
# HW 3. Software Architecture Style Survey

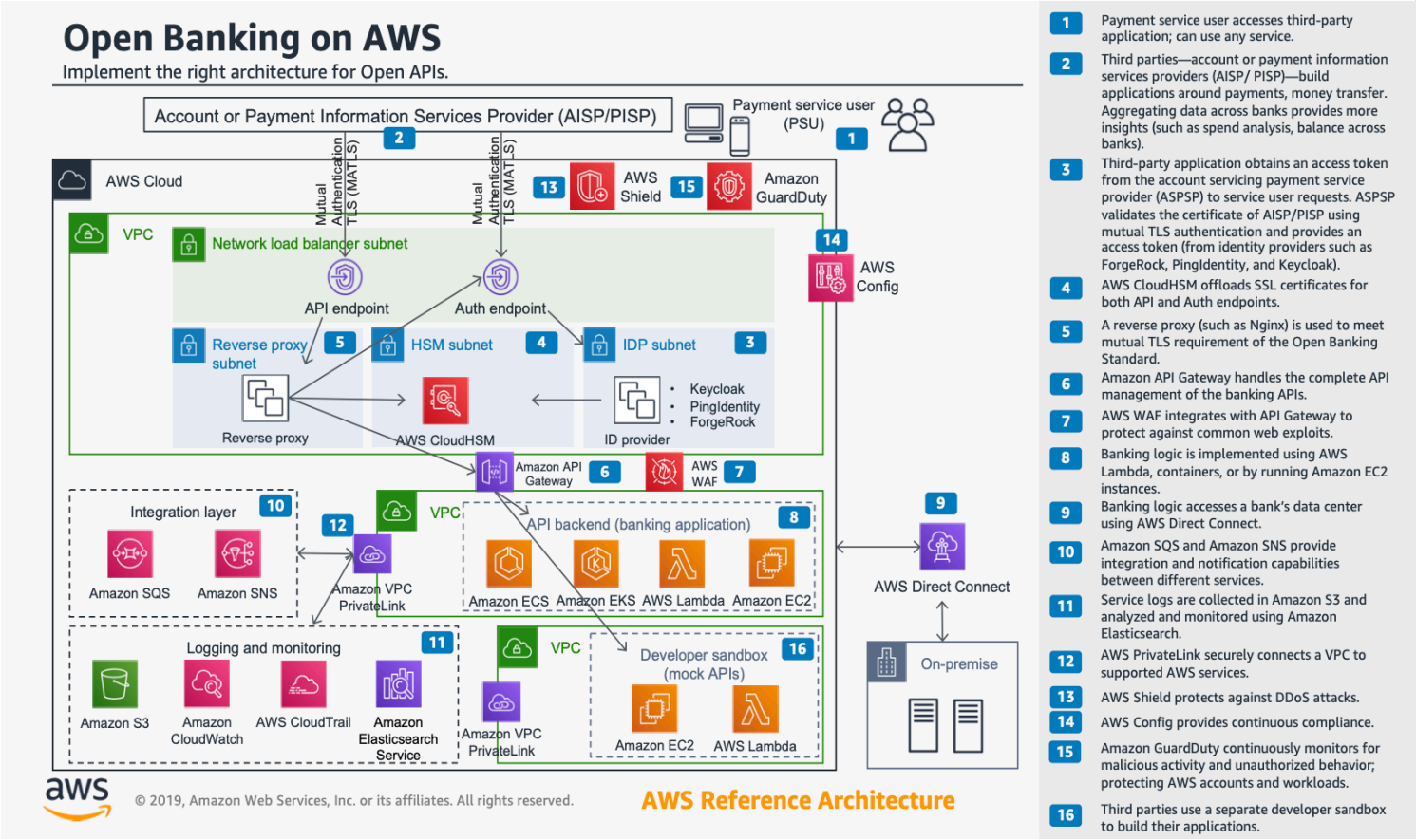
## Open Banking on AWS

소프트웨어전공
20152791
강길웅

# I. 대상 시스템

Open Banking on AWS. AWS에서 제공하는 아키텍처 다이어그램 중 오픈뱅킹에 대한 서비스를 제공하는 시스템이다.

