Ionic Cloud Service 정보

중요 - 사용 절차 - 사용 방법

1. App ID 생성 및 필요한 plugin 설치
2. NgModule 에 필요한 코드 작성. 주의. 아래 요약 항목을 꼭 참고 할 것.
3. 각 서비스에 따른 셋업
    a. 각 서비스에 따른 모듈 설치
    b. 각 서비스에 따른 모듈을 소스코드에 import 및 NgModule 에 등록

설정 (셋업) - Settings - Installation & Settings 프로젝트 설정

@see official document : http://docs.ionic.io/setup.html#installation

Ionic Cloud 서비스를 사용하위해 가장 먼저 설정을 먼저 해야 한다.

Ionic Cloud 서비스의 Packaging 이던 Deploy 던 어떤 서비스를 사용하려면 먼저 설정을 해야 하는 것이다.

이러한 설정은 프로젝트 단위로 하는 것이다.
새로운 프로젝트를 위해서 ionic start 로 앱을 생성했다면

그 앱에 맞는 설정을 하는 것이다.

그리고 그 앱에 대해서 packaging 이나 deploy 등의 서비스를 사용 할 수 있다.

App ID 생성

클라우드 설정을 하기 전에 app ID 를 먼저 생성한다.

해당 프로젝트 폴더로 들어가서,

```
ionic io init
```

이와 같이 하면 http://ionic.io 홈페이지의 dashboard 에 ionic 앱이 생성된다.

- 이 때, 앱 이름은 ionic.config.json 에 있는 이름이 된다.

즉, 홈페이지에서 앱 생성 할 필요 없이, ionic cli 에서 바로 생성 할 수 있는 것이다.

이렇게 하면 ionic.io.bundle.min.js 에 app_id 와 app_key 를 지정한다고하는데,

실제로는 .io-config.json 과 ionic.config.json 파일에 앱 관련 정보가 저장된다.

- 만약 새로운 앱 ID 를 만들고 싶다면,
    - .io-config.json 을 지우고, ionic.config.json 에서 app id 를 지우면 된다.

처음 ionic start 로 앱을 생성하면 ionic.config.json 파일에는 app_id 가 공백 값을 가지는데 ionic io init 를 하면 앱 아이디의 값을 얻을 수 있다.

- **참고: 이미 App ID 가 생성되어져 있으면 그냥**
    - ionic.config.json 에
        - app id 를 입력하고,
        - “name” 을 변경하고
    - .io-config.json 에
        - app id 와

- Legacy API 의 public key 를 입력하면 된다.

Ionic Cloud Service 사용을 위한 모듈 설치

Ionic 2 앱 프로젝트 소스 폴더에서 아래와 같이 클라우드 이용에 필요한 모듈을 설치한다.

```
npm install @ionic/cloud-angular --save
```

- 참고 : 이것은 ionic 클라우드 사용에 필요한 모듈 설치하는 것이다. 굳이 ionic 2 frame 이 아니라, 다른 프레임워크에서도 사용이 가능하다.
- Ionic Cloud 를 사용하는 모든 앱에 이 과정을 해야 한다.
- 한번만 하면 모든 Ionic Cloud 서비스를 사용 할 수 있다.

앱의 프로젝트 설정을 하고 난 다음에는 앱이 Cloud Service 를 사용 할 수 있도록 기본 사용 준비를 한다.

참고) @ionic/cloud-angular 를 설치하면 아래와 같이 ionic-native 를 설치하므로 ionic 해킹을 할 때, 주의 하도록 한다.

```
C:\work\www\ionic\ionic-0.3>npm install @ionic/cloud-angular --save
ionic-hello-world@ C:\work\www\ionic\ionic-0.3
+-- UNMET PEER DEPENDENCY @angular/common@2.2.4
+-- UNMET PEER DEPENDENCY @angular/core@2.2.4
+-- UNMET PEER DEPENDENCY @angular/platform-browser@2.2.4
`-- @ionic/cloud-angular@0.8.0
  `-- @ionic/cloud@0.14.1
    +-- ionic-native@2.2.11
    `-- superagent@1.7.2
      +-- component-emitter@1.2.1
      +-- cookiejar@2.0.6
      +-- form-data@0.2.0
      | +-- async@0.9.2
      | +-- combined-stream@0.0.7
      | | `-- delayed-stream@0.0.5
      | `-- mime-types@2.0.14
      |   `-- mime-db@1.12.0
      +-- formidable@1.0.17
      +-- qs@2.3.3
      +-- readable-stream@1.0.27-1
      | `-- isarray@0.0.1
      `-- reduce-component@1.0.1
```

인증서 - Certificate 등록

http://ionic.io 에서 앱을 선택하고 Dashboard ⇒ Settings ⇒ Certificates ⇒ New Security Profile 메뉴를 클릭해서 새로운 프로필을 만든다.

- 이 때, Type 에는 Production 을 선택한다.

- 인증서 Edit 을 클릭해서 keystore 파일을 등록한다.
  참고 : keystore 파일 생성 방법 요약
    - 따라서 app 에서 keystore 파일을 먼저 생성해야 한다.

키가 이미 생성되어져 있으면 생략한다.

키 생성 예제)

```
keytool -genkey -v -keystore study.keystore -alias study -keyalg RSA -keysize 2048 -validity 10000
```

- study.keystore 대신에 자신의 앱 이름을 적어주면 된다.
- study 대신에 앱 이름을 적어주면 된다.

FCM 키 등록

이것은 push notification 을 사용 할 때만 하면 되는 것 같다.

