stringRequest)
```

JSONRequest

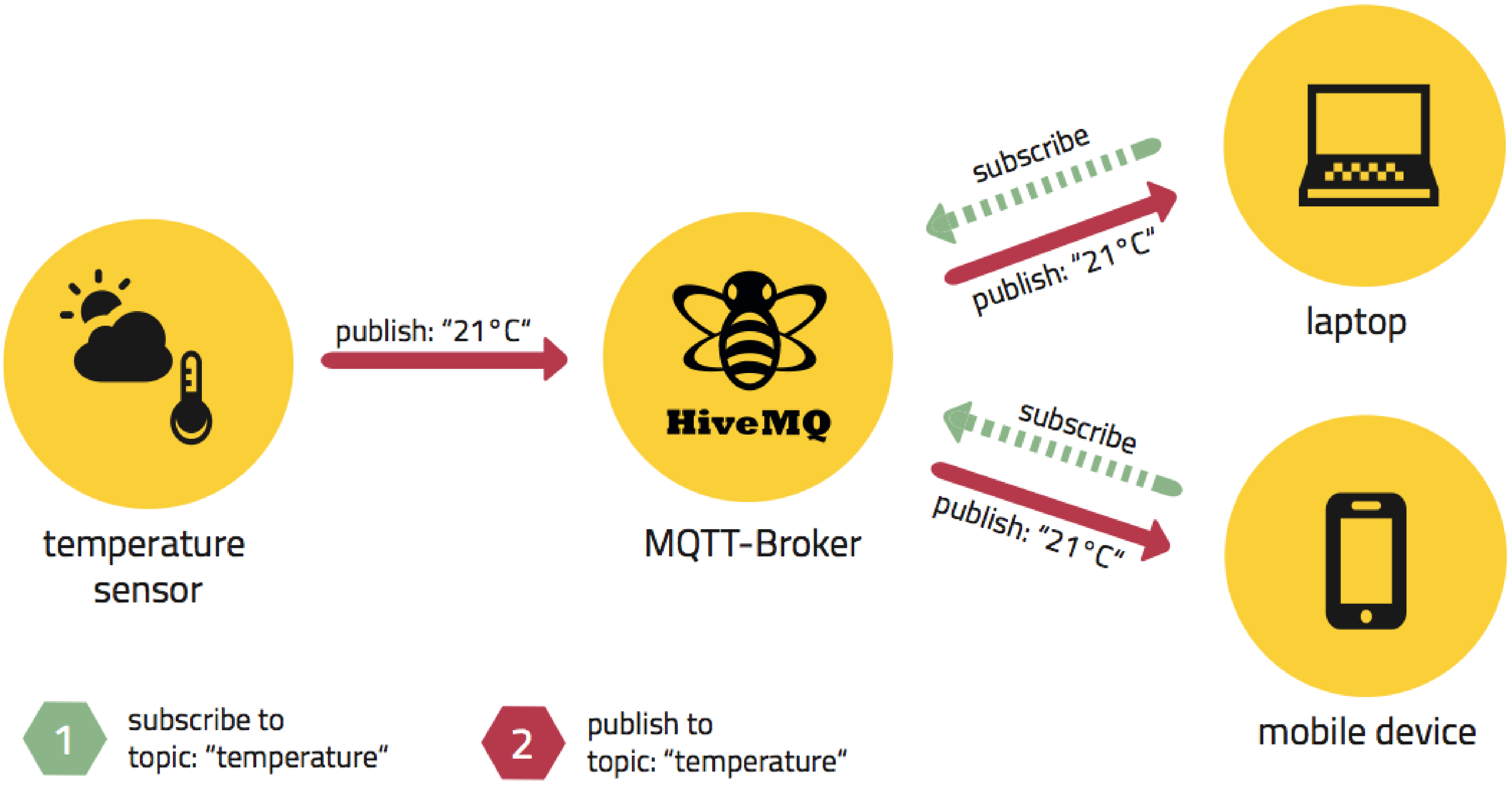
Note: Both JsonArrayRequest and JsonObjectRequest are based on JsonRequest class. The same basic pattern you use for these two types of requests.

```
val url = "http://my-json-feed"

val jsonObjectRequest = JsonObjectRequest(Request.Method.GET, url, null,
        Response.Listener { response ->
            textView.text = "Response: %s".format(response.toString())
        },
        Response.ErrorListener { error ->
            // TODO: Handle error
        }
)

// Access the RequestQueue through your singleton class.