## I.1. Open Banking on AWS




**[그림 1-1] 아키텍처 다이어그램**[1]

### I.1.A. 대상 시스템에 대한 조사

해당 시스템은, 실제로 제공하는 서비스가 아닌 AWS에서 AWS의 솔루션을 이용하는 시스템에 대한 예시로 만들어 놓은 아키텍처이다. 오픈 뱅킹에 대한 서비스를 제공하는 시스템이다.

오픈뱅킹은 은행의 송금·결제망을 표준화시키고 개방해서 하나의 애플리케이션으로 모든 은행의 계좌 조회, 결제, 송금 등의 금융 서비스가 이뤄지는 계좌를 조회하고 송금할 수 있는 서비스를 말한다.(네이버 시사상식사전(2020))

### I.1.B. 참고문헌

AWS 설명서

[1] 출처 - https://d1.awsstatic.com/architecture-diagrams/ArchitectureDiagrams/open-banking-on-aws.pdf?did=wp_card&trk=wp_card

# I.2. 시스템 개요 및 특성 분석

## I.2.A. 개요

아키텍처 전체 시스템 구성에서는 복합적 아키텍처 스타일이 나타난다. 사용자는 모바일 어플이나 웹과 같은 플랫폼을 통해 서비스를 제공받고 이 형태는 Client-Server아키텍처 스타일이 나타나며, 시스템 내부는 책임에 따라 몇가지 레이어로 나뉜다. 또한 프록시를 이용해서 API backend 로 접근하는 형태인 프록시 아키텍처 스타일이 나타난다.
금융 시스템이기 때문에 보안 강도가 높은 시스템이라는 특성이 나타나는데, 인증과 백엔드 등 레이어를 다른 VPC를 두어 분리 시킨 형태이며, API Gateway를 통해서 통신한다. 인증 레이어 또한 각 목적에 따라 보안을 위해 Subnet을 모두 나눈 형태가 특징이다. 데이터 베이스의 경우도 Cloud 플랫폼 기반 시스템을 구성함에도 당 시스템 내에 포함 시키는 것이 아닌, 외부에 구성된 형태를 보여 강도 높은 보안을 위한 설계가 되었음을 알 수 있고, AWS Shield, Amazon GuardDuty 등 보안 솔루션도 눈에 띈다.

## I.2.B.Client-Server 아키텍처 스타일

3티어의 배치 형태를 보인다. 유저에게 UI를 제공하는 클라이언트 노드, 데이터를 받아 처리하는 백엔드 서버 노드, 데이터를 저장하는 데이터베이스 서버 노드 형태를 가지고 있다. 그 중 클라이언트 노드와 데이터베이스 서버 노드는 이미 개발되었음을 전제로 하고 현재 아키텍처 구성도는 백엔드 서버에 대한 아키텍처 설계가 이루어져 있다.
클라이언트 노드로부터 요청을 받아 백엔드 노드에서 보안 인증과 데이터 처리를 수행한 뒤, 데이터베이스 노드에서 데이터를 저장,조회 등 작업을 수행하고 이를 받아 클라이언트 노드에 다시 보내주는 형식이다.

## I.2.C.레이어 아키텍처 스타일

시스템 내부를 크게 책임에 따라 5개의 레이어로 구분하였다. ID발급이나 인증 검증과 같은 처리를 위한 보안/인증 레이어, 진행 알림이나 진행 정도 등을 파악 하기 위한 Integration 레이어, 시스템 로그나 모니터링을 위한 로그 레이어, 요청 처리를 위한 API 백엔드, 타행 요청 처리를 위한 타행 레이어 이다. 이중, 보안/인증 레이어는 다시 로드밸런서, 프록시, HSM, ID provider 4개의 레이어로 쪼개지는 형태를 보인다.
각 레이어는 분리되어 있으며, Gateway 및 VPC private link로 통신하게 된다.