참고 : Android 설정

app.module.ts 수정

@see official document : http://docs.ionic.io/setup.html#configuration

아래와 같이 클라우드 정보를 등록한다.

```
import { NgModule } from '@angular/core';
```

```
import { CloudSettings, CloudModule } from '@ionic/cloud-angular';
const cloudSettings: CloudSettings = {
  'core': {
    'app_id': '2e401812'
  }
};
@NgModule({
  imports: [
    CloudModule.forRoot(cloudSettings)
  ]
})
export class IonicApiModule { }
```

이렇게 하면 클라우드 서비스를 사용 할 준비가 되었다.

- 주의: Auth 와 User 와 같은 모듈을 import 하고 providers: [] 에 등록하면 에러가 발생한다.
-
- 참고: CloudModule 의 type definition 파일을 보면 provideAuth, provideUser, providerDeploy 등을 통해서 자동으로 providing 하는 것 같다.

Package

- 패키징은 소스 코드로 부터 binary 파일을 만드는 것이다.

윈도우즈에서 iOS 바이너리를 만들고 싶거나 Android 릴리즈 버전을 만들고 싶으면 Package 를 하면 된다.

- 패키징은 deploy 를 바탕으로 하므로, 먼저 deploy 를 해야 한다.

Ionic Cloud 패키징 과정

패키지 하기 위해서 먼저 Deploy 를 한다.

1. 앱을 클라우드 Deploy 하기 위해서는 단순히 “ionic upload 만 하면 끝난다.”

; 이 앱 업로드는 명령은 앱 루트 폴더에서 하면 된다. www 폴더에서 하는 것이 아님.

```
$ ionic upload

또는

$ ionic upload --note "some note" --deploy production
```

- 로그인이 안 되어져 있으면 이메일과 비밀번호를 물어온다.
- 업로드는 ionic.config.js app_id 를 보고 자동으로 해당 앱 프로젝트에 하는 것 같다.
- 참고로 --deploy production 을 해야지만, 앱에서 업데이트를 한다.
- 대시보드의 deploy 에 업로드한 앱을 볼 수 있다.
- 이것으로 deploy 는 끝났다.

2. 인증서를 등록한다. (keystore 생성하고 Dashboard 에 등록하는 것을 말 함. 프로젝트 설정 과정에서 미리 했어야 함.)

3. 업로드된 앱을 빌드한다. (이것이 패키징 하는 과정이다)

```
ionic package build android --profile helper_certificate --release
```

; 위에서 helper_certificate 값은 Ionic cloud console 의 Certificate 페이지의 Tag 값이다.
; 이렇게 빌드를 하는 것이 패키징을 하는 것이며
; Dashboard 의 Package 메뉴에 패키징된 바이너리를 다운로드 할 수 있다.
; Your app has been successfully submitted to Ionic Package!
; Build ID: 1
; We are now packaging your app.

- 이렇게 빌드를 하면 jarsigner 나 zipalign 을 할 필요가 없다.
  자동으로 Ionic cloud 가 하는 것이다.

4. 컴파일된 바이너리 파일 다운로드하기

```
ionic package download 3
```

- 또는 ionic console 에서 바로 다운로드 해도 된다.

패키징과 Deploy 관계

- 최초 패키징을 1 회 하고 바이너리가 device 에 들어가면,
- 그 이후 부터는 계속해서 deploy 로 업데이트를 하면 된다.
- 그런데 만약 deploy 를 잘못해서 앱이 크리티컬 에러 상태로 빠지면,
- 앱을 다시 바이너리로 만들어서 device 에 넣어야 한다.
    - 구글 play 스토어에는 새로운 앱 APK 를 등록하면 자동으로 업데이트를 한다.

패키지 정보 보기

앱이 비동기적으로 패키징 되므로 정보를 보고자 할 때, 아래와 같이 ionic package list 를 할 수 있다.

예제)

```
$ ionic package list
```

결과)

```
 id │ status  │ platform │ mode
────┼─────────┼──────────┼──────
 1  │ SUCCESS │ android  │ release

Showing 1 of your latest builds.
```

패키지 ID 별 정보 보기 - 실패 사유 등

예제)

```
$ ionic package info 1

 id        │ 1
 status    │ SUCCESS
 platform  │ android
 mode      │ release
 started   │ Oct 27th, 2016 18:35:40
 completed │ Oct 27th, 2016 18:36:17
```

Deploy - 배포

순서

1. Ionic Cloud 설정. 참고 - 설정 (셋업) - Settings - Installation &
2. Deploy 모듈 설치
3. Deploy 프로그래밍
4. 소스 코드 수정 후 Deploy 테스트

참고 - 동작환경

- Cordova 에서만 동작하는 것 같다. 웹에서는 동작 안 함.
    - Deploy plugin 이 웹 브라우저에서 실행되면 에러가 발생한다.
- Cordova 에서 모두 동작하는지?

- GenyMotion 과 AVD 에서 되는가?
- AndyOS 에서는 되는 것 같다.

참고 : Ionic Deploy 와 <base href=''> 관련 사항

Ionic Cloud Dashboard 에서 프로젝트 생성

새로운 프로젝트를 하나 시작한다.

앱 프로젝트 설정

참고 : 앱 프로젝트 설정

앱 사용 준비

참고 : 본 문서의 앱 사용 준비 참고

Deploy Module 설치 - Deploy 를 사용 할 수 있도록 하는 준비 - 설정

Android 와 iOS 장치에서 앱을 업데이트 할 수 있도록 해 주는 ionic-plugin-deploy 모듈을 설치한다.

```
cordova plugin add ionic-plugin-deploy
```

루트 컴포넌트에 Deploy 를 Injecting

app.component.ts 에 아래와 같이 injection 을 한다.

```
import { Deploy } from '@ionic/cloud-angular';
constructor( public deploy: Deploy ) {}
```

그리고 아래와 같이 코딩을 한다.

