Intro to RPC

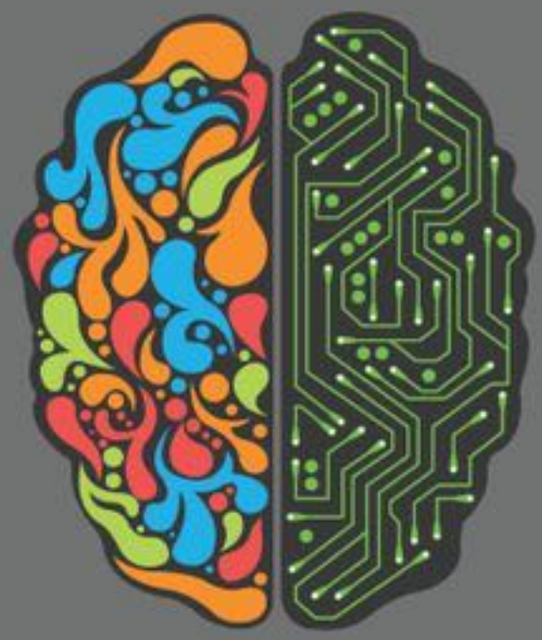
Apache Thrift

appkr(김주원)

2017년 3월

modern
php user group

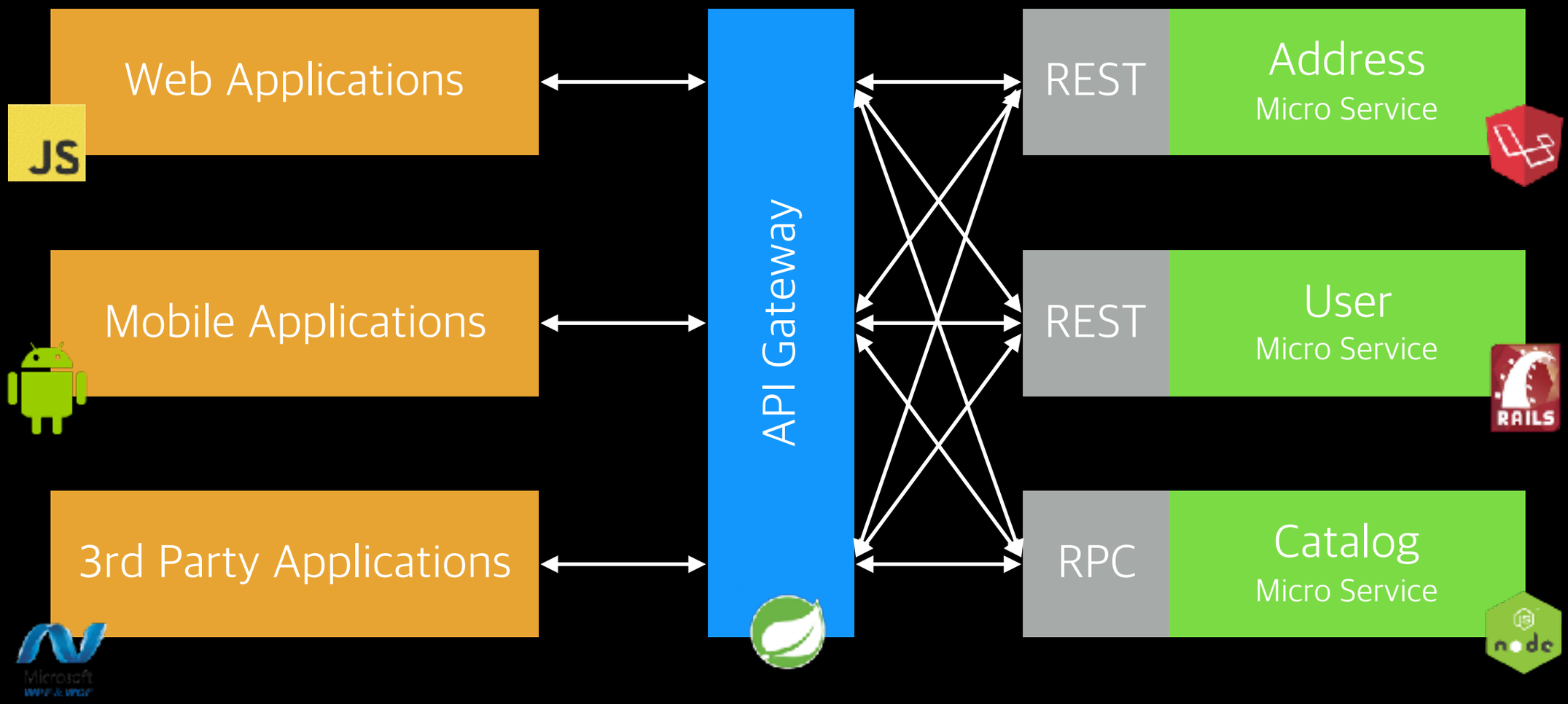
“It's all about Web APIs(==Data)”



