热更新就是为了在不用重新下载包体的情况下，在线更新游戏中的一些资源，比如活动资源和热补丁等

///////正文

热更新主要采用一个xlua热更新

热更新流程主要是 游戏启动后，判断p目录是否存在，如果不存在，就创建一个，把s目录的版本号、资源清单拷贝到p目录，然后链接资源服务器，下载服务器上的版本号，对比版本号 ，如果版本号不同，就说明有新版本，然后就是下载资源清单，对比资源清单上文件的名称和md5码，如果本地没有或者md5码不同，就说明是新的资源或者更改的资源就进行下载或替换，然后就是也可以打包的时候只把需要更新的资源打包，放到资源服务器上，就是打一个增量包，然后客户端直接就是下载就行，如果版本号差距过多就让玩家去重新下载游戏的主包体，文件下载完毕之后，需要储存一下新的版本号和资源清单

然后下载文件这里还做了一个断点续传的处理，就是热更下载文件的过程中，断网了，联网后在断掉的位置继续下载，在他断开的时候socket会报错，然后就可以记录一下断掉的文件名写入json存到本地，然后下载的时候判断一下这个文件在不在，如果在的话就从这个位置开始下载，不在就直接从头下载。但是这个需要考虑数据的一致性和下载的有效性，确保在断点续传时不会导致文件损坏或下载数据的重复

热更新的文件删除方案 一个是，遍历服务器的资源清单和本地对比，本地没有的以及md5码和本地不一样的，需要更新。还有就是，也可以遍历本地的资源清单和服务器的对比，服务器没有的本地也没必要留着，但是一般不会去删除资源，比如王者荣耀还会有返厂皮肤的时候，如果用完就删 那到时候还得下载。所以一些游戏的客户端就是越来越大

市面上其他热更新的方案 有华佗热更，ILRunTime

华佗热更主要是把代码分离成非热更程序集和热更程序集；把热更程序集打成dll，作为资源以供下载更新，其他了解不多

打ab包这一块，我记得当时项目开会讨论的时候，

有人提了说所有资源打一个包，然后被否定了，这样的话，一个包全加载进去内存直接就满了，会很卡，影响用户体验。

然后第二个方案是，按照场景打包，公共资源打一个包，对这个资源进行一个引用计数，如果计数大于等于2就打公共包,需要注意的是有的模型上面会出现名称一样但是后缀名不一样，这样的话还是打一个包会报错 可以把名称一样后缀名不一样的打成一个包 或者 找美术改一下名称，然后除去公共资源外的资源，同一个场景的打一个包，这个主要是按照场景进行加载ab包，然后缓存中基本上就是2个场景的资源加上公共包的资源，2个场景是为了防止有玩家在两个场景之间来回切换，导致频繁加载资源消耗性能。

还有一个备用方案是分类打包，就是模型打一块，音频打一块，特效打一块等等，这种方案其实也是有缺点的，就是你会把好多当前用不到的资源也给加载进来，会造成资源的冗余，所以我们当时就选择了第二种按场景打包的那个方案

然后这些资源都会受到资源管理框架的一个统一管理

//////问题

AB包与Resources 的比较

AB包 Resources

1： 资源可以分布在多个包中 所有资源达打成一个打包

2： 储存的位置自定义灵活 必须放在Resources文件夹下

3： 压缩的方式灵活 全部压缩成二进制

4： 支持后期进行动态更新 打包后资源只读无法动态更改