- 주의 아래는 테스트용으로 30 초 마다 업데이트를 확인하도록 했는데,
    - 실제로는 10분에 한번씩 또는 1분에 한번씩 확인하게 하면 된다.

```
  constructor(
    private router: Router,
    public deploy: Deploy
    ) {
    document.addEventListener("deviceready", () => this.onDevinceReady(), false);
  }
  onDevinceReady() {
    console.log("yes, I am running in cordova.");
    this.updateApp();
  }
  updateApp() {
    this.updateNewSnapshot();
    setInterval( () => this.updateNewSnapshot(), 30 * 1000 );
  }
  updateNewSnapshot() {
    console.log("MyApp::updateSnapshot()");
      this.deploy.check().then( (snapshotAvailable: boolean) => {
        if ( snapshotAvailable ) { // snapshotAvailable 이 true 이면, 새로운 snapshot 을 사용
할 수 있다.
          let opt : DeployDownloadOptions = {
             onProgress: p => {
                console.info('Downloading = ' + p + '%');
             }
          };
          this.deploy.download( opt ).then( () => { // 새로운 snapshot 을 다운로드
            let opt : DeployDownloadOptions = {
               onProgress: p => {
                  console.info('Extracting = ' + p + '%');
               }
            };
            return this.deploy.extract( opt ) // snapshot 압축 해제
              .then( () => {
                this.router.navigateByUrl( '/' ); // base href='' 때문에 안전하게 home 으로 가서
```

```
load() 함.
              setTimeout( () => {
                this.deploy.load(); // reload 해서 새로운 snapshot 을 적용
              }, 1234);
            } );
          });
        }
      });
  }
```

- 끝.
    - 위와 같이 하면 사실 상 끝난 것이다.
- 위와 같이 했으면 ionic cloud 에 deploy 하면 자동으로 업데이트 된다.
- 참고로 angular routing 을 위해서 base href 를 지정하는데, 기본 값이 빈 문자열이어서 앱을 홈(루트) 경로로 이동 한 다음 업데이트를 한다.

Ionic Could Deploy ⇒ 패키징 ⇒ 앱 다운로드 ⇒ 스마트폰 설치 및 실행

이제 부터는 그냥 deploy 하면 된다.

- 참고 : 패키징 하는 방법. Deploy 하는 방법.
- 앱을 다운로드해서 Genymotion 에서 실행한다.

소스코드 수정 - Deploy 테스트

- VSCode 에서 소스를 수정하고
  (준비 과정 없이 그냥 루트폴더에서 ionic upload 하면 된다. 그러면 ionic app id 에 맞는 곳에 그냥 업데이트 한다.)
- **npm run build --prod** 와 같이 production version 으로 www 에 배포 파일을 생성한다.

(이거 안하고, www 폴더 내용을 다 지우고 하니 에러가 난다.)
(반드시, --prod 빌드를 해야 한다.)

- ionic upload --note "update 5" --deploy production 와 같이 deploy 를 한다.
  (www 폴더가 아니라 root 폴더)

- 주의 : 이 때, packaging 은 필요 없다.
  패키징은 맨 처음 한번만 하는 것이고
  업데이트는 deploy 만 하는 것이다.

끝. 테스트 잘 되면 끝난 것이다.

주의 : 아주 중요 : HTML 에러 등 작은 에러에도 앱은 critical error 상태로 빠지며 동작하지 않는다.

예를 들어 Heading 태그를 P 태그에 넣었는데, 앱에서는 이것이 허용되지 않는다.

그래서 앱이 동작을 멈추었는데, blank 화면만 나타나고 아무런 동작을 하지 않는다.

강제 종료, 새로운 deploy, 이전 deploy 등을 해도 안된다.

따라서 작은 에러가 큰 재앙을 불러 올 수 있으므로 매우 주의 한다.

참고 : Deploy 구현 방법

앱이 deploy 방식으로 동작하기 위해서는

- Check if a new snapshot has been deployed
- Apply the snapshot
- Reload the app ; 중요 ; 앱이 자동적으로 reload 하도록 load() 함수를 사용한다.

와 같은 순서로 진행되어야 한다.

참고 : 채널 - Channels

버전 관리 비슷한 것이다. 기본적으로 production 이 된다.

아래와 같이 하면 로컬 앱을 서버에 업로드하고 snapshot 을 만든다.

```
ionic upload --note “NOTE_MESSAGE”
```

채널로 바로 deploy 하기

--deploy 를 사용하지 않으면 dashboard 에서 수동으로 채널에 deploy 해야 한다.

아래와 같이 하면 채널에 바로 deploy 한다.

```
ionic upload --note “NOTE_MESSAGE” --deploy CHANNEL_TAG
```

CHANNEL_TAG 는 production 으로 하면 된다.

채널을 변경하기 위해서는 deploy 관련 코드를 사용하기 전에 먼저와 같이 하면 된다.

```
this.deploy.channel = 'dev';
```

기본적으로 production 모드가 사용된다.
만약 존재하지 않는 channel 을 사용하면 자동으로 production 으로 fallback 된다.

참고 : Snapshot 관리하기

Listing, Removing, Metadata 등을 할 수 있다.

기타

Deploy 를 하기 위해서는 앱의 소스에 코드를 추가 해야 한다.

- Deploy 와 Cloud 플러그인 설치

```
cordova plugin add ionic-plugin-deploy --save
npm install @ionic/cloud-angular --save
```

- App module 에서 cloud 와 deploy 설정
- 수정 할 페이지에서 Deploy 를 inject 한다.

- 수정된 소스 코드를 Ionic Cloud 로 업로드

```
ionic upload --note "Update for deploy 2" --deploy production
```

- 테스트 : 다운로드한 apk 파일을 genymotion 의 에뮬레이터로 드래그해서 집어 넣는다.