UI

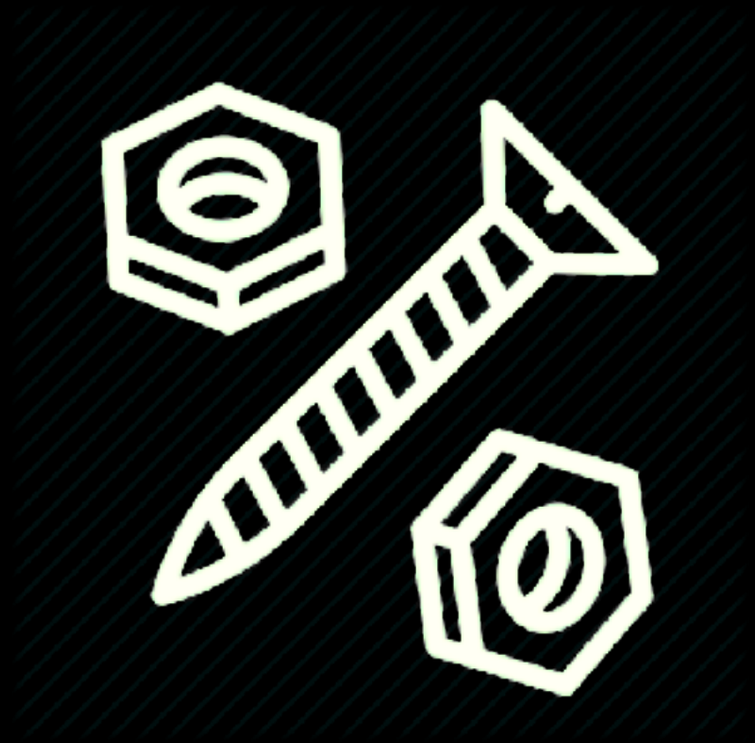
Data

Why Web API?


Web Applications
JS
Mobile Applications
3rd Party Applications
Microsoft
API Gateway
REST
Address
Micro Service
REST
User
Micro Service
RAILS
RPC
Catalog
Micro Service
node

Decoupling - Web API Good Point

API/Data Compatibility - Web API Pain Point



REST vs. RPC - 2 major paradigms

REST

- **Resource**-centric view
  - **ONE** uniform set of methods (**HTTP verbs**)
  - **Standard** media types
  - **Resources** (**nouns**)
  - Some **common knowledge** required to access **ALL** resources
- POST /posts
  title=foo&content=bar

RPC

- **Command**-centric view
  - Custom, unique **sets** of methods (verbs)
  - **Custom** media types
  - **Objects** (nouns)
  - Prior **specific knowledge** required to locate / access **each** object
- $postService->store($post);

출처: https://www.slideshare.net/trilancer/restful-user-experience-1421793

REST in Detail

REST는 모던퍼그에서 이미 발표했던 주제입니다. 슬라이드를 참고해 주세요 (클릭하여 링크 열기)
http://blog.appkr.kr/files/restful-url-design.pdf

Why RPC

- **장점**
  - IDL(Interface Definition Language)로 API 인터페이스를 정의하면
    데이터 타입(모델) 및 서비스(메서드) 클래스와 API 문서를 Generate할 수 있다.
  - 이기종 시스템/컴퓨터 언어간에 모델(as a Native Object) 자체를 주고 받을 수 있다.
    (Transparent interaction between multiple programming languages)
    - 데이터(모델) 스키마가 통제되고, API 버전간 역호환성이 유지된다.
  - REST 대비 응답 속도가 빠르고, (특히 클라이언트의) 자원 사용량이 적다.

