Google developers

1. 官方文档

https://developers.google.com/android/guides/overview

2. Google login

Google Account Login

```
compile com.google.android.gms:play-services-auth:11.6.0
```

1> 说明

https://developers.google.com/identity/sign-in/android/

- 获取用户信息
- 获取邮箱
- 获取 idToken

- 获取服务器验证码 - web 端鉴权

2> 登录方式

> 11.6 版本使用 GoogleSignInClient 登录
>
> *GoogleSignInOptions.DEFAULT_SIGN_IN*
>
> 9.4 版本使用 **GoogleApiClient**

- 界面 UI 登录

```
GoogleSignInOptions gso = new
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
        .requestEmail()
        .build();
mGoogleSignInClient = GoogleSignIn.getClient(this, gso);

private void signIn() {
    Intent signInIntent = mGoogleSignInClient.getSignInIntent();
    startActivityForResult(signInIntent, RC_SIGN_IN);
}
```

- 静默登录

  如果静默登录失败，需要主动 UI 登录；

```
Auth.GoogleSignInApi.silentSignIn(mGoogleAuthApiClient).setResultCallback(new
ResultCallback<GoogleSignInResult>() {
    @Override
    public void onResult(@NonNull final GoogleSignInResult signInResult) {
        handleSignInResult(signInResult);
    }
});
```

3. Google game

Google Play Game services

```
compile com.google.android.gms:play-services-games:11.6.0
```

1> 说明

https://developers.google.com/games/services/android/quickstart

提供跨平台的游戏服务，集成游戏特性

- 成就
- 排行榜
- 游戏存档
- 平台或者移动游戏多玩家实时
- 礼物

- 游戏录制

2> 登录方式

> 11.6 版本使用 GoogleSignInClient 登录
>
> *GoogleSignInOptions.DEFAULT_GAMES_SIGN_IN*
>
> 9.4 版本使用 **GoogleApiClient**

- 判断是否已经登录

```
private boolean isSignedIn() {
    return GoogleSignIn.getLastSignedInAccount(this) != null;
}
```

- 静默登录

  只有在已有账号登录的情况下，静默才会成功，若失败（错误码为 CommonStatusCodes.SIGN_IN_REQUIRED），需要主动去使用非静默方式登录。

```
private void signInSilently() {
    GoogleSignInClient signInClient = GoogleSignIn.getClient(this,
            GoogleSignInOptions.DEFAULT_GAMES_SIGN_IN);
    signInClient.silentSignIn().addOnCompleteListener(this,
            new OnCompleteListener<GoogleSignInAccount>() {
                @Override
                public void onComplete(@NonNull Task<GoogleSignInAccount> task) {
                    if (task.isSuccessful()) {
                        // The signed in account is stored in the task's result.
                        GoogleSignInAccount signedInAccount = task.getResult();
                    } else {
                        // Player will need to sign-in explicitly using via UI
                    }
                }
            });
}
@Override
protected void onResume() {
    super.onResume();
    signInSilently();
}
```

- 界面 UI 登录

```
private static final int RC_SIGN_IN = 9001;

private void startSignInIntent() {
    GoogleSignInClient signInClient = GoogleSignIn.getClient(this,
            GoogleSignInOptions.DEFAULT_GAMES_SIGN_IN);
    Intent intent = signInClient.getSignInIntent();
    startActivityForResult(intent, RC_SIGN_IN);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == RC_SIGN_IN) {
        GoogleSignInResult result = Auth.GoogleSignInApi.getSignInResultFromIntent(data);
        if (result.isSuccess()) {
            // The signed in account is stored in the result.
            GoogleSignInAccount signedInAccount = result.getSignInAccount();
        } else {
            String message = result.getStatus().getStatusMessage();
            if (message == null || message.isEmpty()) {
                message = getString(R.string.signin_other_error);
            }
            new AlertDialog.Builder(this).setMessage(message)
                    .setNeutralButton(android.R.string.ok, null).show();
        }
    }
}
```

- 获取数据 `GoogleSignInAccount`

```
// 成就
mAchievementsClient = Games.getAchievementsClient(this, googleSignInAccount);
// 排行榜
mLeaderboardsClient = Games.getLeaderboardsClient(this, googleSignInAccount);
// 游戏玩家信息
mPlayersClient = Games.getPlayersClient(this, googleSignInAccount);
mPlayersClient.getCurrentPlayer()
                .addOnCompleteListener(new OnCompleteListener<Player>() {
                    @Override
                    public void onComplete(@NonNull Task<Player> task) {
                        String displayName;
                        if (task.isSuccessful()) {
                            displayName = task.getResult().getDisplayName();
                        } else {
                            Exception e = task.getException();
                            handleException(e, getString(R.string.players_exception));
                            displayName = "???";
                        }
                        mMainMenuFragment.setGreeting("Hello, " + displayName);
                    }

                });
```

- 登出

```
mGoogleSignInClient.signOut().addOnCompleteListener(this,
        new OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                boolean successful = task.isSuccessful();
                Log.d(TAG, "signOut(): " + (successful ? "success" : "failed"));

                onDisconnected();
            }
        });
```

3> 功能

- 排行榜

```
mLeaderboardsClient.getAllLeaderboardsIntent()
        .addOnSuccessListener(new OnSuccessListener<Intent>() {
            @Override
            public void onSuccess(Intent intent) {
                startActivityForResult(intent, RC_UNUSED);
            }
        })
        .addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                handleException(e, getString(R.string.leaderboards_exception));
            }
        });
```

- 成就榜

```
mAchievementsClient.getAchievementsIntent()
        .addOnSuccessListener(new OnSuccessListener<Intent>() {
            @Override
            public void onSuccess(Intent intent) {
                startActivityForResult(intent, RC_UNUSED);
            }
        })
        .addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                handleException(e, getString(R.string.achievements_exception));
            }
        });
```

Google Play 发布

发布应用之前需要先注册开发者账户，然后可以使用Google Play 开发者控制台创建应用并发布到 Google Play。

1. 创建、发布应用

1> 创建应用

在`Google Play 开发者控制台`主页上点击左上角标签`所有应用`会列出当前账户下创建的所有应用，包括已发布、草稿、更新信息等不同状态的应用，点击相应的应用会跳转到相应的详情页中。