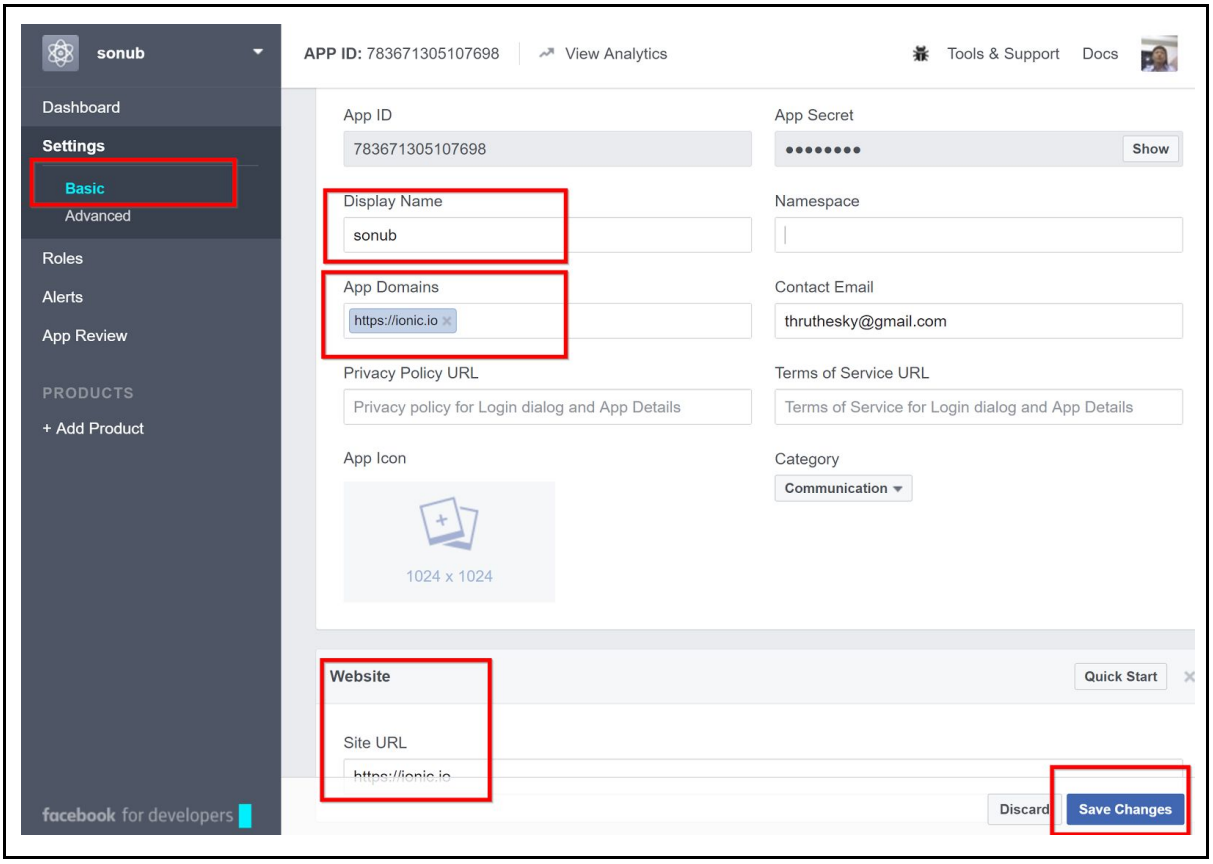
Ionic Authentication

Before anything else, do Ionic Cloud App ID Setup first.

