Microservices using Python and Oracle

Arup Nanda
Longtime Oracle DBA
And
Explorer of New Things

Agenda

What's the problem we are trying to solve?

How microservices solves the problem?

How is it different from regular process

How to do it in python and Oracle? Why?

When not to do it

Next Steps to Learn More.

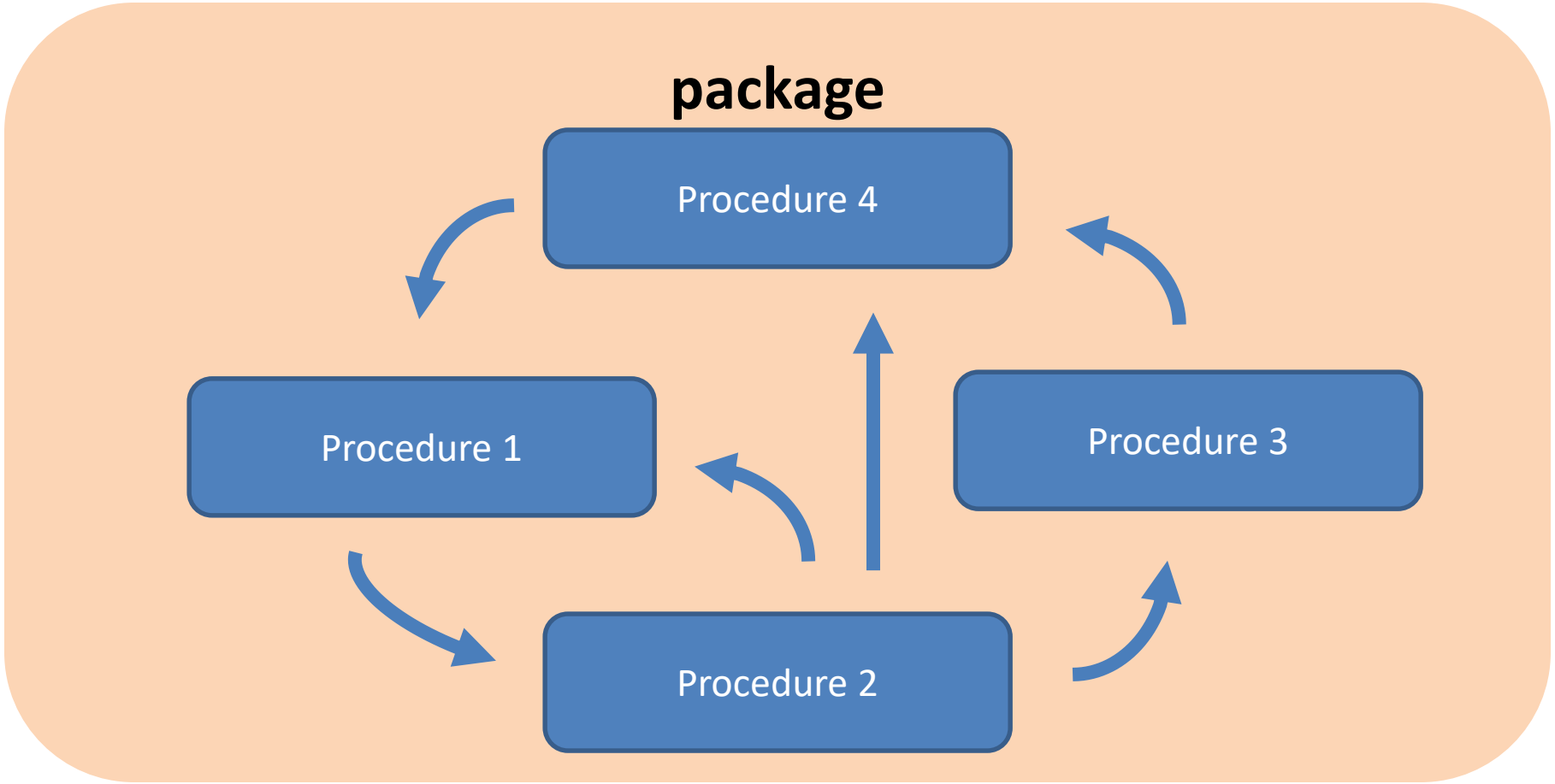
```
procedure my_entire_app (
    parameter1 …,
    parameter2 …
) is
    var1 …;
    …
    var200 …;
begin
     code line 1;
     code line 2;
     …..
     code line 1000
end;
```

```
procedure my_entire_app (
    parameter1 …,
    parameter2 …,
    …
    parameter20 …
) is
    var1 …;
    …
    var500 …;

begin
     code line 1;
     code line 2;
     …..
     code line 5000;
end;
```

```
procedure my_entire_app (
    parameter1 …,
    parameter2 …,
    …
    parameter50 …
) is
    var1 …;
    …
    var2000 …;
begin
     code line 1;
     code line 2;
     …..
     code line 10000;
end;
```

```
procedure my_entire_app (
    parameter1 …,
    parameter2 …,
    …
    parameter50 …
) is
    var1 …;
    …
    var2000 …;
begin
     code line 1;
     code line 2;
     …..
     code line 10000;
end;
```