- **단점**
  - 개발팀의 실력이 필요하다 (서버측 디버깅이 지극히 어려움).

When & Where to Use RPC

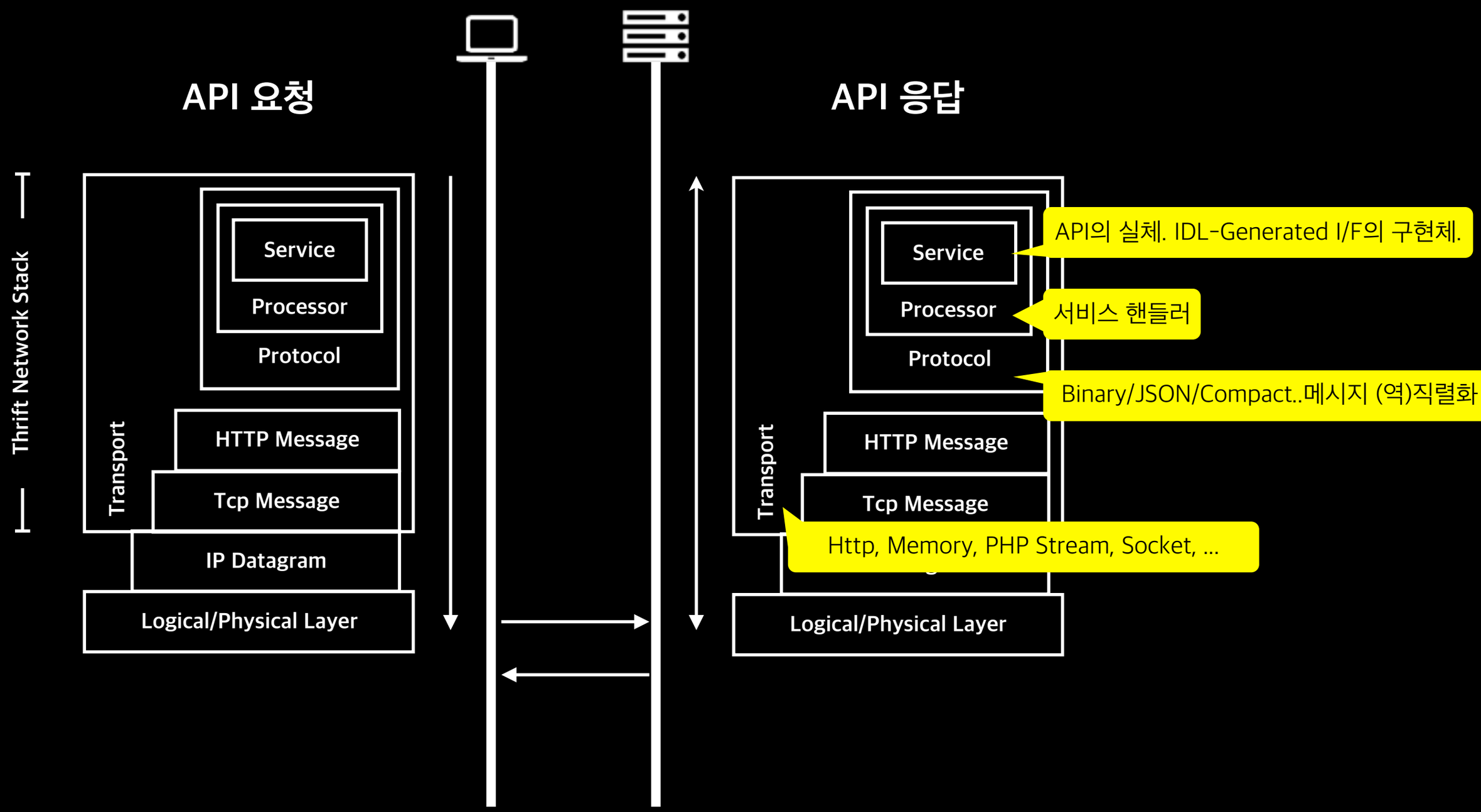
오늘 발표의 목적은

“RPC 라는 녀석이 있구나~”
"PHP 서버 프로젝트에서도 사용할 수 있구나~"

정도의 내용을 알리는 겁니다.