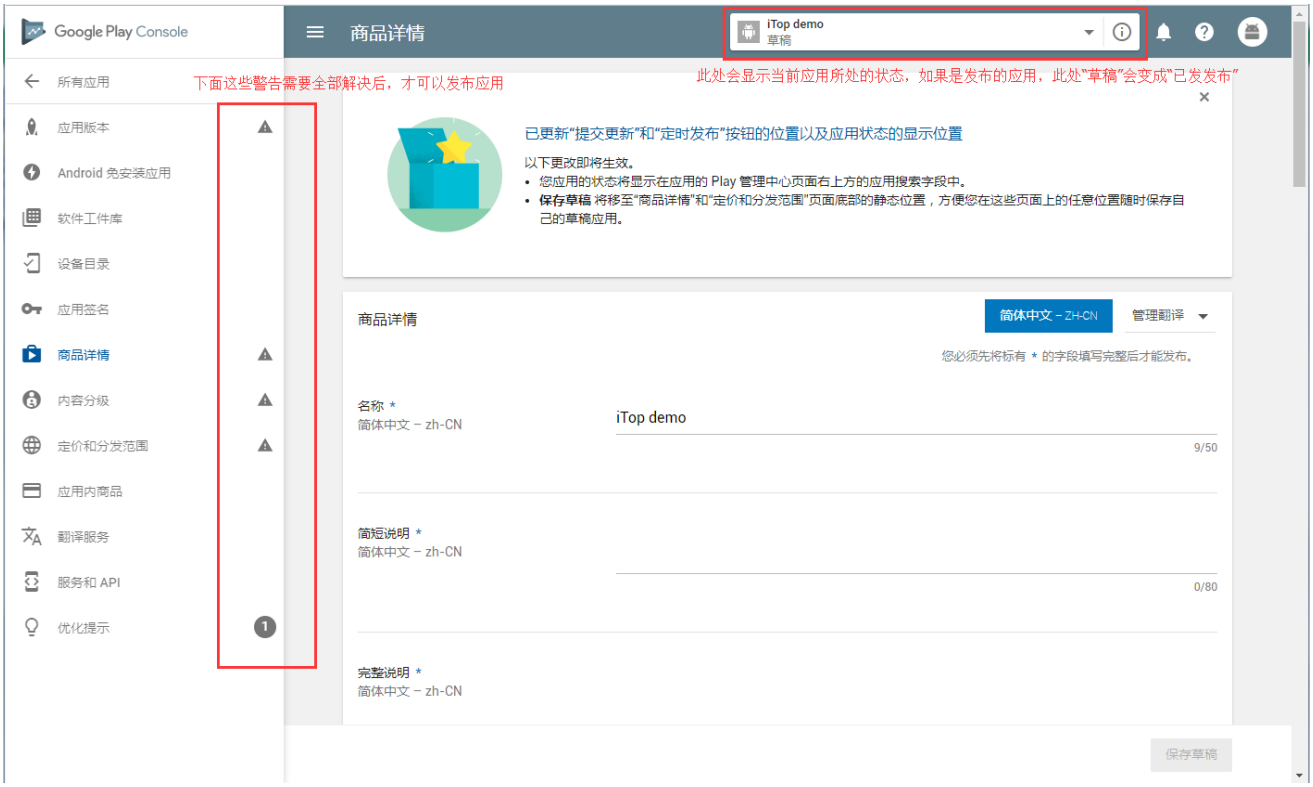
如果还没有创建应用的，可以点击右边的`创建应用`进行创建。

点击创建后会出现弹窗输入

- 默认语言
- 名称 - 在Google Play上看到的应用名称

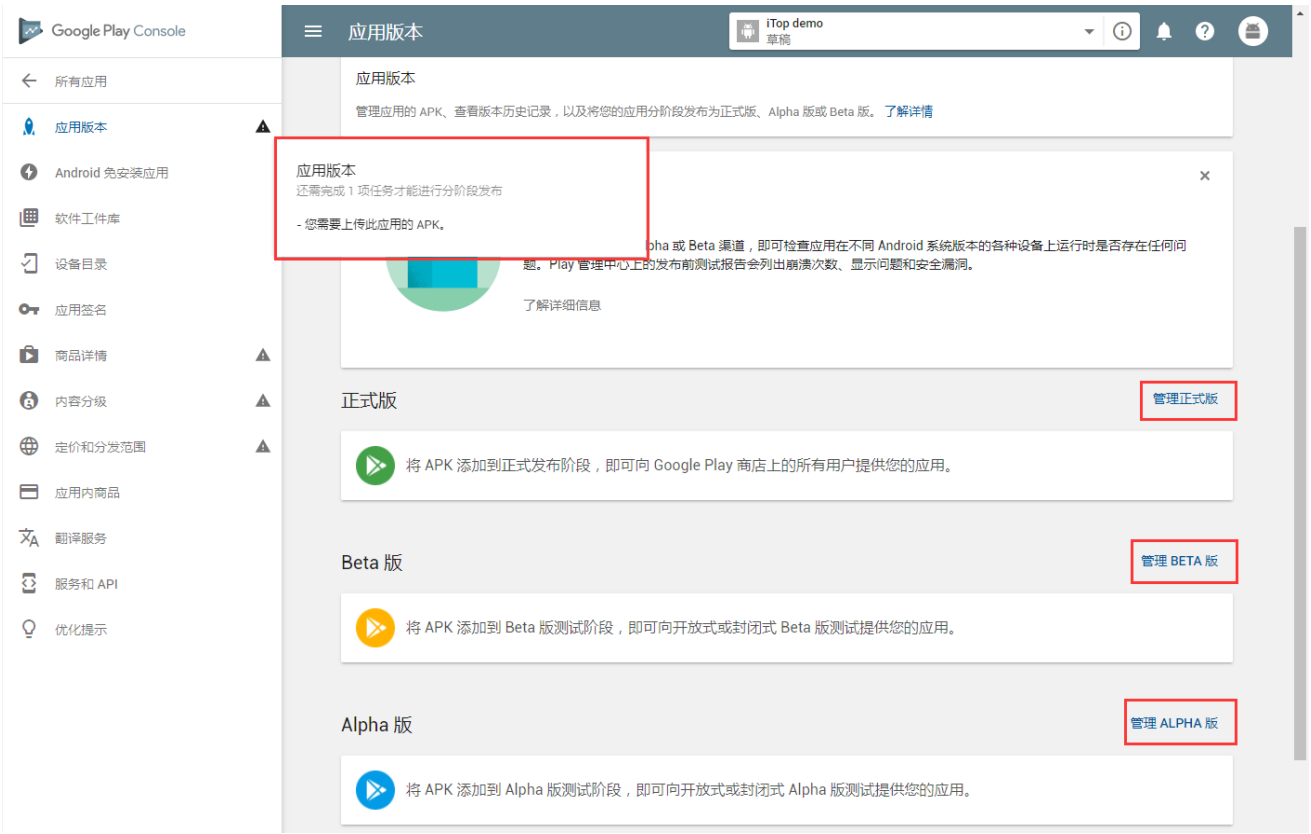
后即可创建成功。

创建成功后，该应用的详情页如下，有很多告警信息，只有一步步按照提示处理完这些信息后，才能达到发布状态。



2> 配置应用版本

请按告警提示处理：



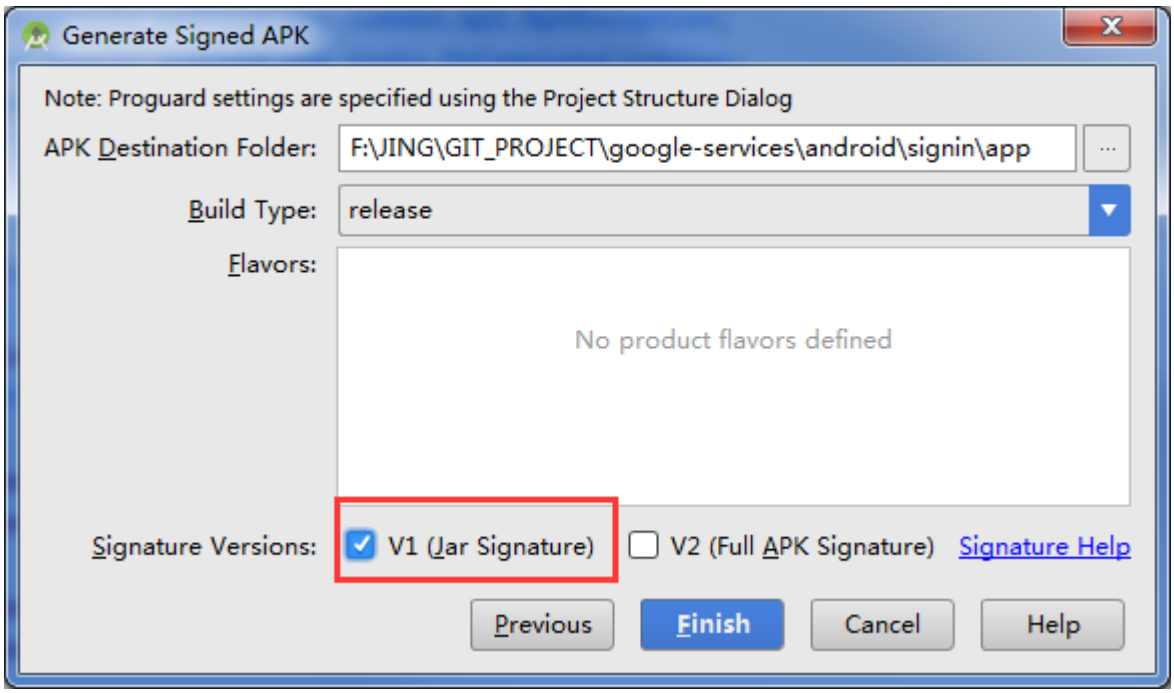
此处需要上传应用的 APK 包，Google Play 会解压该apk，作一些操作，比如：