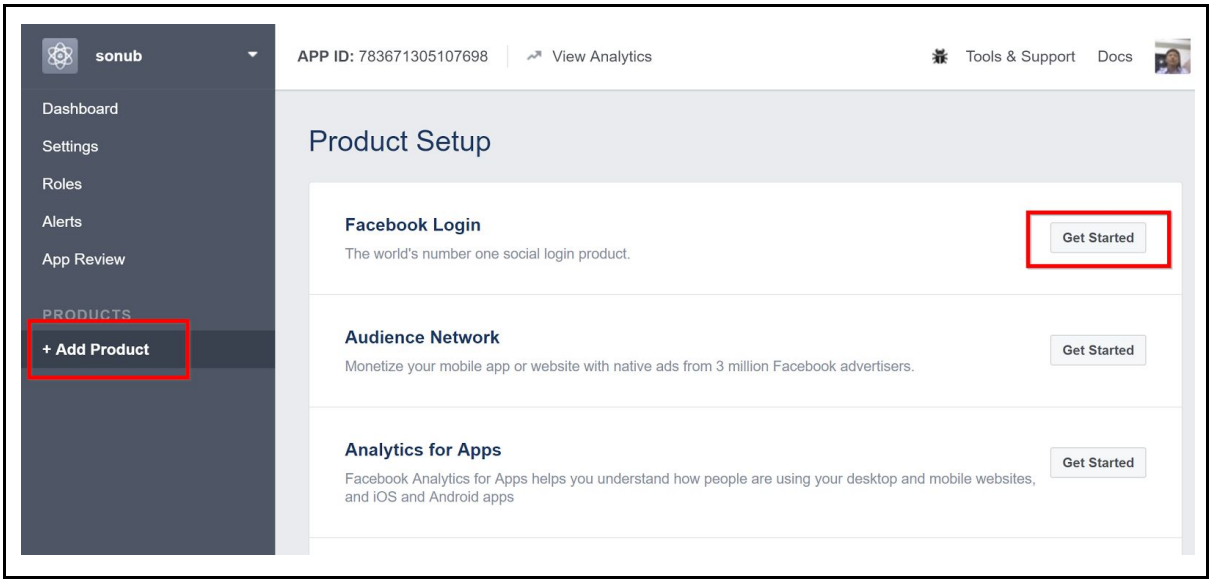
Facebook In-App-Browser

Setup Facebook In-App-Browser

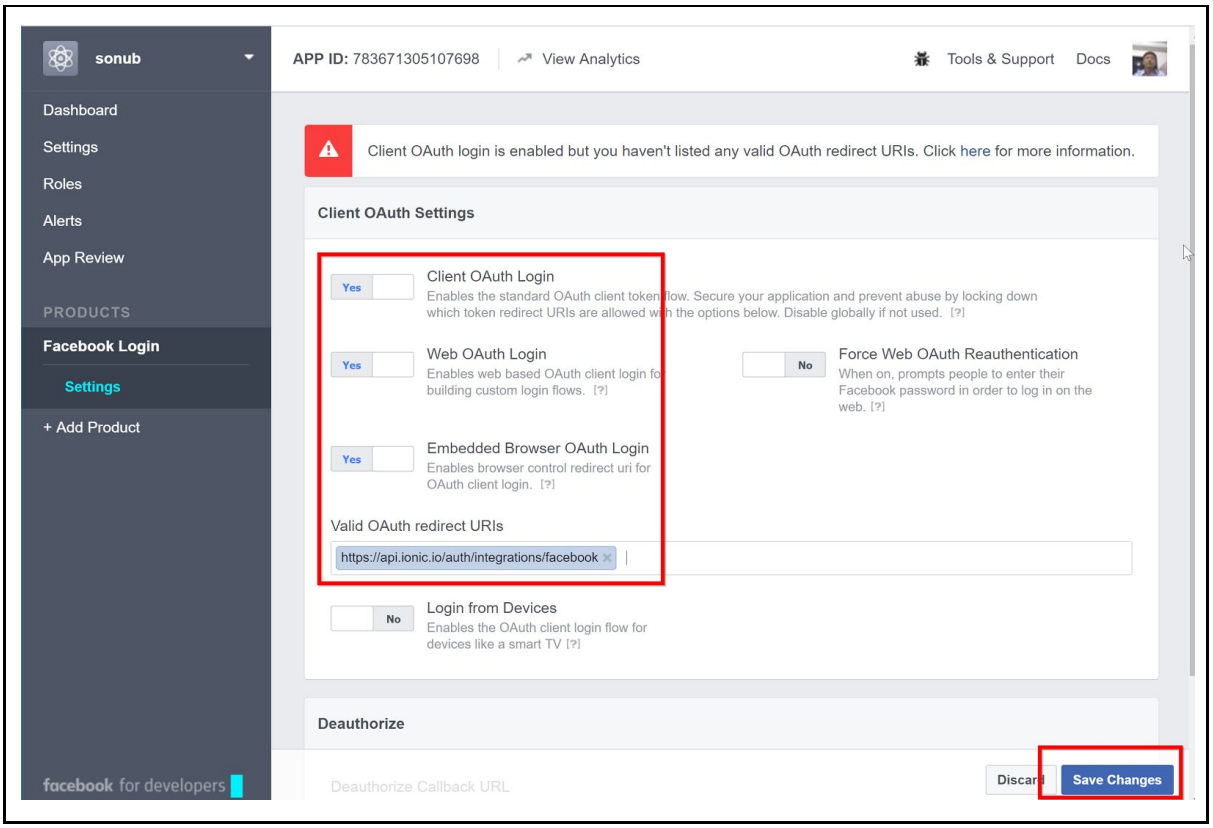
1. Create facebook app.
2. App domain must be https://ionic.io
3. Add website platform and put address : https://ionic.io
4. Save changes.



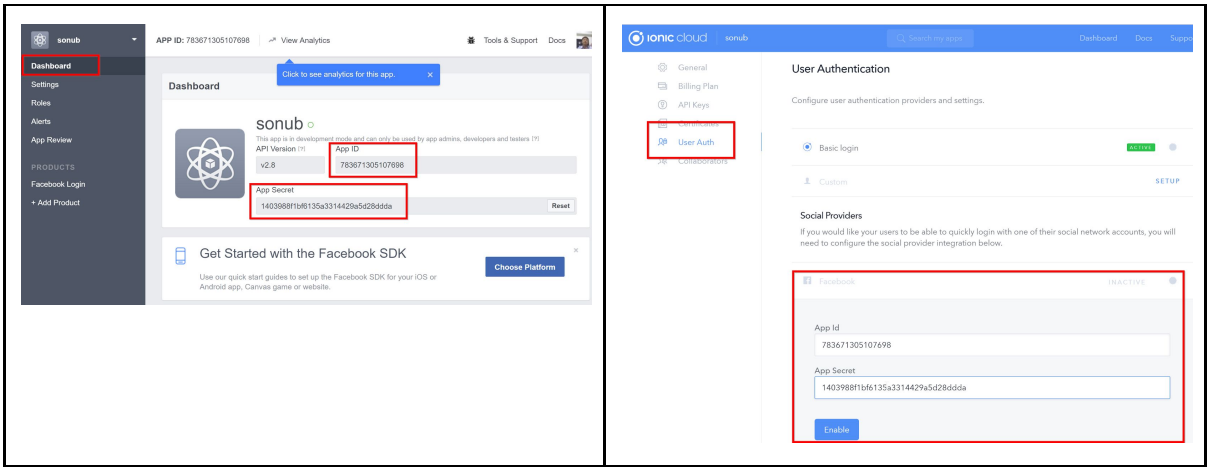
- Go to “Add Product” menu and choose “Facebook Login”



- Enable Client OAuth, Web OAuth, Embedded Browser OAth
- Input Valid OAuth redirect URIs : https://api.ionic.io/auth/integrations/facebook



- Copy App ID and Secret from Facebook and Paste them into Ionic Cloud



- Install cordova in-app-browser plugin

```
cordova plugin add cordova-plugin-inappbrowser --save
```

- Now program it.

@see official doc :
http://docs.ionic.io/services/auth/facebook-auth.html#injecting-auth-and-user

Facebook Native Authentication

In their document, it is stated that ‘native’ is ‘preferred’.

But there is a serious problem on it. Native works only native (device). I does not run in webapp. Let’s say, your webapp is running on android web browser, this will not work.