실전 프로젝트에 적용하려는 순간... 헬 게이트가 열릴 수 있습니다.

How Machines Talk to Each Other



Workflow

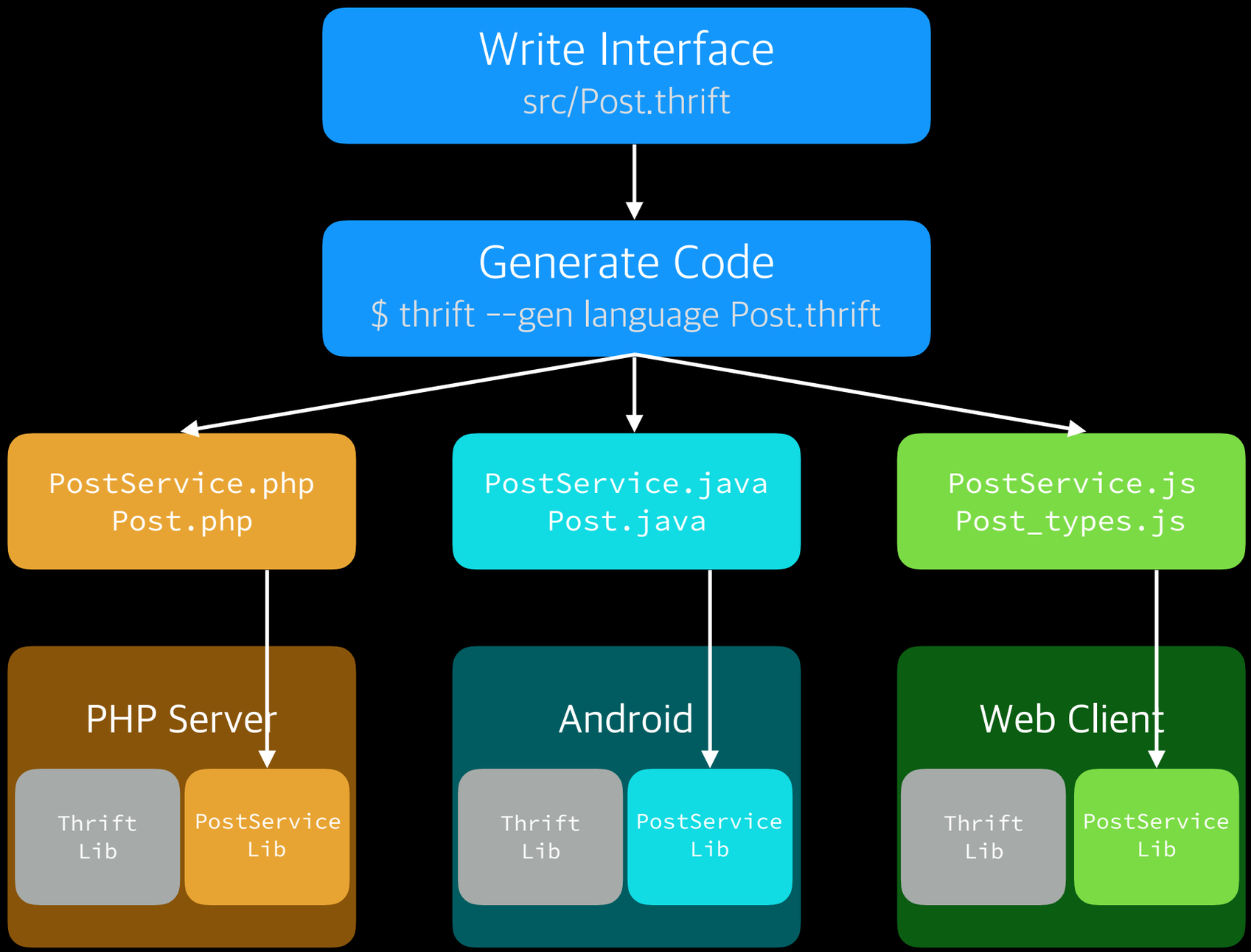
1. IDL로 메시지 형식 및 서비스 인터페이스를 정의한다.
   (서버와 클라이언트간의 약속을 정의하는 과정)