- 获取包名，确认该包名是否可用，比如不能 com.google 开头。

- 分析权限，如果有特殊权限，可能需要填写隐私政策。

  > `android.permission.READ_PHONE_STATE` 如果 APK 中申明了此权限，此处上传应用通过后，会提示需要设置隐私政策，但没有明确标注设置的路径在哪里，这个设置是在`商品详情`标签页的最后一步设置！

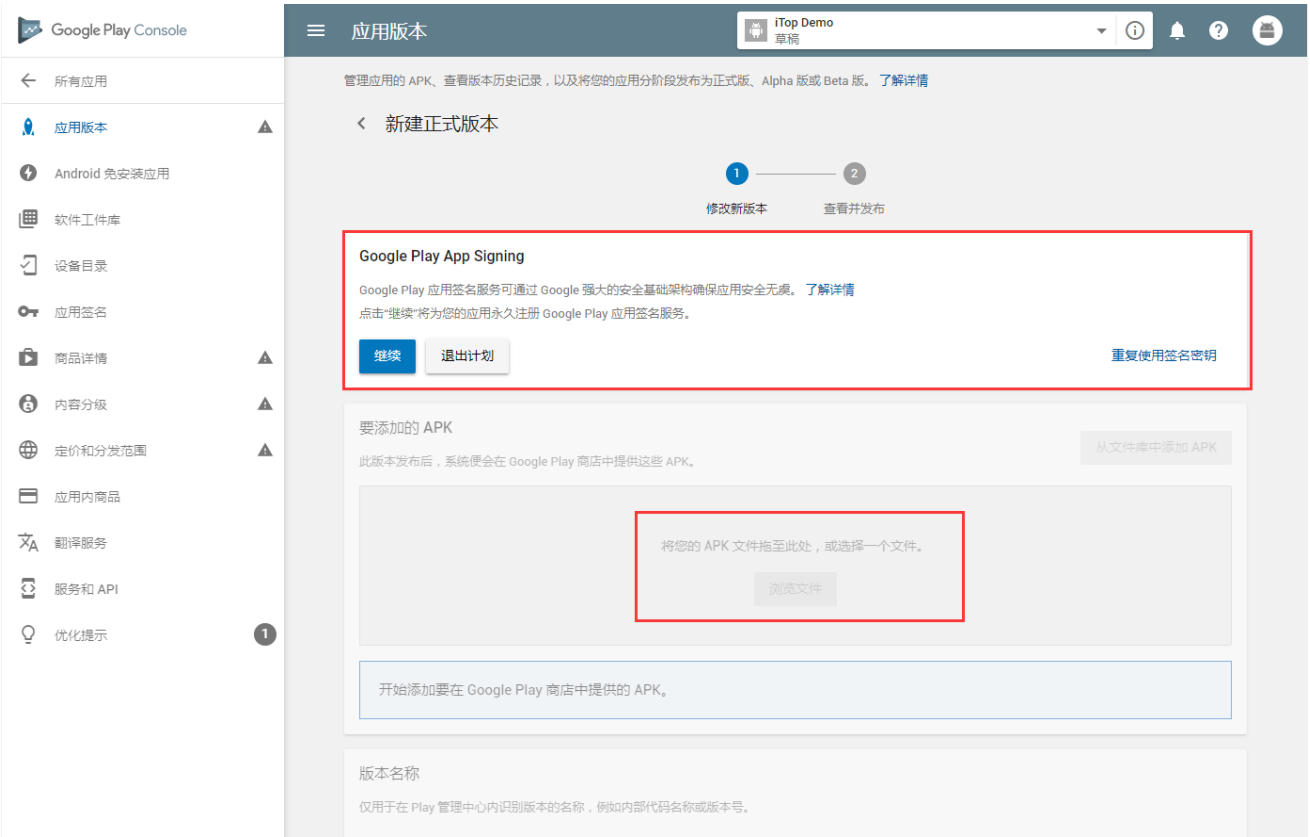
- 分析签名。

  > 签名版本中必须勾选 V1
  >
  > 
  >

根据需要上传相应的发布版本：

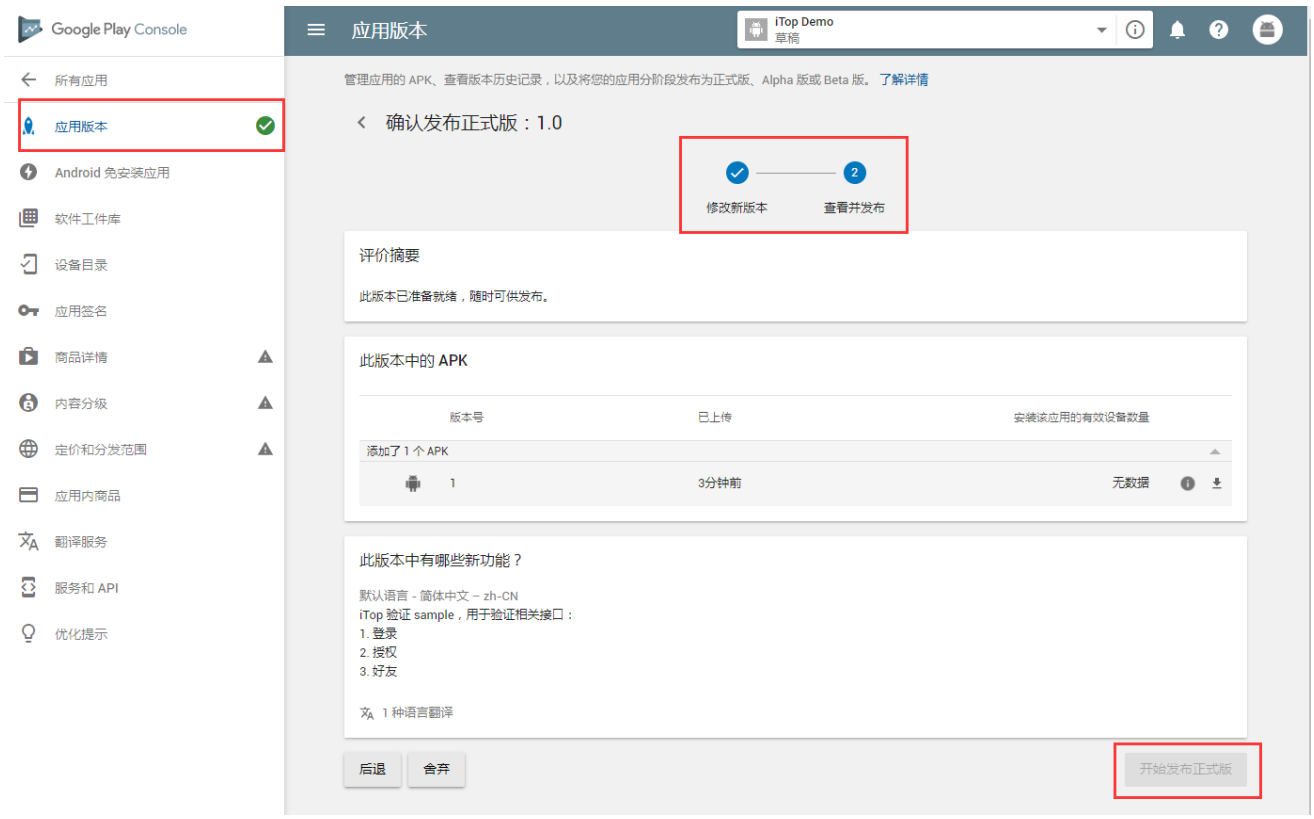
- Alpha 版 - 内测版本，有限用户的体验测试版本；
- Beta 版 - 公测版本，针对所有用户公开的测试版本，做过一些修改，成功正式版的候选版本；
- 正式版