If you plan to develop webapp & app, then go for In-app-browser.

@see http://docs.ionic.io/services/auth/facebook-native.html#create-a-facebook-app

Native setup for facebook

1. Create app in facebook.
   @see official doc
   http://docs.ionic.io/services/auth/facebook-native.html#create-a-facebook-app

Authentication 예제

먼저 준비를 한다.

예제) 로그인을 위한 코드

```
import { Component } from '@angular/core';
import { Auth, User } from '@ionic/cloud-angular';

@Component( ... )
export class LoginPage {
  constructor(public auth: Auth, public user: User) {
    ...
  }
}
```

예제) 로그인 코드

- 로그인을 한 다음에 this.user.details 에 기본 정보가 들어가 있다.
- 이 this.user.details 는 사용자의 정보가 들어가 있다.
- 아이디와 비밀번호를 입력하지 않아도 새 창이 열리면서 사용자가 입력을 한다.

```
  onClickTwitterLogin() {

    this.auth.login('twitter', { remember: true })
      .then( re => {
        console.log(re);
        console.log("user: ", this.user.details );
      })
      .catch( e => {
        console.log(e);
      });

  }
```

한번 로그인을 한 다음에는 home.ts 에서 아래와 같이 해도 로그인 정보를 얻을 수 있다.

```
import { User  } from '@ionic/cloud-angular’;

constructor( private user: User ){}

  ionViewWillEnter() {
    console.log('HomePage::ionViewWillEnter()')
    console.log('social login user: ', this.user.details );
```

```
  this.checkLogin();
}
```

Auth.set() 다음에 Auth.save() 를 호출해야 서버에 데이터가 저장 됨

auth.set() 은 local device 에만 정보를 저장하는 것 같다.

Angular2 에서 Ionic Cloud 사용하기

- Ionic2 가 IE 를 지원하지 않아 Angular2 를 사용하는데, Deploy 기능이 몹시 필요하다.

Angular2 와 Ionic Cloud Deploy

- Ionic cloud 에 회원 가입
- @ionic/cloud-angular 모듈 설치

```
npm install @ionic/cloud-angular --save
```

- ~~프로젝트 폴더에서 “ionic io init” 로 ionic cloud 초기화를 해야 한다.~~
  하지만 Angular2 앱이라서, “Couldn't find ionic.config.json file. Are you in an Ionic project?” 에러가 발생한다.
  따라서, “ionic io init” 는 생략한다.

  대신, ionic.io 로 로그인을 해서, 기존의 앱 ID 를 가져온다.

- app.module.ts 에 아래와 같이 추가를 한다.

```
import { CloudSettings, CloudModule } from '@ionic/cloud-angular';
const cloudSettings: CloudSettings = {
  'core': {
    'app_id': ‘YOUR_APP_ID’
  }
};

@NgModule({
  imports: [ CloudModule.forRoot(cloudSettings) ]
})
```

Push Notification

Ionic 의 Push 는 Android 의 경우, Google firebase 를 바탕으로 한다.

순서

1. Ionic Cloud 설정. 참고 - 설정 (셋업) - Settings - Installation &
2. Android 설정
3. App module 에 등록
4. Cordova 플러그인 설치
5. API Token 과 함께 프로그래밍

클라이언트 앱의 ionic cloud 셋업 - 공통

프로그램 중인 앱 소스 폴더에 ionic cloud 셋업한다.

이것은

1. 새로운 프로젝트를 생성하거나 기존 프로젝트의 “app id” 를 얻어야 하며,
2. 개발 프로젝트에서 @ionic/cloud-angular 설치하고
3. app.module.ts 에 ionic cloud 설정 코드 추가 해야 한다.

Android 설정

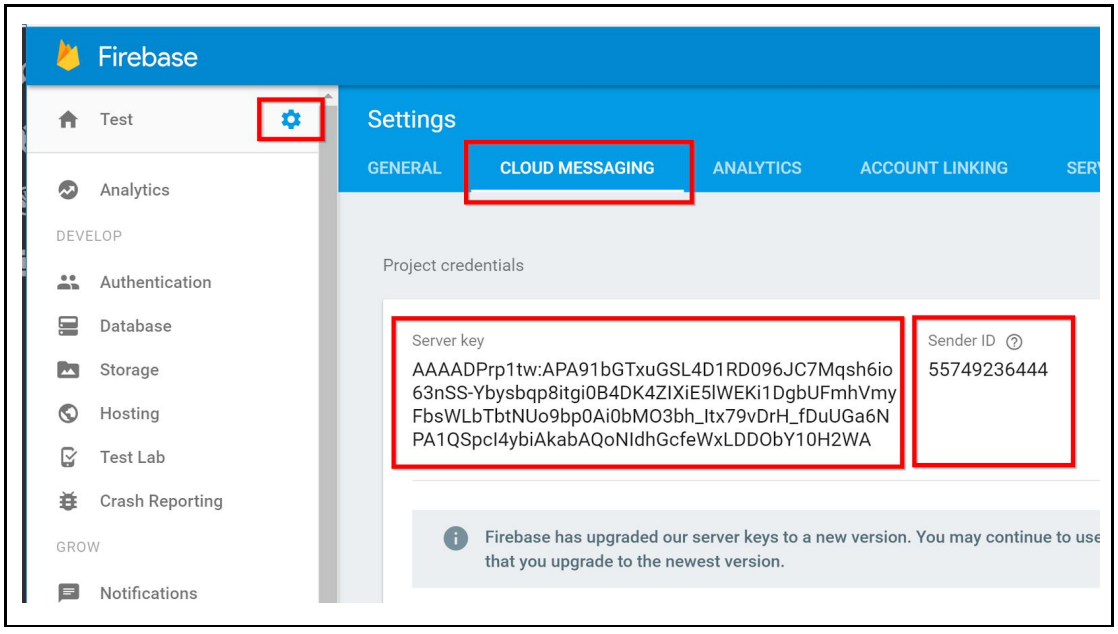
공식문서 : http://docs.ionic.io/services/push/#prerequisites

참고로 Android 와 iOS 에서 설정이 약간 틀리다.