2. Thrift Generator로 정의한 인터페이스를 API 문서와 플랫폼 코드로 컴파일 한다.

   ```
   $ thrift -r --gen php:server,psr4 src/Post.thrift
   ```

3. 생성된 보일러플레이트 코드를 프로젝트로 가져와서,
   서버 또는 클라이언트 애플리케이션을 개발한다.

Workflow


Write Interface
src/Post.thrift
Generate Code
$ thrift --gen language Post.thrift
PostService.php
Post.php
PostService.java
Post.java
PostService.js
Post_types.js
PHP Server
Thrift Lib
PostService Lib
Android
Thrift Lib
PostService Lib
Web Client
Thrift Lib
PostService Lib

Proof of Concept Projects

- **appkr/thrift-example-idl**

  Apache Thrift IDL의 사용법 데모를 위한 프로젝트

  https://github.com/appkr/thrift-example-idl

- **appkr/thrift-example-project**

  Apache Thrift의 사용법 데모를 위한 PHP 서버/클라이언트 프로젝트

  https://github.com/appkr/thrift-example-project

- **appkr/thrift-js-poc-project**

  Apache Thrift의 사용법 데모를 위한 Javascript 클라이언트 프로젝트

  https://github.com/appkr/thrift-js-poc-project

Code Review & Demo

Alternative RPC Systems

Protocol Buffer

- by Google
- BSD license since 2008
- 구글이 운영 환경에서 사용 중
- 공식 지원 플랫폼은 C++/Java/Python/Javascript
- 훌륭한 문서와 커뮤니티
- Google, ActiveMQ, Netty 등에서 사용 중

gRPC

- by Google
- BSD-3 license since 2015
- Protocol Buffer의 IDL 차용
- HTTP/2 transport
- C++, Java, Python, Go, Ruby, C#, Node.js, Android Java, Objective-C, PHP(client only)
- Netflix, CoreOS, Cisco/Juniper 등에서 사용 중

RPC RECAP

- 빠르게 인터페이스를 정의하고 보일러플레이트 코드를 Generate 할 수 있다.
- 컴퓨터 기종/언어에 종속적이지 않다
- 데이터 스키마를 통제하고, 버전간 호환성을 유지할 수 있다.
- 컴퓨터 자원 소모가 적고, 통신 속도가 빠르다.

```
{"deposit_money":"12345678"}
```

JSON

```
'0x6d','0x6f','0x6e','0x65',
'0x79','0x32','0x32','0x33',
'0x34','0x35','0x36','0x37',
'0x38'
```

Binary

```
'0x01','0xBC614E'
```

Resources

- **더 자세한 구현 내용은 블로그 포스트를 참고해주세요.**
  - RPC - Apache Thrift 입문 1부
    http://blog.appkr.kr/work-n-play/how-to-use-apache-thrift-in-php-part-1/
  - RPC - Apache Thrift 입문 2부
    http://blog.appkr.kr/work-n-play/how-to-use-apache-thrift-in-php-part-2/
- **발표자가 관심 있게 지켜보고 있는 포럼 쓰레드**
  https://groups.google.com/forum/#!topic/grpc-io/F3IyYaI_6S0