点击`管理正式版`后，再点击`创建版本`创建第一个正式版本



> 相关说明：
>
> - Google Play App Signing - 如果启用了该配置的应用，只能使用唯一的签名，不能与账户名下其它 APK 共用签名。
> - 要添加的 APK - 上传 APK 后获取包名签名信息。
> - 版本名称
> - 此版本中还有哪些新功能 - 即版本说明，在 Google Play 商店应用详情上能看到。

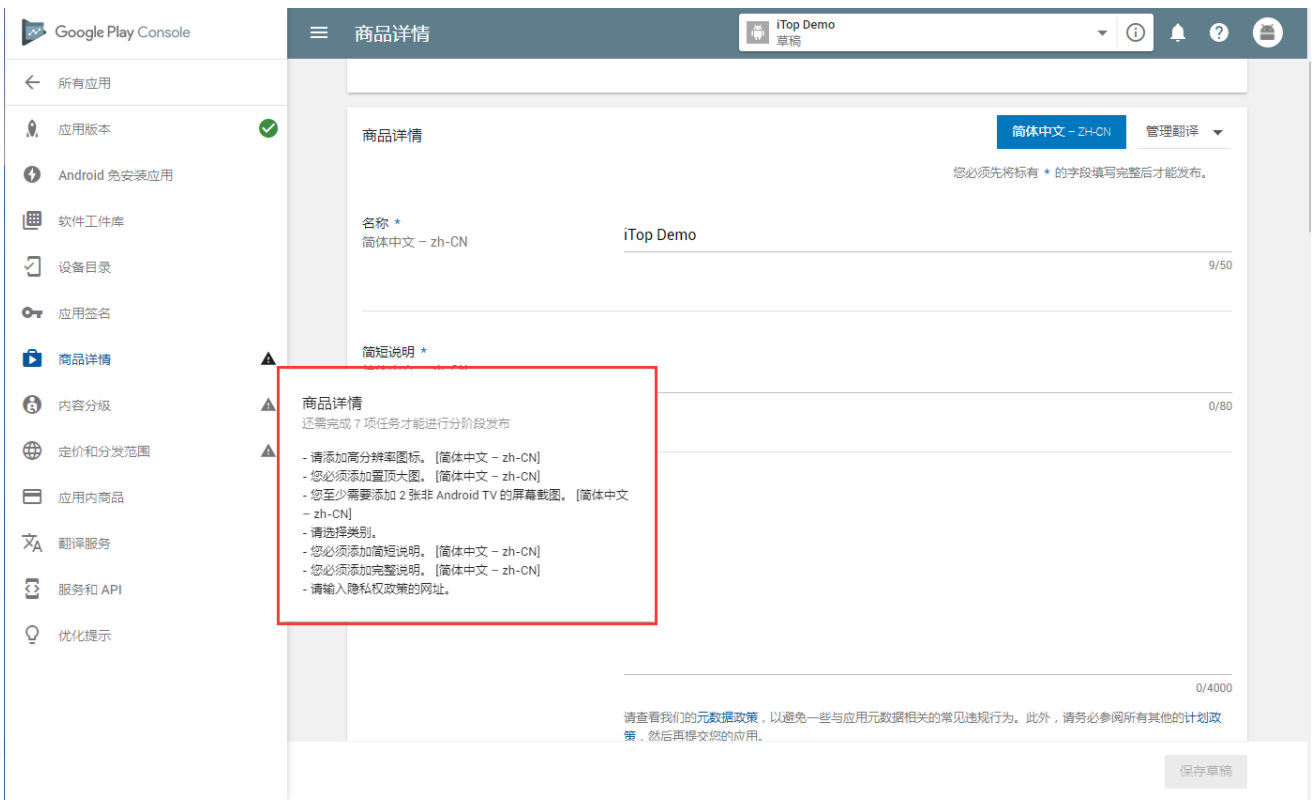
所有数据填写完成后，点击保存，上传应用成功。



此时 应用版本 告警已经处理完毕，因为还有告警未处理完，此时 开始发布正式版 按钮不可操作。

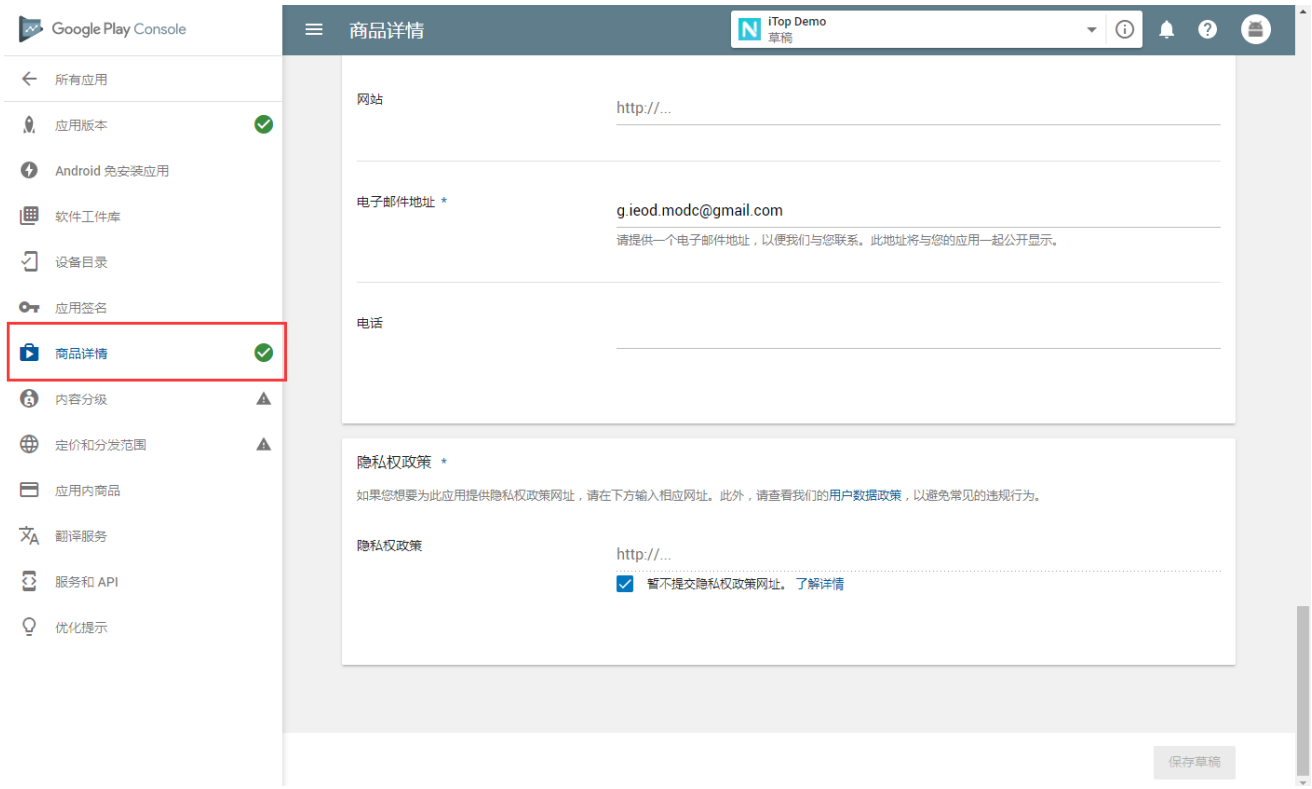
3> 配置 商品详情

请按告警提示处理：



该版本没有需要特别注意的地方，处理所有带*号的内容即可。

隐私权政策部分需要提供一个托管网站，具体设置请稳步隐私权政策，也可以暂时忽略。



4> 配置 内容分级

请按告警提示处理：

点击 继续 会要求填写一调查问卷，保存后点击 判断分级 即可查看结果，如对分级结果有异议可以返回重新填写，如没有问题，点击 确定分级 即完成配置。