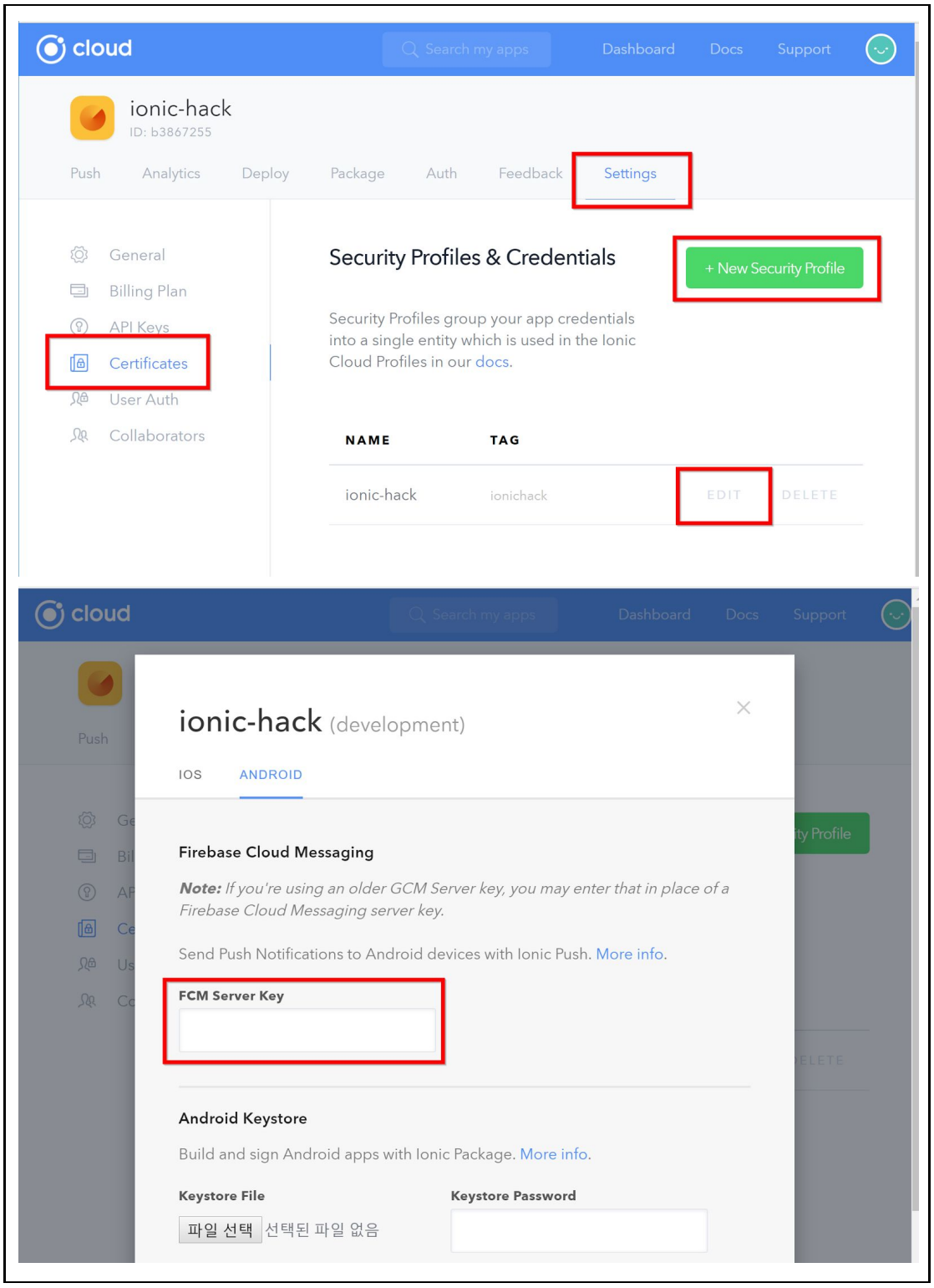
설정만 틀릴 뿐 나머지 사용하는 방법이 동일하고 하나 코딩으로 동작하는 것 같다.

본인의 에 로그인을 한 다음,

1. FCM server key 와 sender id 를 구해서



2. **Ionic cloud** 의 해당 앱 설정에 등록한다.
   FCM Server key 에 위의 server key 값을 기록한다.



참고로 Android Keystore 는 ionic Package 를 사용 할 때, “build and sign” 을 위해서 필요한 것 같다. Push Notification 과는 상관이 없는 것 같은데, 어쨌든 앱을 만들기 위해서는 이 과정이 필요하다.

App module 에 등록 - 공통

Module 에 아래와 같이 등록한다.

```
const cloudSettings: CloudSettings = {
  'core': {
    'app_id': 'APP_ID',
  },
  'push': {
    'sender_id': 'SENDER_ID',
    'pluginConfig': {
      'ios': {
        'badge': true,
        'sound': true
      },
      'android': {
        'iconColor': '#343434'
      }
    }
  }
};
```

위에서

- APP_ID 는 ionic app id 이고,
- SENDER_ID 는 구글 firebase push notification setting 의 sender_id 이다.
- 그리고 Server Key 는 Ionic Cloud 설정에 저장되어져 있다.

Cordova 플러그인 설치 - 공통

그리고 push 를 위한 cordova 플러그인을 설치한다.

```
$ cordova plugin add phonegap-plugin-push --variable SENDER_ID=xxxxx --save
```

위에서 sender_id 는 Google Firebase 의 FCM sender id 이다.를 입력하면 된다.