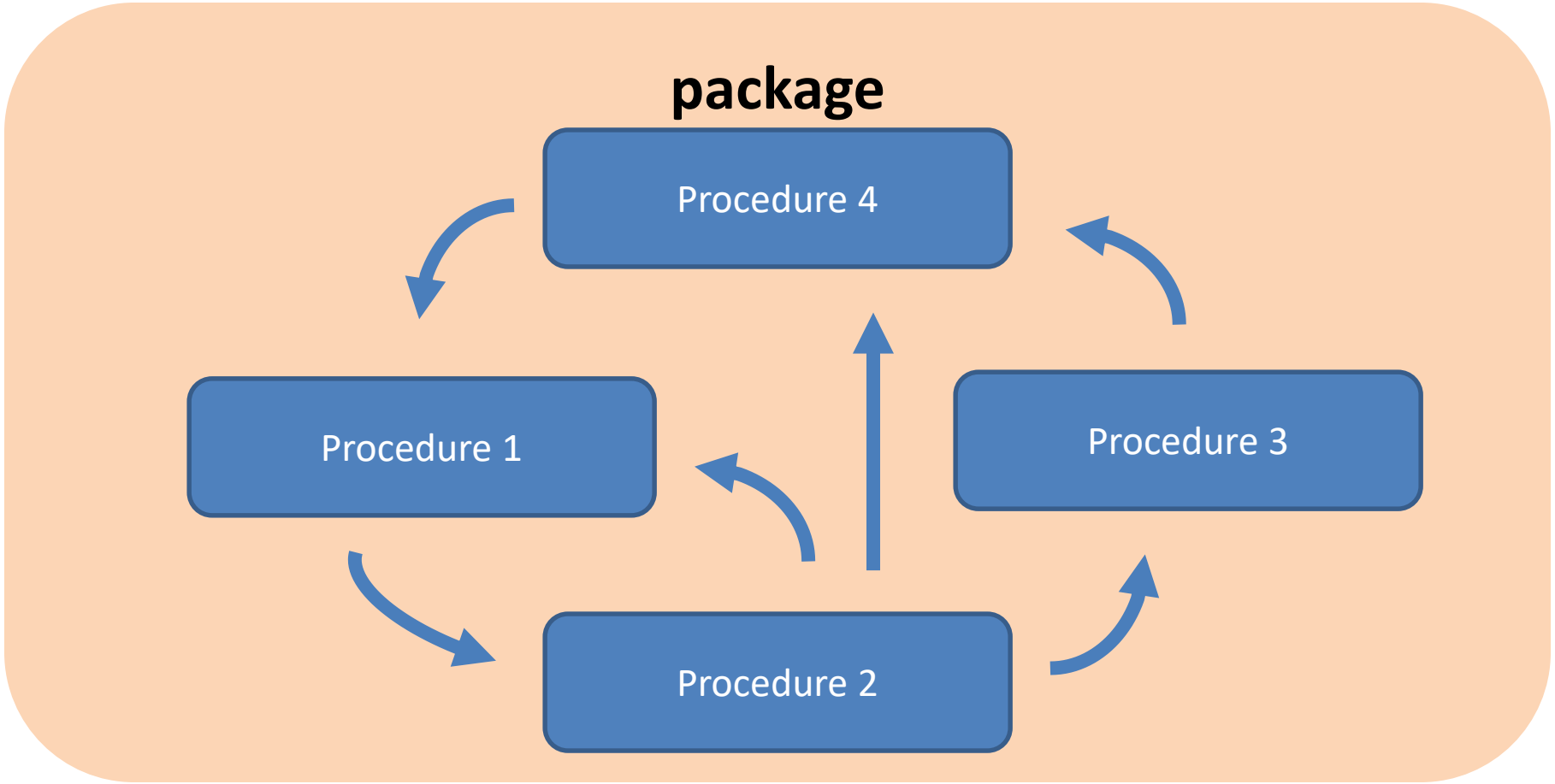
```
package my_entire_app (
   procedure manage_cust;
   procedure manage_order;
   procedure manage_payment;
```