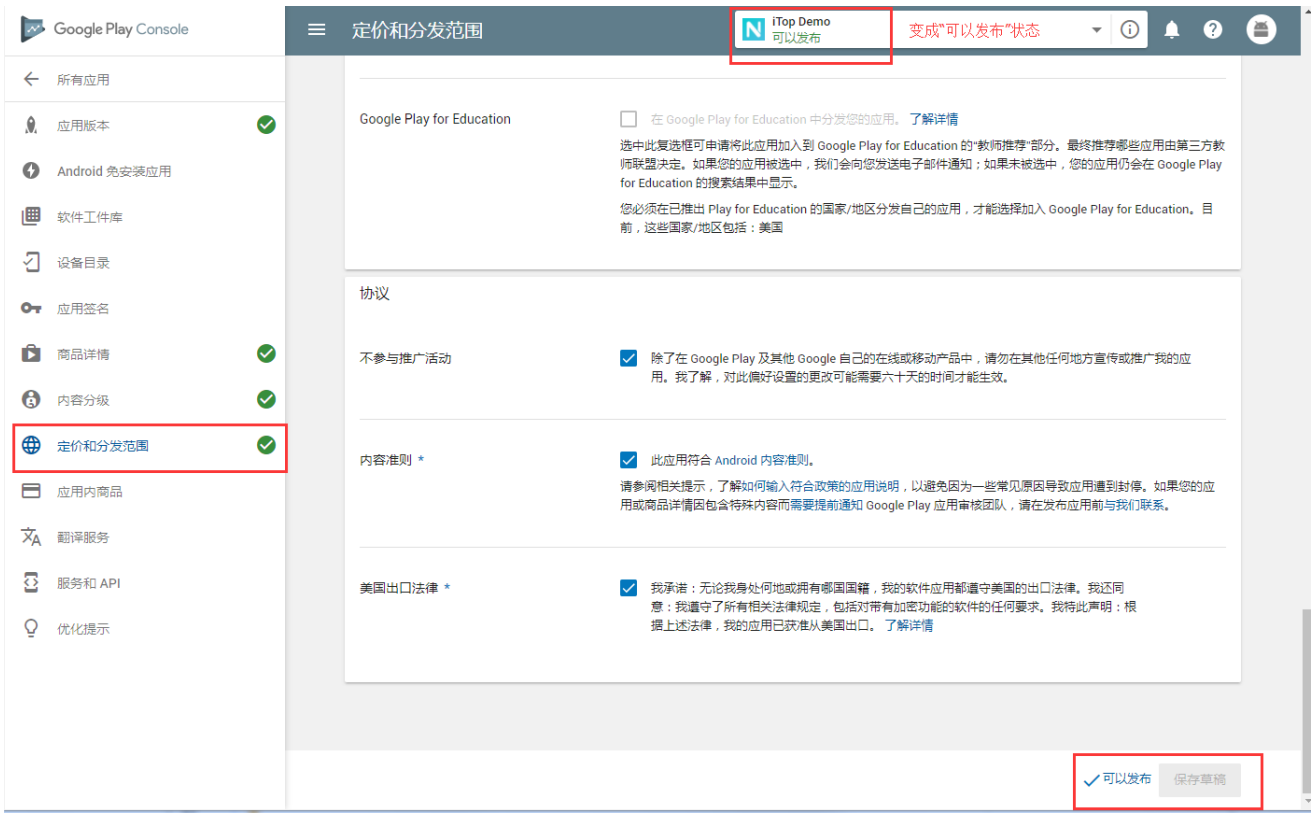
5> 配置定价和分发范围

请按告警提示处理：

此处需要处理：

- 应用是否收费
- 应用发布的地区
- 面向儿童设置
- 广告设置
- 一些法律条款

处理完点击保存，此时会提示应用处于可以发布的状态。



点击`可以发布`会出现弹窗：

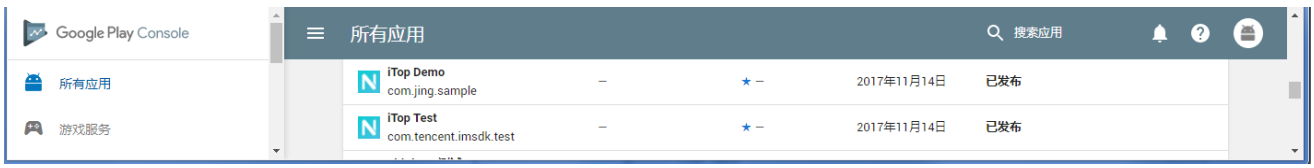
点击`管理版本`会跳转**应用版本**标签中，依次点击`管理正式版`->`修改版本`，在页面最下面有一个`查看`按钮

点击 查看 按钮查看待发布的版本。

点击 开始发布正式版 后弹窗确认，开始发布；

发布需要耗费半个小时，此时状态更新为`正在等待发布`，发布成功后，在账户的`所有应用`便签页能看到应用状态。

成功发布后状态改为：



2. 创建游戏服务

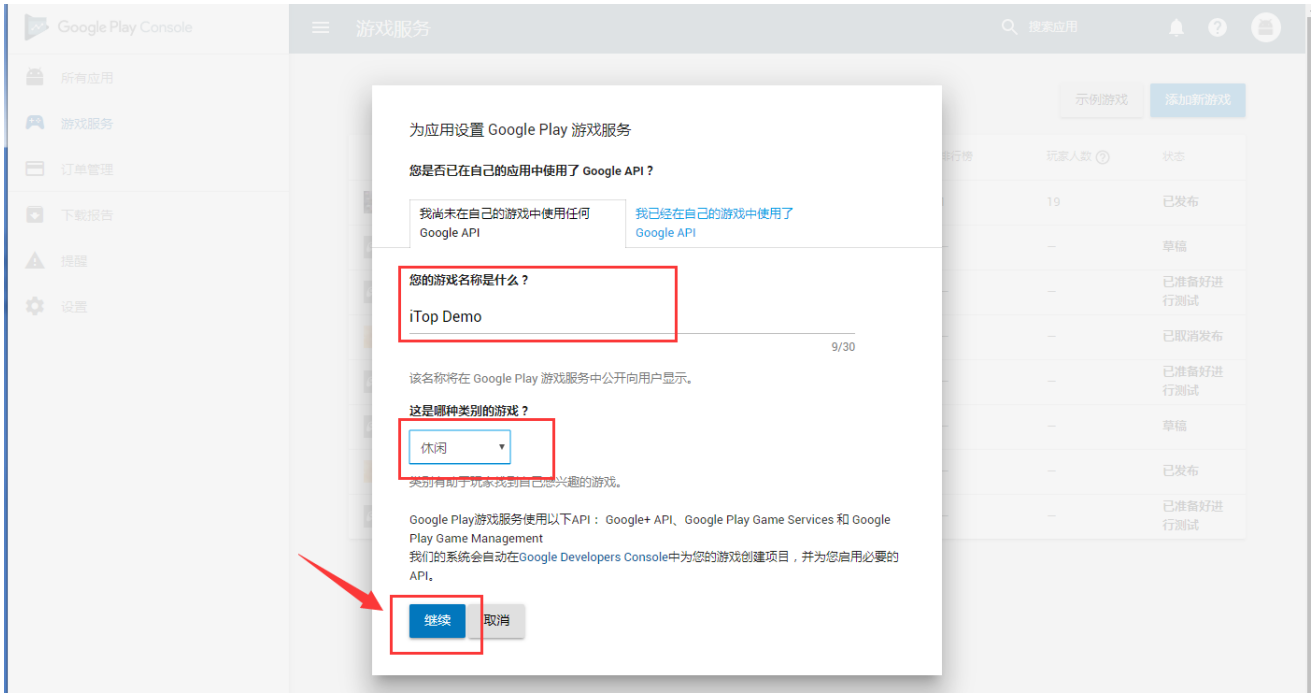
如果上传的 APK 应用需要接入 Google Auth / Google Game 等服务的话，就需要创建`游戏服务`。

