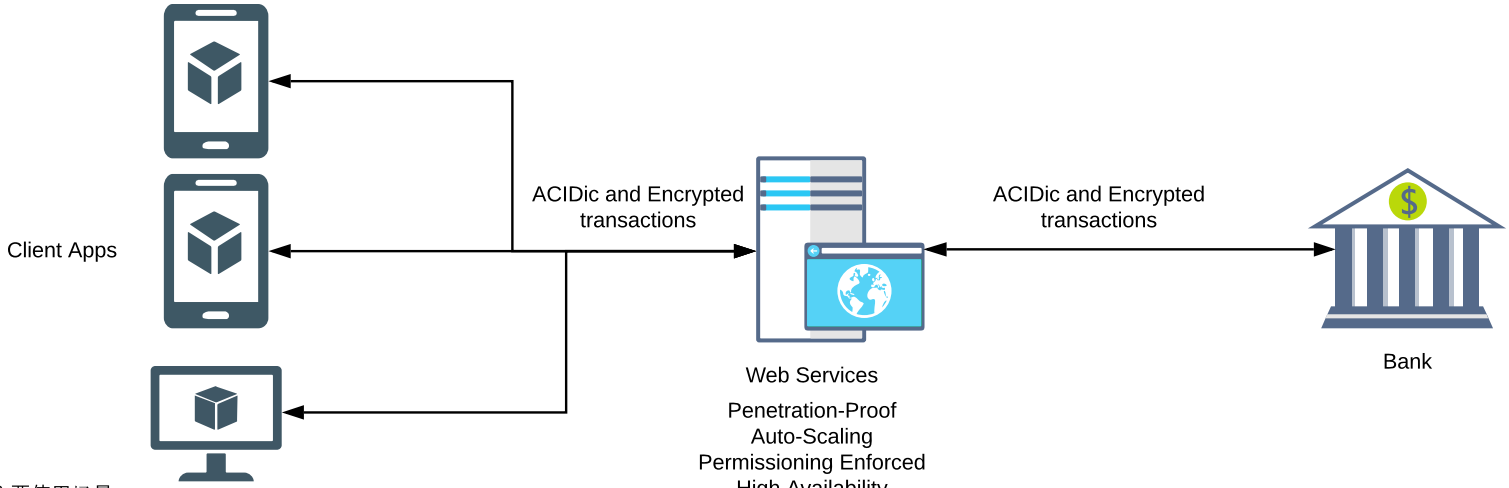

主要使用场景：
1. 客户通过客户端申请从银行转账余额到第三方支付账户

主要风险点：
1. 数据在储存和传输过程中的加密
2. Transaction需要满足ACID原则：
    A: Atomic: 一项含有多条Statements的Transaction，要么全部Statements都完成，要么没有任何一条Statement完成。
    C: Consistent: 确保Transactions完全符合数据库定义好的修改规则。
    I: Isolated: 多线程处理Transaction要确保最终的结果等价于单线程按顺序处理Transaction的结果。没有完成的Transaction不能被其他的Transaction读取或修改。
    D: Durable: 完成的Transaction必须永久保存，直到被合法删除或修改，不受死机影响，有抵抗恶意修改的能力。
3. 时效性。确保反应速度达到SLA要求。

测试方案大致思路：
1. 加密测试：
    A. 确保客户端SDK在用户操作时不被窃取信息
        - 试着黑进用户操作界面，读取用户个人信息，密码，等隐私信息。
    B. 确保客户端SDK在传输Transactions的路径上不被窃取信息
        - 试着截获用户的Transaction Request，检查是否加密，试图破解。
    C. 确保Transaction在业务系统里存储和流动时始终处于加密状态
        - 检查业务系统各个节点内，用户数据的加密状况
        - 检查KMS（数据密匙）的使用情况是否符合规则
        - 检查IAM（用户授权）的使用情况是否符合规则
    D. 确保Transaction在发给银行的路径上被加密
    E. 确保返程Transaction (Bank -> Web Service -> Clients)的加密情况。
2. ACID测试：
    A: 让Transaction包含多条读取，修改，删除的有效指令，和一条无效指令，确保最终整个Transaction失败，没有任何指令被执行。
    C: 试图执行非法指令，确定Transaction失败 。
    I: 多线程修改同一条用户信息（比如五次增加，三次减少账户数额），确保最终数额正确。
    D:
        - 在用户发送完成Transaction得到确认后关闭客户端，确保Transaction到达Web Service
        - 在处理用户Transaction的过程中关闭若干Web Service内的处理节点，再恢复，确保数据没有丢失
        - 熔断测试
        - 在Web Service完成处理用户Transaction并且发送出去以后，关闭Web Service服务的若干节点，确保没有数据丢失
        - 数据备份测试
        - 在Web Service正在部署新代码时测试Transaction，确保没有数据丢失，被修改。
        - Penetration Test
3. 时效性测试：
    1. 模拟极大量的用户在短时间内同时发送Transaction，确保Auto-Scaling 能够及时Scale Up, 能够及时处理所有信息，满足SLA需求
    2. 模拟从世界各地发送请求，确保服务反应速度满足SLA需求