

API Orchestration in the Cloud

It's just regular API Orchestration

Drew Olson

May 24th, 2021

About Me

Hi, I'm Drew Olson

- Chief Architect at GoFundMe
- Previously Chief Architect at Braintree
- <https://drewolson.org>

Agenda

Agenda

- API Orchestration
- GraphQL
- AWS + API Orchestration

API Orchestration

API Orchestration is about decoupling your API from your architectural decisions.

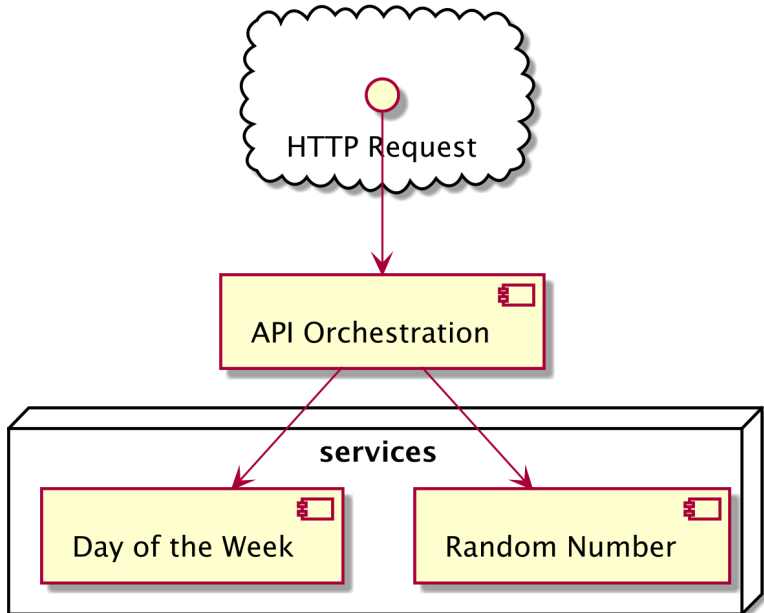
We'd like to expose a simple, coherent API to our users regardless of how we choose to build the service(s) that power our application.

Suppose we have two simple services within our application:

1. Generates a random number between 1 and 10
2. Returns the current day of the week

We can build an API Orchestration layer that provides the capabilities of each of these services by delegating to them in order to generate an API response to our user.

API Orchestration



Let's assume initially these “services” are just functions within our application (please never start with microservices). We can build this “orchestration layer” quite simply.

Here's our random_number function:

```
def random_number():  
    return randrange(0, 10)
```

Here's our `day_of_the_week` function:

```
def day_of_the_week():  
    response = requests.get(  
        "http://worldclockapi.com/api/json/cst/now"  
    )  
    body = response.json()  
  
    return body.get("dayOfTheWeek")
```

Because this code exists locally, our API orchestration layer is very simple:

API Orchestration

Because this code exists locally, our API orchestration layer is very simple:

```
app = FastAPI()
```

```
@app.get("/")
```

```
def index():
```

```
    return {
```

```
        "number": random_number(),
```

```
        "dayOfTheWeek": day_of_the_week(),
```

```
    }
```

```
$ curl localhost:8000  
{"number":9,"dayOfTheWeek":"Sunday"}%
```


If we eventually chose to extract `random_number` and `day_of_the_week` to separate services, our orchestration layer gets more complicated.

API Orchestration

```
app = FastAPI()

@app.get("/")
def index():
    return {
        "number": requests
            .get("http://random.service")
            .json()
            .get("number"),
        "dayOfTheWeek": requests
            .get("http://day-of-the-week.service")
            .json()
            .get("number"),
    }
```

Now we're making two HTTP requests to downstream services.

Now we're making two HTTP requests to downstream services.

If one service call fails, the whole API request fails.

Now we're making two HTTP requests to downstream services.

If one service call fails, the whole API request fails.

If either service call is expensive, we pay this penalty for every API request.

We can do better.

GraphQL

AWS + API Orchestration

Fin