> 游戏服务 与 应用是一对多的关系，即一个游戏服务可以关联N个应用。

1> 创建服务

在`Google Play 开发者控制台`主页上点击左上角标签`游戏服务`会列出当前账户下创建的所有游戏服务，如果没有，可以点击`创建游戏`来创建服务。

输入游戏名称、类型后，点击`继续`。

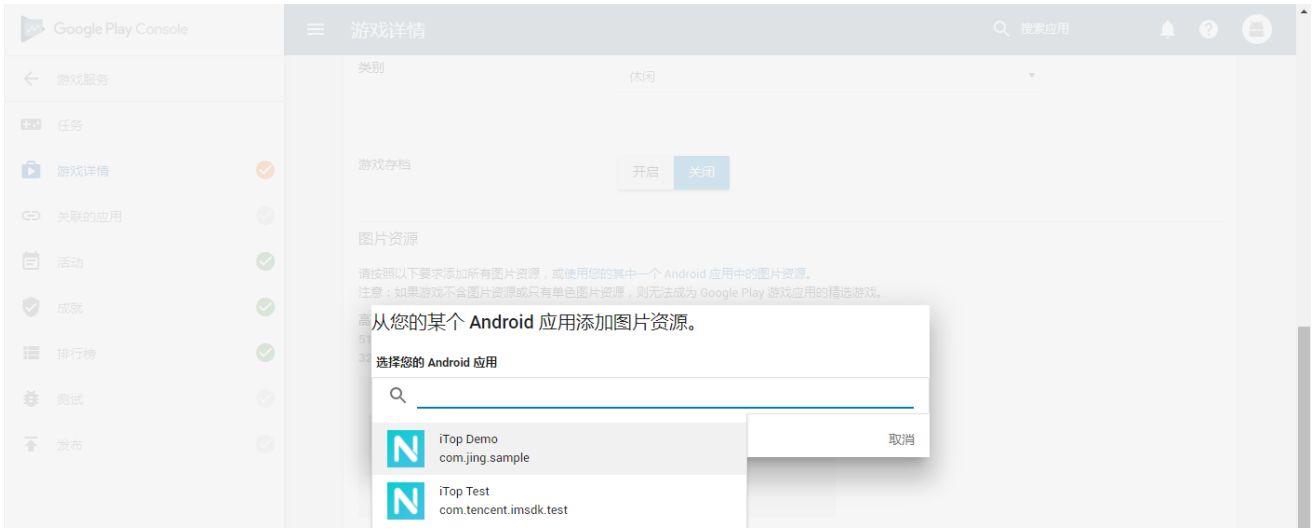


初始服务创建成功后，如下图。

2> 配置游戏详情

此处主要是配置图片资源，可以通过当前账户下面已有的 游戏服务中获取图片资源，当然也可以重新上传。

点击`使用您的其中一个 Android 应用中的图片资`后弹窗，通过包名选择对应的应用即可。

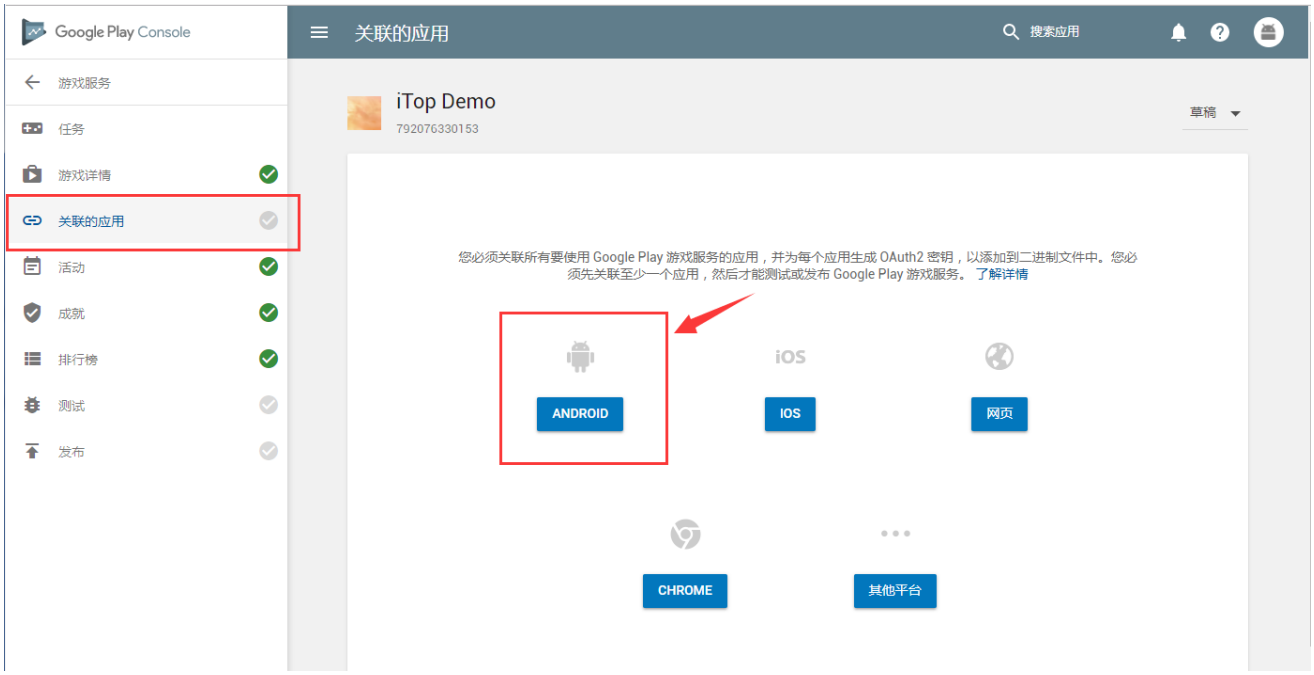


选择成功后，保存。

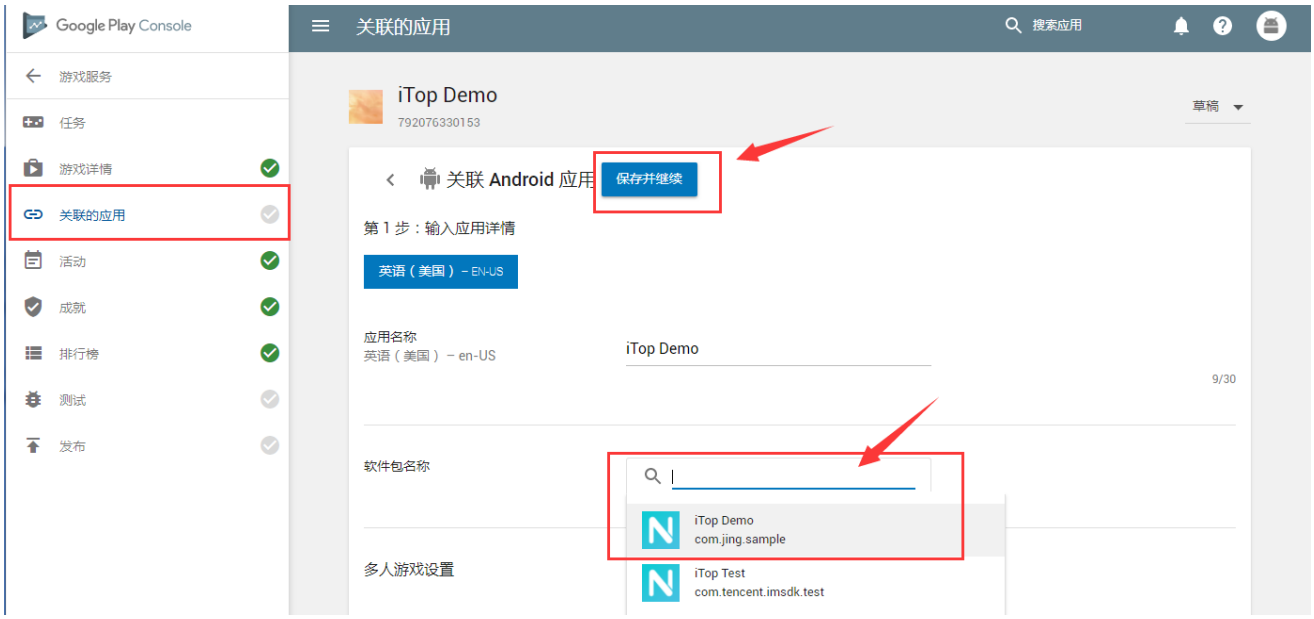
3> 配置`关联的应用`

游戏服务需要关联相应的应用，才可以达到测试状态。

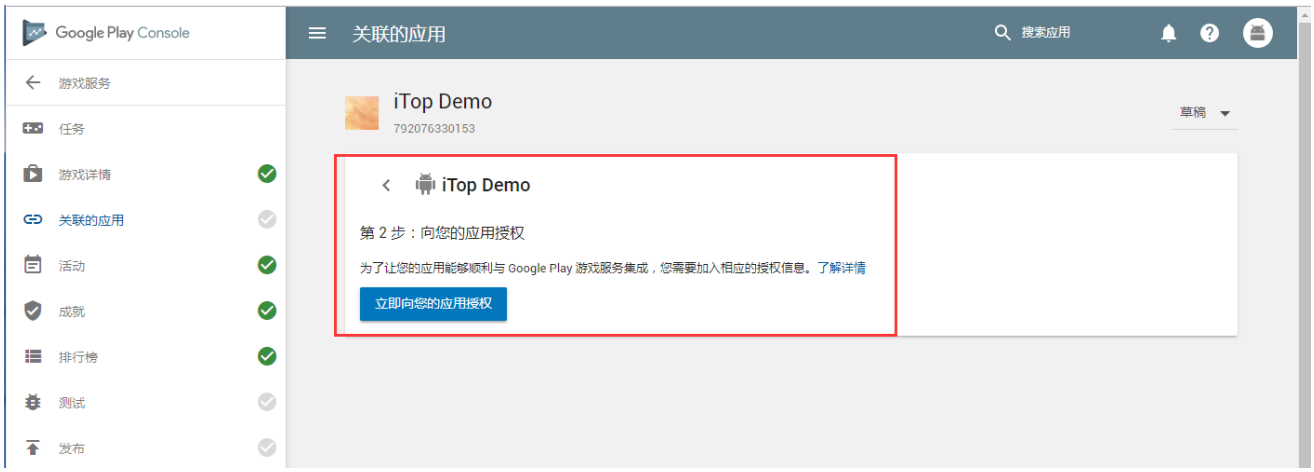
点击`关联的应用`，然后点击`ANDROID`添加应用。



通过 软件包名称 关联相应的应用，然后点击 保存并继续 完成关联。

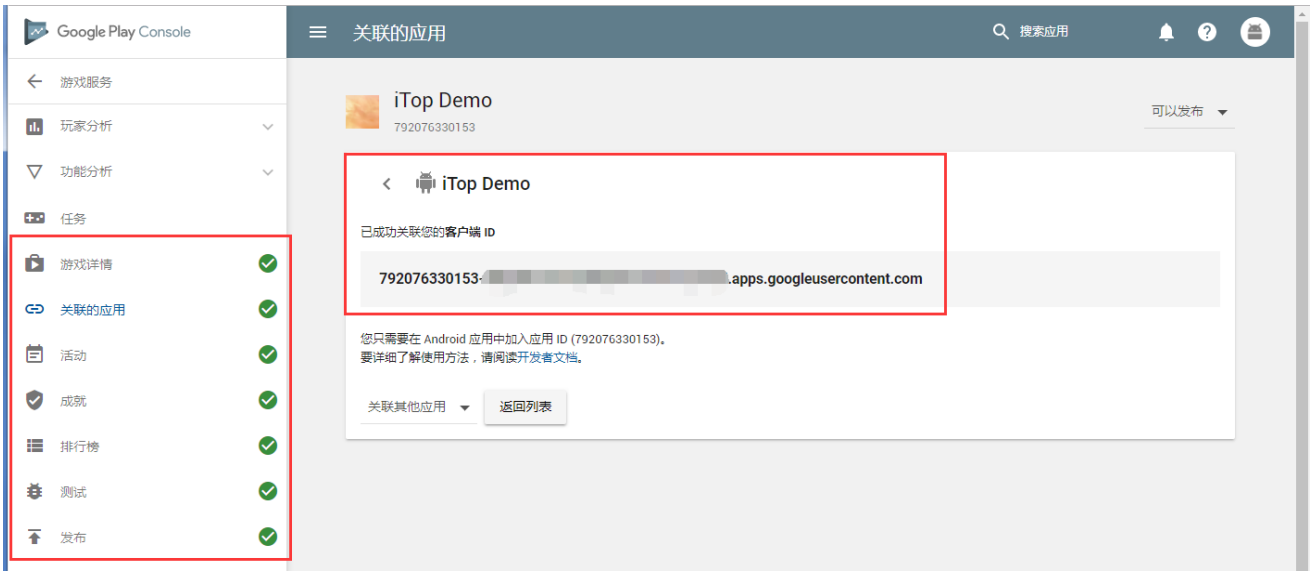


关联成功后，需要进行 应用授权 。



点击授权，会出现弹窗，确认即可。

完成配置后，游戏服务创建完成，并处于可测试及发布状态。



4> 添加测试权限

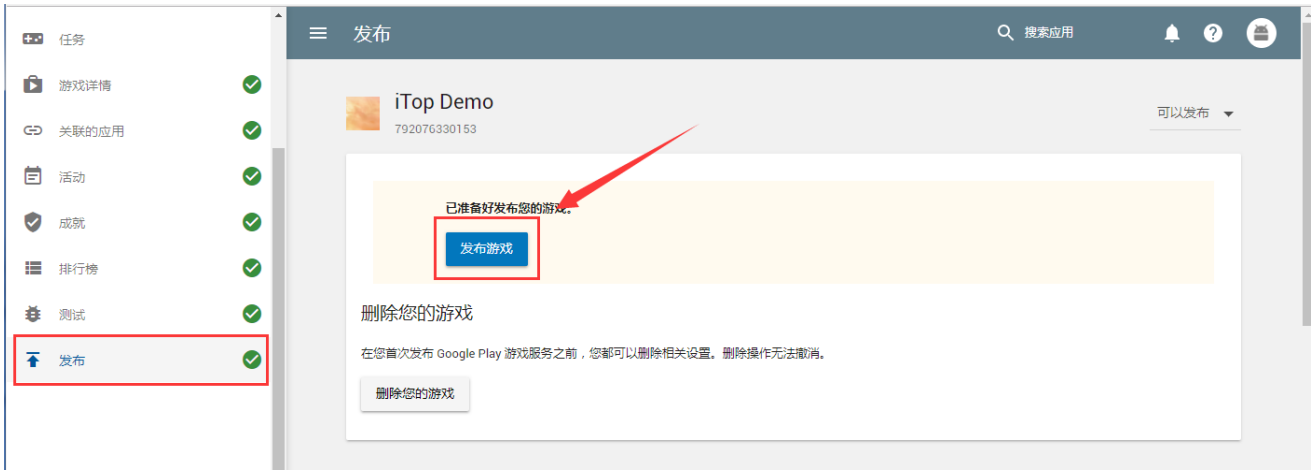
进入 测试 标签，Google Play 游戏服务已经是可测试状态。

在游戏服务还没有发布的情况下，如果想验证游戏服务，可以通过添加测试账号来解决。

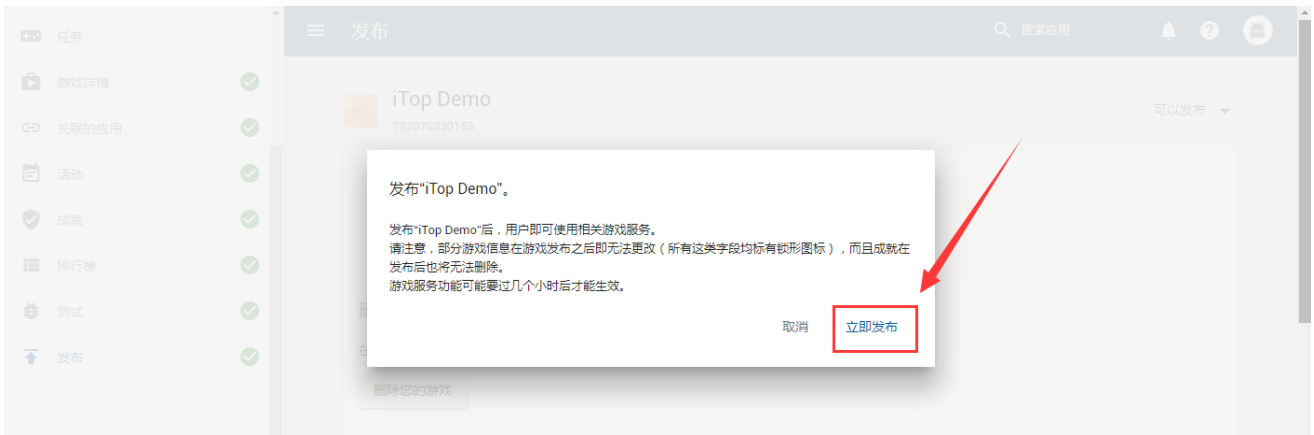