package
Procedure 4
Procedure 1
Procedure 3
Procedure 2

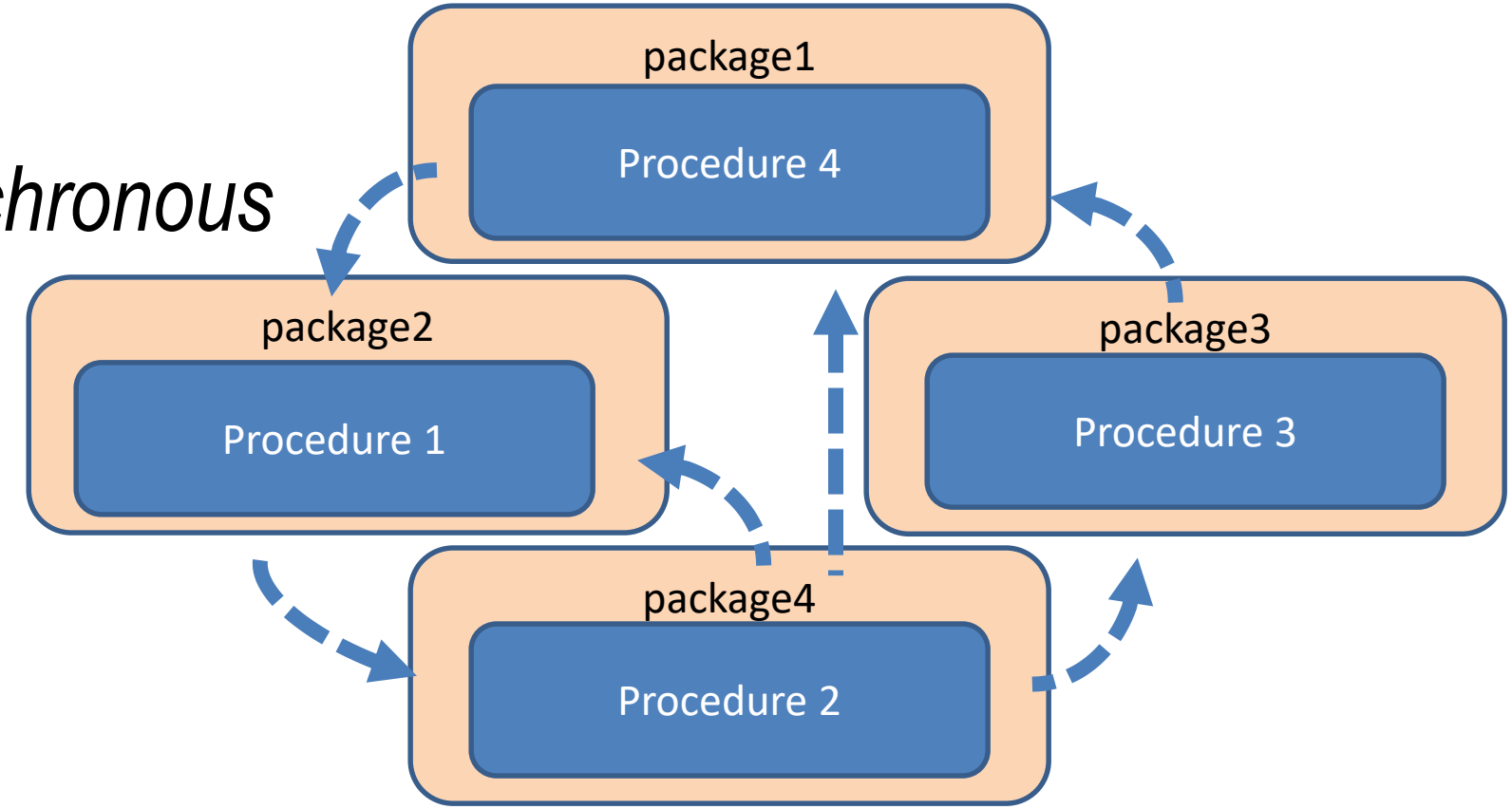
Scaling up

Problems with monoliths

- Risk of change to any functionality
- Lack of innovation
- Compile time
- Exposure to failure
- Scalability.