## I.2.D. 프록시 아키텍처 스타일

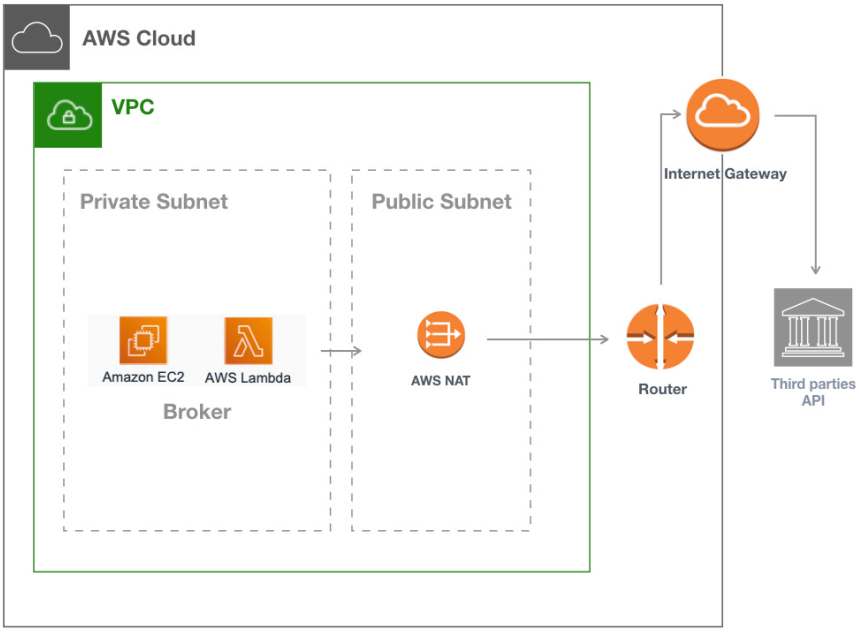
보안/인증 관련 데이터 값을 보안/인증 레이어 밖을 벗어나도록 설계하면 안된다. 그렇다고 API 백엔드나 타행 요청 처리 API를 보안/인증 레이어 내부에 두기에는 너무 무겁고 책임에 맞지 않는다. 때문에 보안/인증 레이어 내부에 프록시를 두어 외부에 존재하는 백엔드 API를 이용 할 수 있도록 하고, 레이어를 분리 시킬 수 있다.

# II. 재 해석

## II.1. 타행 레이어 어플리케이션 빌드에 관해

현재 타행 처리는, 해당 어플리케이션을 직접 빌드하여 처리하도록 구성되어있다. 해당 방식은 타행 수만큼 어플리케이션을 빌드해야 하고, 각 은행의 어플리케이션을 빌드하는 것이므로 타행이 어플리케이션 업데이트나 변경 시 직접 빌드의 변경을 해야 하기 때문에 유지 보수가 쉽지 않다. 또한 은행의 추가나 확장 등에 대해 어플리케이션을 추가로 빌드해야 한다는 문제가 있다.