MySingleton.getInstance(this).addToRequestQueue(jsonObjectRequest)
```

MQTT

- Message Queuing Telemetry Transport (MQTT)
- Works on top of TCP/IP
- Machine-to-machine connectivity protocol
- Uses publish/subscribe messaging transport

Internet of Things Real-time messaging



temperature sensor

publish: "21°C"

HiveMQ

MQTT-Broker

subscribe

publish: "21°C"

laptop

subscribe

publish: "21°C"

mobile device

1 subscribe to topic: "temperature"

2 publish to topic: "temperature"

Find out more

- Introduction to MQTT

  https://github.com/mqtt/mqtt.github.io/wiki

- Eclipse PAHO MQTT for Android

  https://github.com/eclipse/paho.mqtt.android

- Kotlin-MQTT implementation
  - MQTT Client

    https://gist.github.com/hussanhijazi/4fd7c737ccb4f1006ad2e36f3108ddcc

  - Android app that uses MQTT

    https://github.com/marciogranzotto/mqtt-painel-kotlin

Questions?

Why MQTT is a suitable network protocol for a mobile app developed to communicate with the IoT devices?

Mobile Server Options

- Mobile app serves as the UI or front-end

- Back-end provides databases, scripting (API) and the architecture of the app – connect DB to app

- Three main core functions:
  - Application server
  - Web server
  - Database

Popular Backend Technologies

1. Ruby on Rails – a web application development framework