点击`添加测试人员`，输入相应的 Google 账户即可。

5> 发布游戏

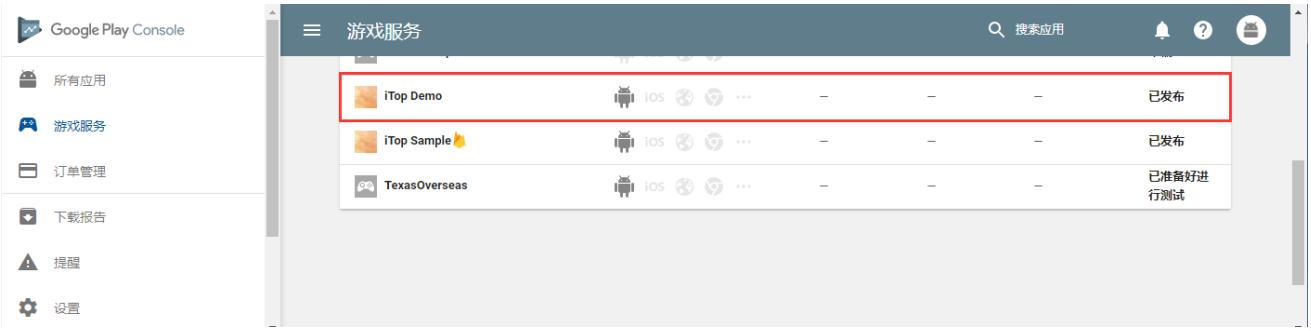
进入`发布`标签，点击发布游戏。



弹窗显示确认框。



返回到`游戏服务`标签中查看服务状态。



3. 创建 Auth 授权

发布的应用在使用游戏服务时，比如登录，排行榜等等，需要鉴权，防止非法请求。

在`游戏服务`标签页选择相应的游戏服务，点击进去会到达`游戏详情`页，找到`API 控制台项目`。

会有`此游戏已与名为xxx的 API 控制台项目关联`，点击`xxx`进行跳转。

> 注意这里的关联操作是在创建游戏服务时自动关联的

进去之后，发现已经存在了一个 Android 类型的 OAuth 2.0 客户端 ID，这个是用于 Android 客户端鉴权的。

它是在什么时候创建的：

- 上传应用的时候，获取到了应用签名 + 应用包名
- 使用`keytool -exportcert -keystore path-to-debug-or-production-keystore -list -v`获取 SHA1

除了默认创建这个，还可以手动创建 ID。

1> 创建客户端ID Android

点击上面截图中的`创建凭据`

选择`android`

配置下面信息保存即可：

- 签名证书的指纹

  针对上传的应用的 APK 签名，使用命令获取 SHA-1 证书指纹

  ```
  keytool -exportcert -keystore 你的签名文件 -list -v
  ```

```
E:\JING\AS-keystore
λ keytool -exportcert -keystore googleplay.jks -list -v
输入密钥库口令:

密钥库类型: JKS
密钥库提供方: SUN

您的密钥库包含 1 个条目

别名: googleplay
创建日期: 2017-11-10