## II.2. 타행 레이어 개선안 - 브로커 아키텍처 스타일 적용




**[그림 2-1] 타행 레이어 - 브로커 아키텍처 스타일**

### II.2.A. 내용

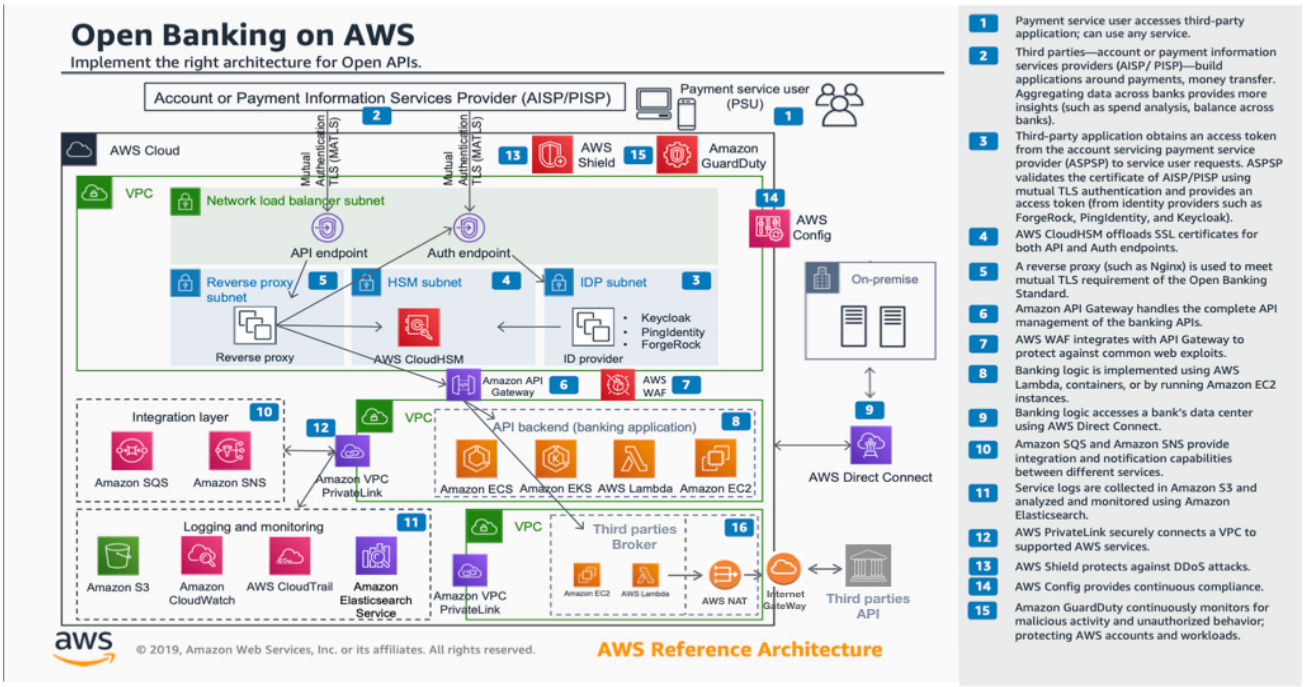
타행 어플리케이션 빌드가 아닌 API연동 방식을 브로커 아키텍처 스타일 기반으로 구성한다. 타행 API를 이용하여 처리하므로 해당 API의 정보만 가지고 있으면 되므로, 시스템의 부하와 무게를 줄일 수 있고, 처리 속도 또한 빌드 과정 없이 처리되기 때문에 성능의 개선이 이루어진다. 또한 타행의 시스템 변경이나 업데이트가 이

루어져도 별도의 수정을 할 필요가 없기 때문에 유지 보수에 유리하다.
브로커가 타행 API에 대한 엔드포인트 및 정보를 등록하여 이용한다. 브로커에게 요청이 들어온 경우, 해당 요청을 파악하여 적절한 타행 API를 호출, 값을 처리하고 결과 값을 받아 사용자에게 제공한다.

### II.2.B. 주의점 및 한계점

외부 API를 이용하는 것으로 변경되기 때문에 네트워크 통신이 포함되게 되고 이는 보안에 취약하게 될 우려가 존재한다. 외부와의 접촉을 막기 위해 반드시 브로커는 Private Subnet 안에 두고 외부 통신은 NAT를 통해서만 하도록 한다. 라우터 및 게이트웨이의 보안 규칙을 엄격하게 적용하고 TSL와 같은 인증 절차를 거치도록 하여 보안상의 이슈가 생기지 않도록 관리한다.
네트워크 통신이 포함되므로 값의 소실과 같은 문제와 보안상의 이슈가 한계점이다.

## II.3. 개선한 아키텍처 다이어그램




[그림 2-2] 개선한 오픈뱅킹 아키텍처 다이어그램

# III. 참고문헌

AWS 기반 오픈 뱅킹(2019). AWS 참조 아키텍쳐 다이어그램 ,
https://d1.awsstatic.com/architecture-diagrams/ArchitectureDiagrams/open-banking-on-aws.pdf?did=wp_card&trk=wp_card

AWS 설명서(2020). AWS 설명서,
https://docs.aws.amazon.com/index.html?nc2=h_ql_doc_do_v

네이버 시사상식사전(2020). 오픈뱅킹 검색 결과,
https://terms.naver.com/entry.nhn?docId=5779672&cid=43667&categoryId=43667