2. Express/Kia/Sails – a web application framework for Node.js
3. Django – a high-level Python web framework
4. PHP Model-View-Controller (MVC) frameworks
5. Google Firebase

Server Options

DIY

- Build your own servers
- API + DB

Subscribe

- Back-End as a Service (BaaS)
- Database
- AI
- Quality control
- Security

Mix and bang

- DIY + Subscribe

Server Options - DIY

- Do-it-yourself (DIY) = create your own server

- Complete control
    - Hardware
    - Software
    - Network
    - Services

- Need more time on monitoring/maintenance

Server Options - DIY

Advantages

- Complete control
  - Hardware
  - Software
  - Network
  - Services

Disadvantages

- Heavy customisation
- Complexity
- Security vulnerabilities
- Insufficient performance
- Defective reliability
- Poor functionality
- Technical debt

Server Options - Subscribe

- Popular methods
  - Cloud - is the delivery of on-demand computing resources over the internet on a pay-for-use basis
  - Container - offers a logical packaging mechanism in which applications can be abstracted from the environment in which they actually run
  - Virtualisation - process of running a virtual instance of a computer system in a layer abstracted from the actual hardware
  - Back-end as a service (Baas) - a delivery model for a set of tools that facilitates collaboration between an organization’s software development team and the operations team