条目类型: PrivateKeyEntry
证书链长度: 1
证书[1]:
所有者: CN=jing, OU=ieg, O=tx, L=china, ST=shenzhen, C=861
发布者: CN=jing, OU=ieg, O=tx, L=china, ST=shenzhen, C=861
序列号: 3c3cd4d7
有效期开始日期: Fri Nov 10 14:16:16 CST 2017, 截止日期: Tue Nov 04 14:16:16 CST 2042
证书指纹:
         MD5:
         SHA1: 07:A6:[redacted]44:7E:B6
         SHA256:
         版本: 3

扩展:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: FF AE 99 F8 23 B0 08 BD   9D DE 3E 42 FF E0 E9 8F  ....#.....>B....
0010: F7 7D 61 51                                        ..aQ
]
]
```

- 软件包名称

> 凭据中同一个Android类型的客户端ID中，不允许有相同的 SHA-1 指纹存在

2> 创建客户端ID 网页应用

在创建客户端ID中，选择 网页应用

点击创建即可。

3> iTop 后台鉴权

在上面配置创建完成以后，该游戏服务存在两种客户端ID

- Android id，在用户接入 Google auth 登录的时候需要校验
- 网页ID，则是游戏后台与GMS之间的校验

iTOP后台要做什么：

- Android 客户端 ID 中的 SHA-1指纹 - 主要用于校验当前请求是否合法，非法会返回 `-905`
- 网页ID - 主要是后台用于与GMS之间校验登录数据合法性，非法会返回`-213`