위와 같이 하면 ionic 프로젝트의
platforms/android/platform_www/plugins/phonegap-plugin-push 를 볼 수 있다.

그러면 아래와 같이 config.xml 에 저장된다.

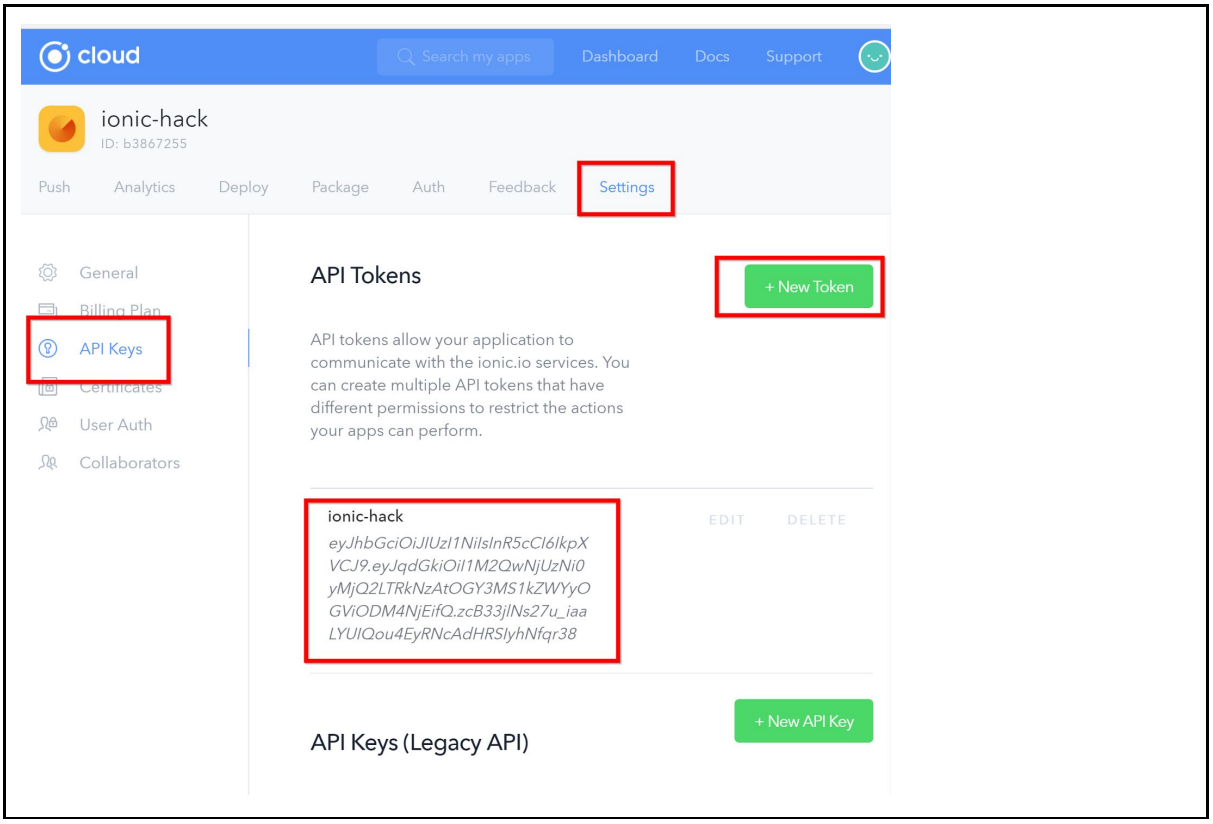
```
<plugin name="phonegap-plugin-push" spec="~1.9.1">
  <variable name="SENDER_ID" value="55749236444"/>
</plugin>
```

API Token API - HTTP Request 를 하기 위한 api token

설정이 끝났으면, 실제로 ionic cloud 와 대화(질의)를 통해서 push 관련 정보를 송/수신한다.

이 때, 인증을 위해서 사용 할 api token 이 필요하다.

아래와 같이 ionic cloud 설정에서 새로운 토큰을 만든다.



Push Notification Coding - 실제로 앱에 코딩하는 방법

공식 문서 - EndPoint API 문서

본 문서는 중요하다.

http://docs.ionic.io/api/endpoints/push.html 에 보면, EndPoint URL 의 설명이 있다.

Register - 장치를 등록

공식 문서 : http://docs.ionic.io/services/push/#registering-device-tokens

가장 먼저 해야 할 것은 장치를 Ionic Cloud 에 등록하는 것인데,

```
push.register()
```

와 같이 하면 된다.

중요한 것은

- (공식문서에서) 앱이 실행될 때 마다 이것을 하라고 한다.
  이렇게 함으로써 장치가 항상 등록되고 push notification 을 받을 수 있는 상태가 된다고 한다.
- Ionic Auth 를 사용하면 … 뭐라고 하는데,
- Ionic Auth 를 사용하지 않으면 saveToken() 의 옵션에 ignore_user 속성을 지정하라고 한다.

Unregister - 장치 등록 해제

굳이 장치를 등록 해제 할 일은 없으니 생략한다.

Handling Notification - 메세지 받기

아래와 같이 subscribe() 를 하면 (누군가 메세지를 전송할 때,) 자동으로 메세지가 수신된다.

```
push.rx.notification()
    .subscribe( (msg) => {
        alert(msg.title + ':' + msg.text);
    });
```

메세지 보내기

공식 문서 : http://docs.ionic.io/services/push/#sending-pushes

메시지는 대시보드에서 전체 메세지를 보낼 수 있다.

개별적으로 보내려면 프로그래밍 작업이 필요하다.

메세지를 보낼 상대 디바이스 토큰을 알아야하는데,

이것은 각 회원 별로 장비를 register() 하고 받은 토큰을 회원 정보에 저장하므로서 가능하다.