Server Options - Subscribe

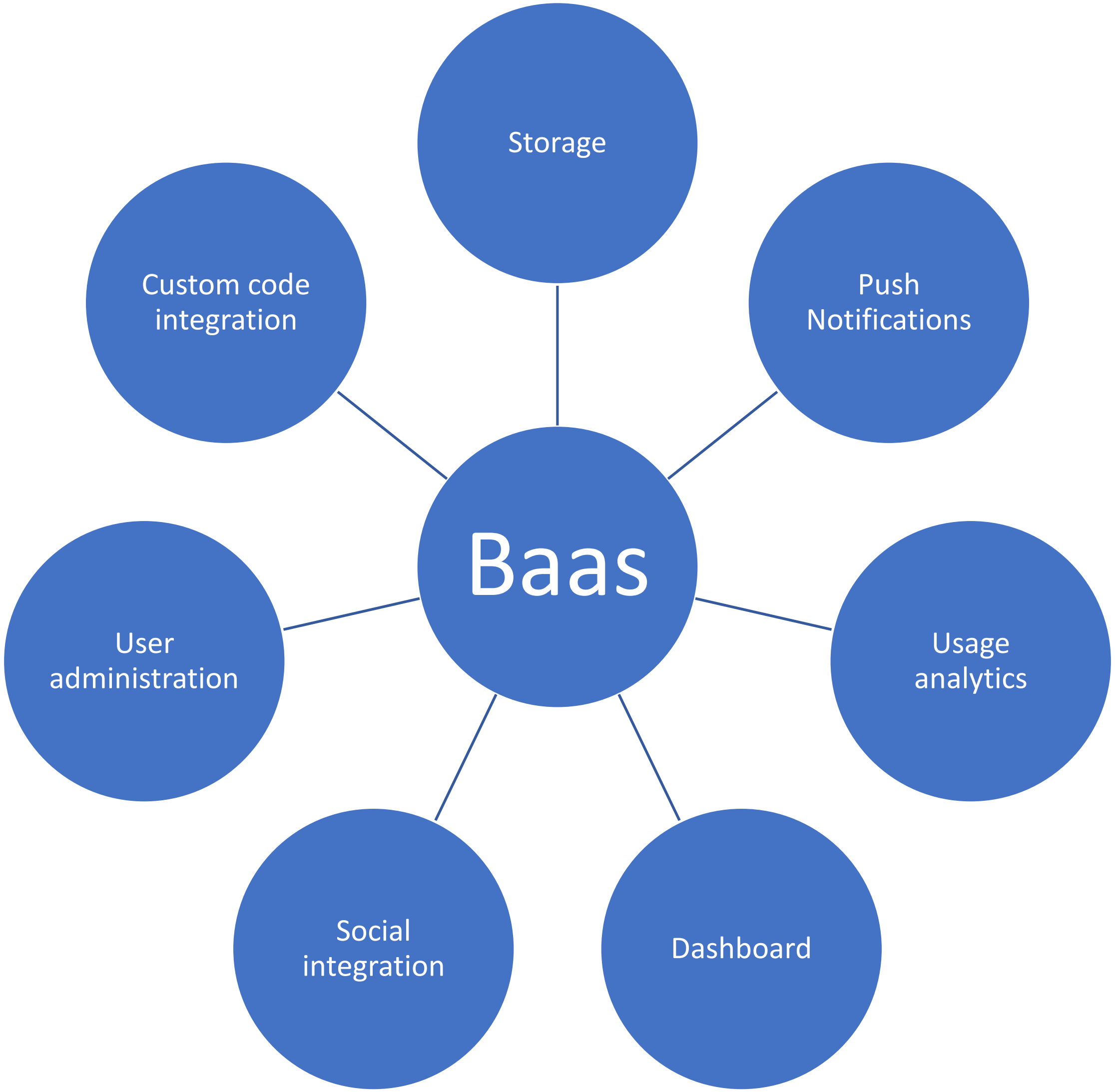
Advantages

- Cost efficiency
- Scalability
- Speed
- Integration
- Audit and compliance
- Business continuity planning

Disadvantages

- Pay-for-use
- Security

Services



Baas

Storage

Push Notifications

Usage analytics

Dashboard

Social integration

User administration

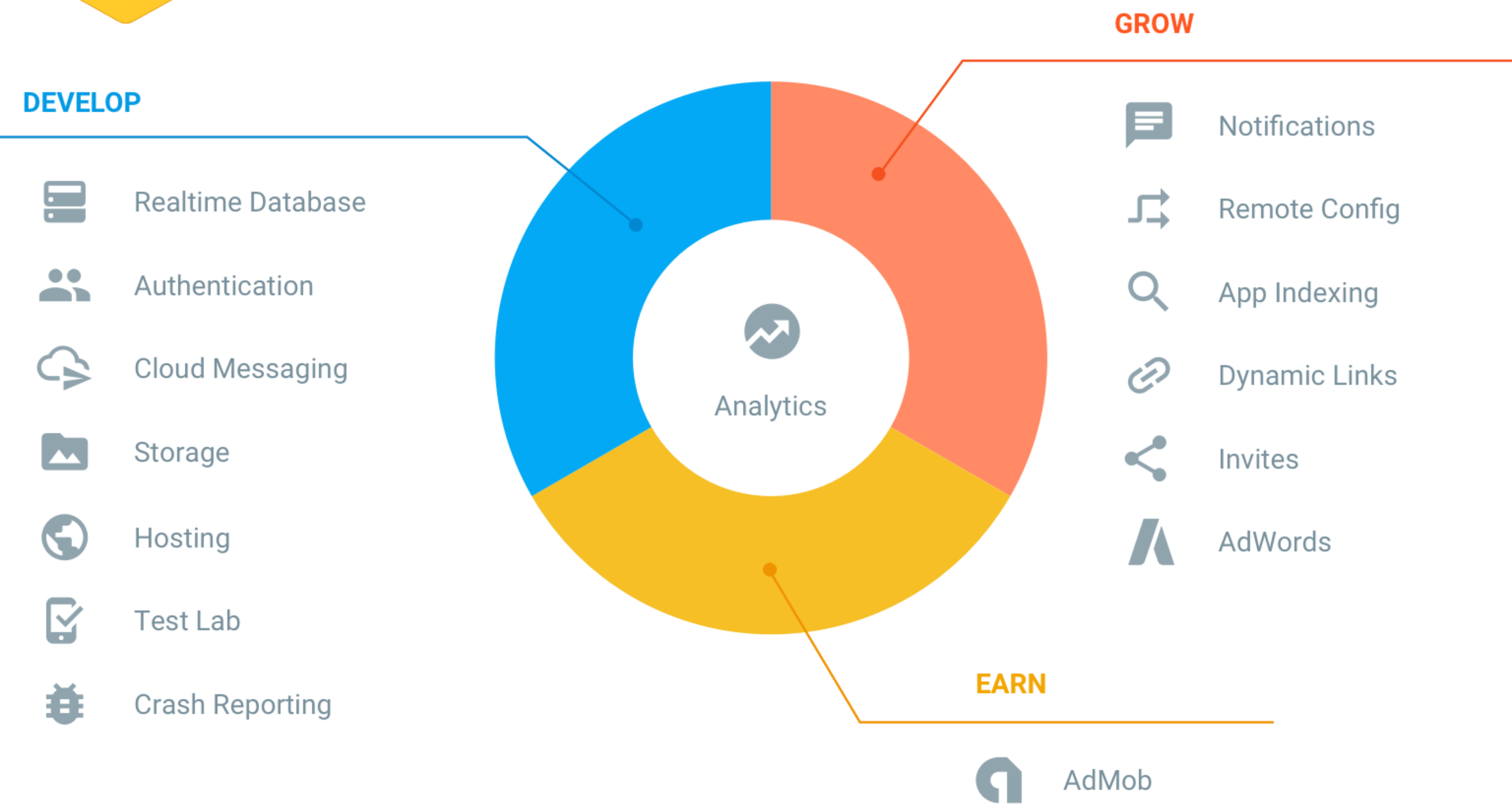
Custom code integration

Back-End as a Service (BaaS)

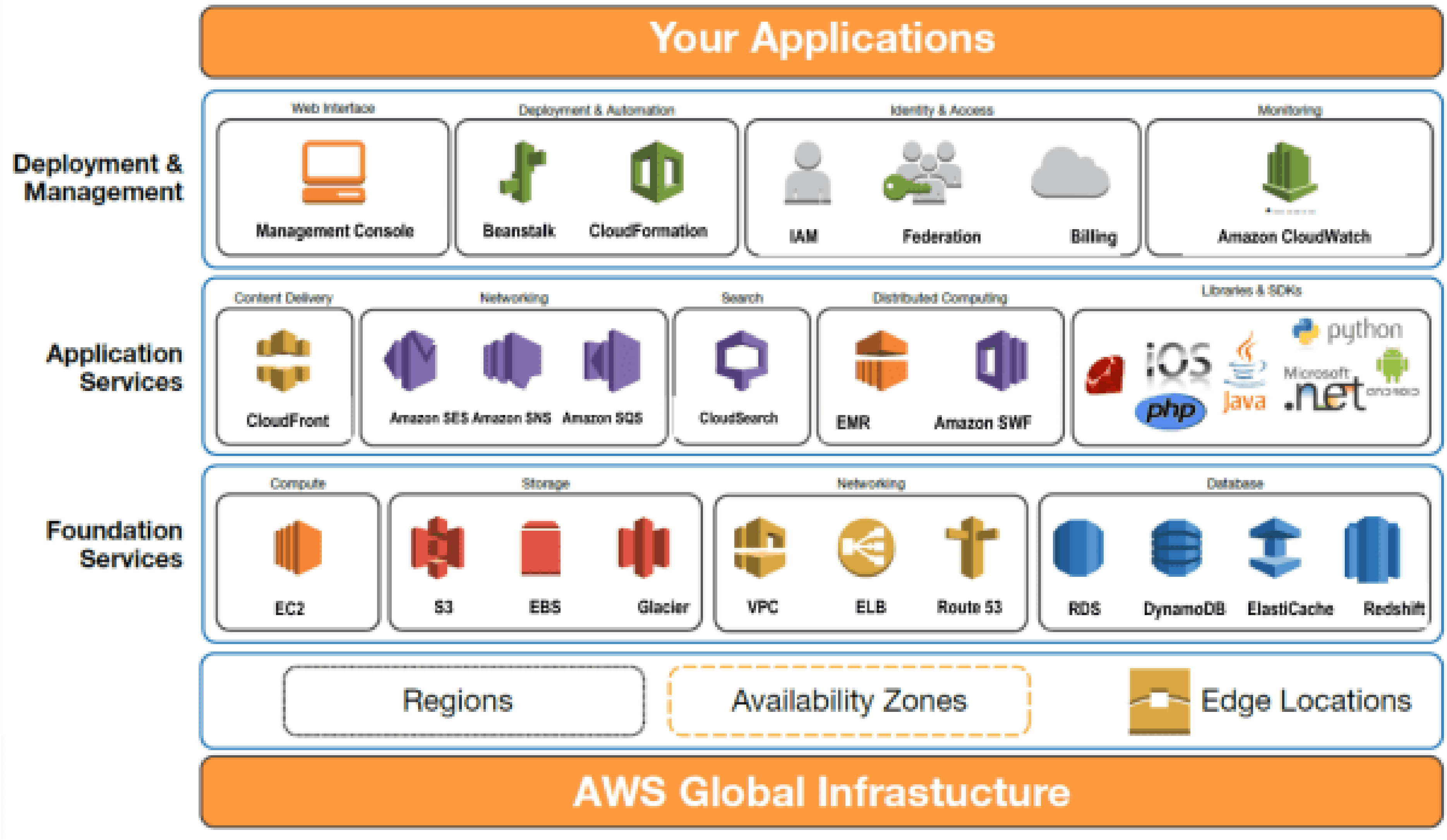
Firebase

Apple CloudKit

Firebase

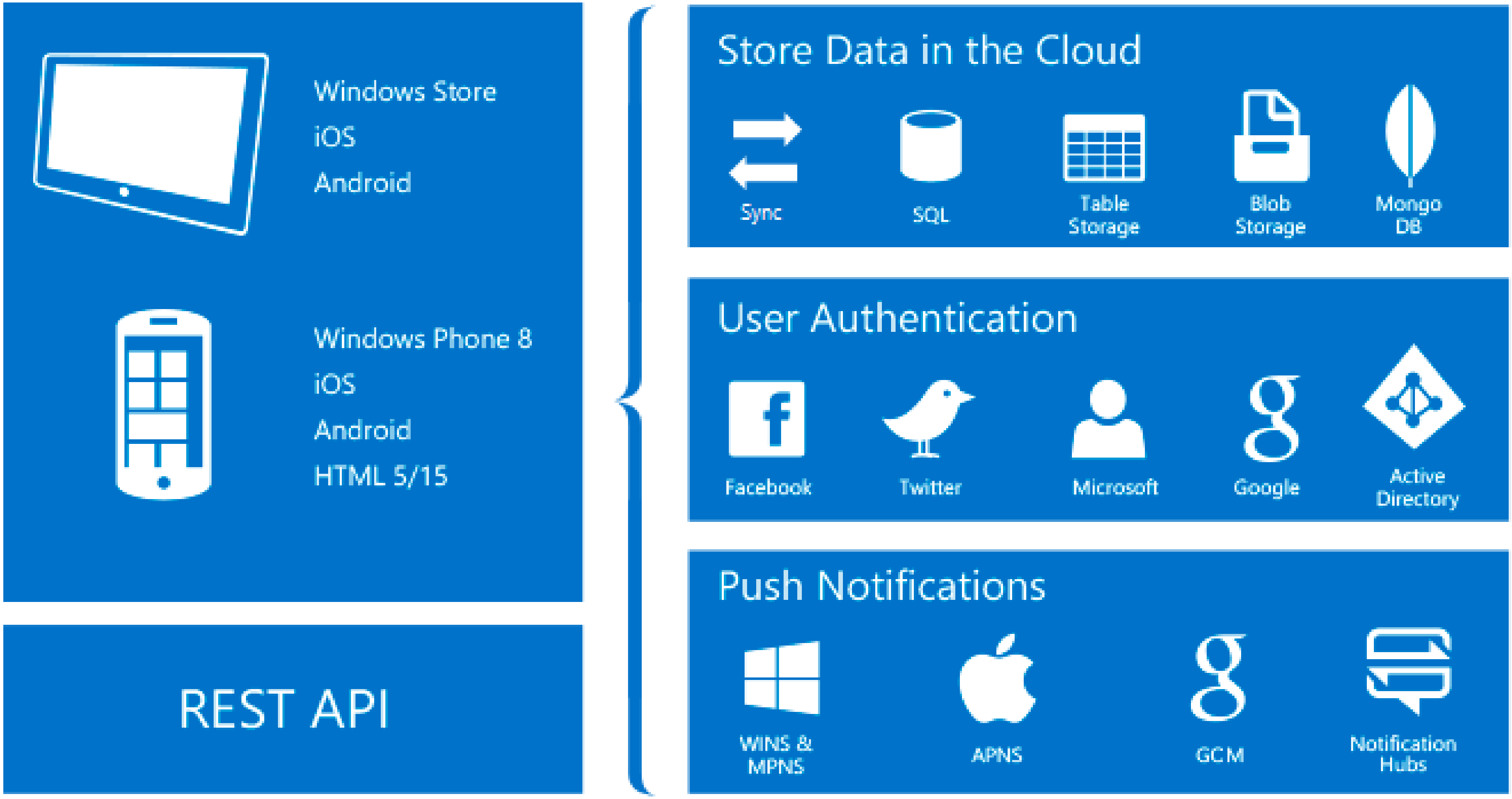


https://cloud.google.com/



https://aws.amazon.com/

Azure



Windows Store
iOS
Android

Windows Phone 8
iOS
Android
HTML 5/15

REST API

Store Data in the Cloud

Sync
SQL
Table Storage
Blob Storage
Mongo DB

User Authentication

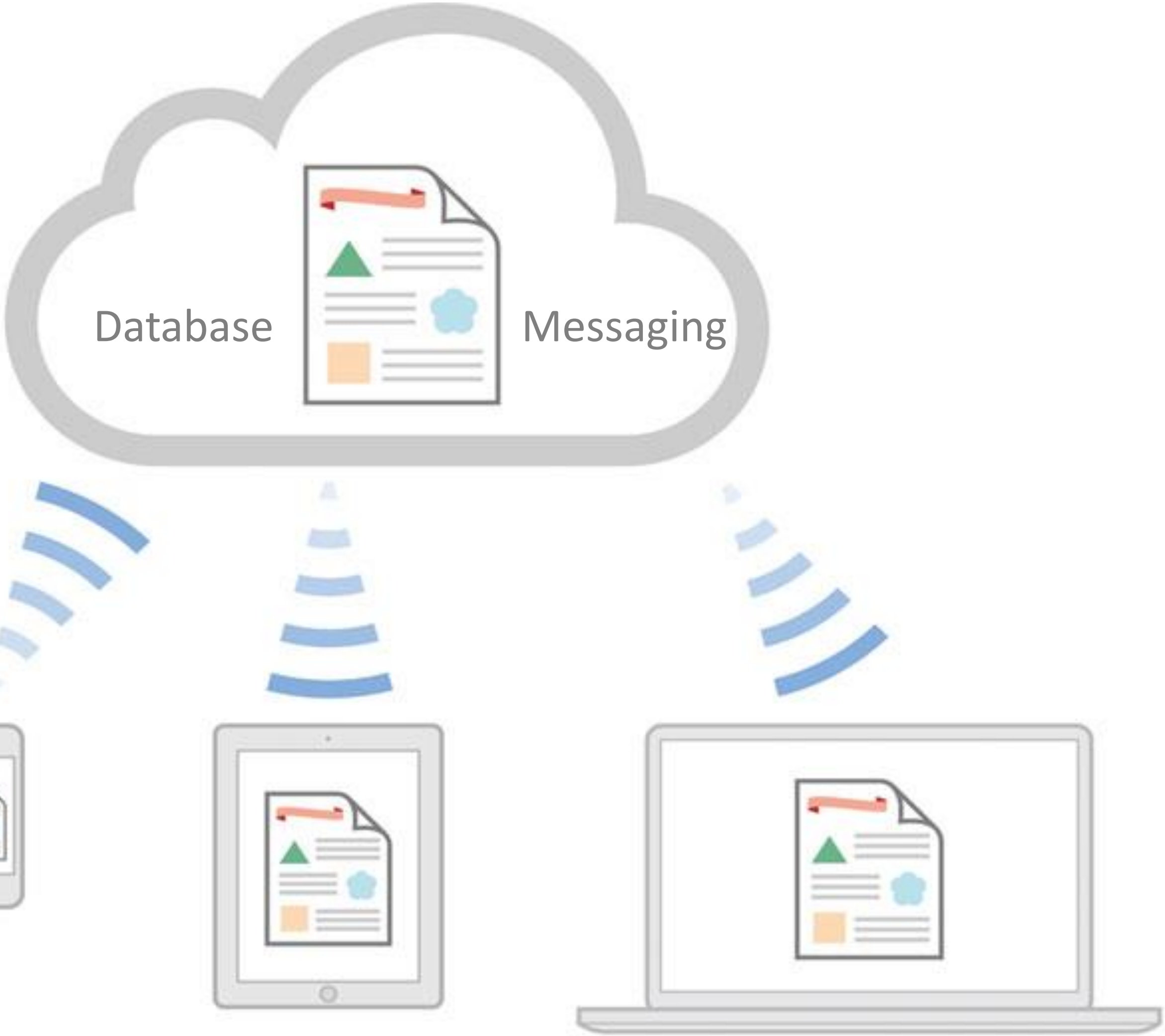
Facebook
Twitter
Microsoft
Google
Active Directory

Push Notifications

WNS & MPNS
APNS
GCM
Notification Hubs

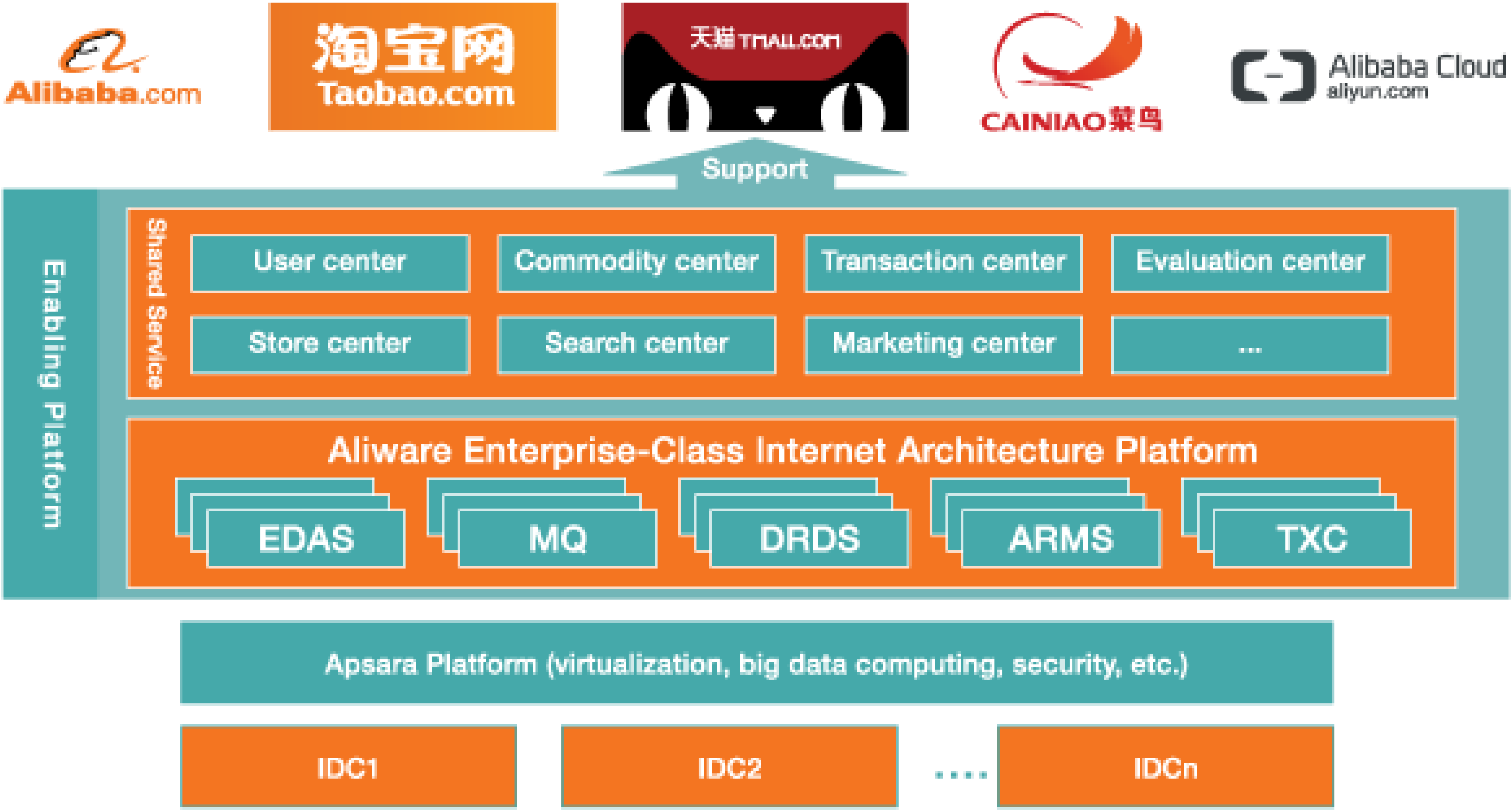
https://azure.microsoft.com/en-us/

Apple CloudKit

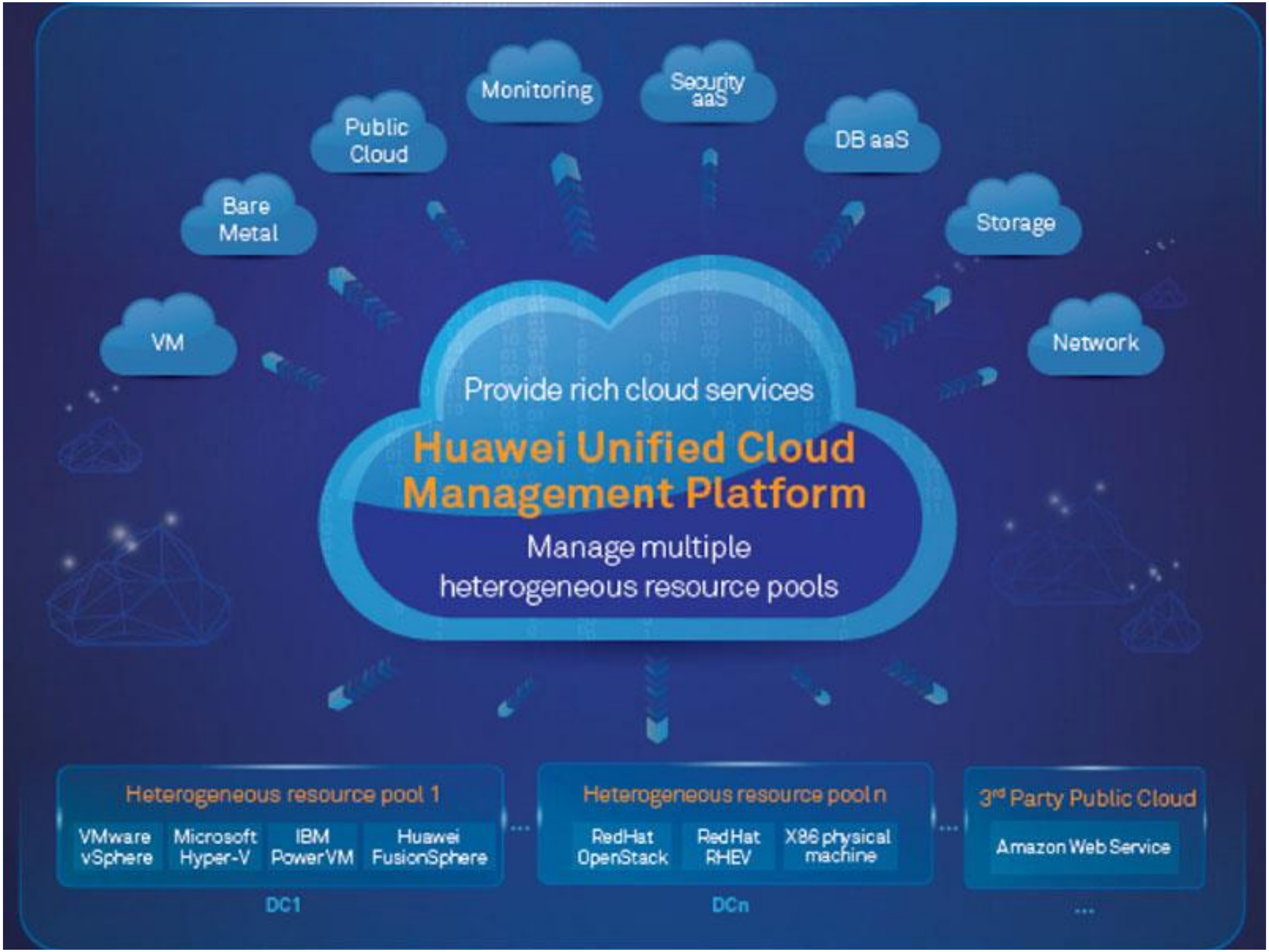


https://developer.apple.com/icloud/cloudkit/

Alibaba Cloud



https://my.alibabacloud.com/



https://intl.huaweicloud.com/en-us/

Questions?

1. Compare subscription-based and traditional DIY server solution for the mobile app.

2. H&H is Malaysia's largest car parts supplier. The company hires more than 500 salespersons to sell its products. Currently, the company operates a web-based inventory system. H&H's team needed a mobile solution to solve two problems: (i) search function on the web-based system was slow, resulting in a poor user experience, and (ii) inefficient territory management; the territory is measured by the number of salesperson against the active customers in a specific location. Proposed new features include push notification and chatbot. Suggest a way to deploy these features in the most secure and cost-effective manner.