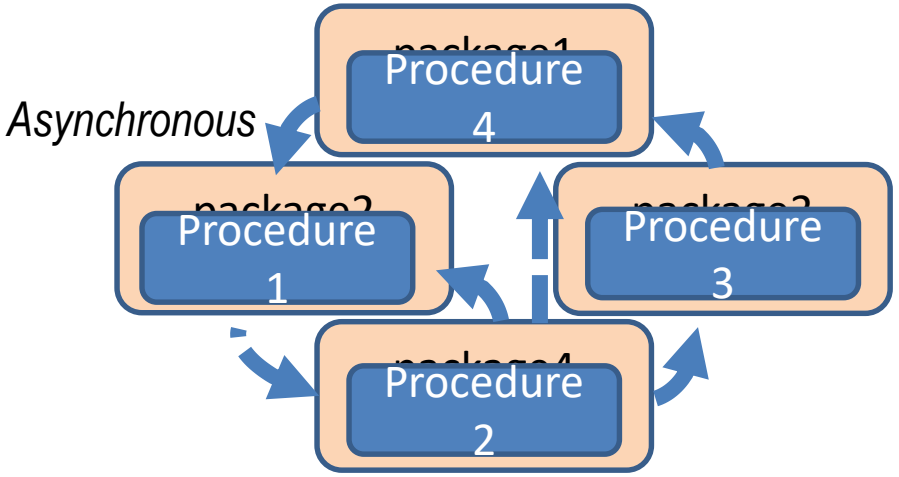

package
Procedure 4
Procedure 1
Procedure 3
Procedure 2

Asynchronous



Stateful Problem

- Typical application
  - Connect to database
  - Fetch something
  - Let the database handle where the last fetch was done



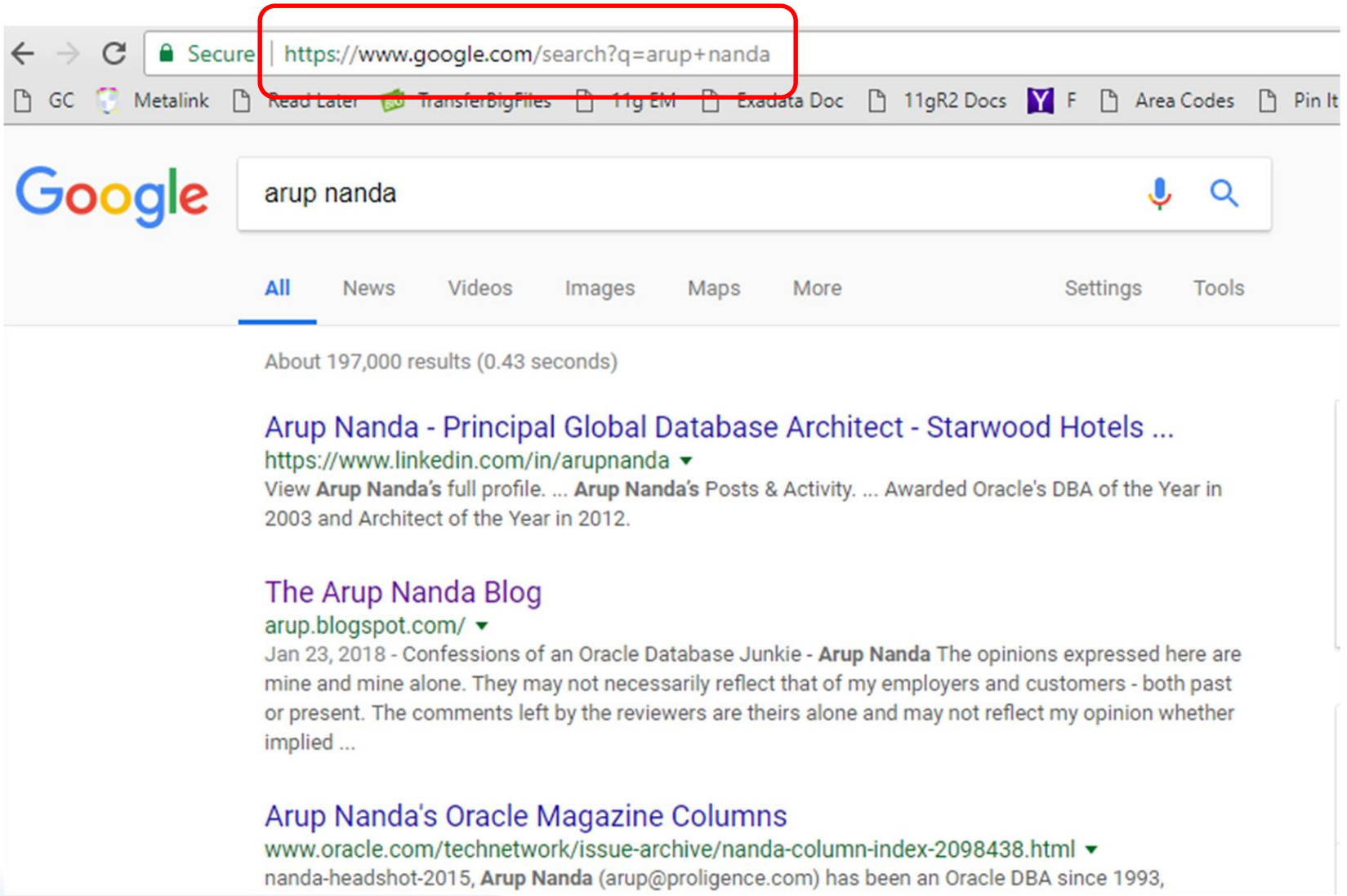
Stateless

API Call

API Call Example

Browser

Google API



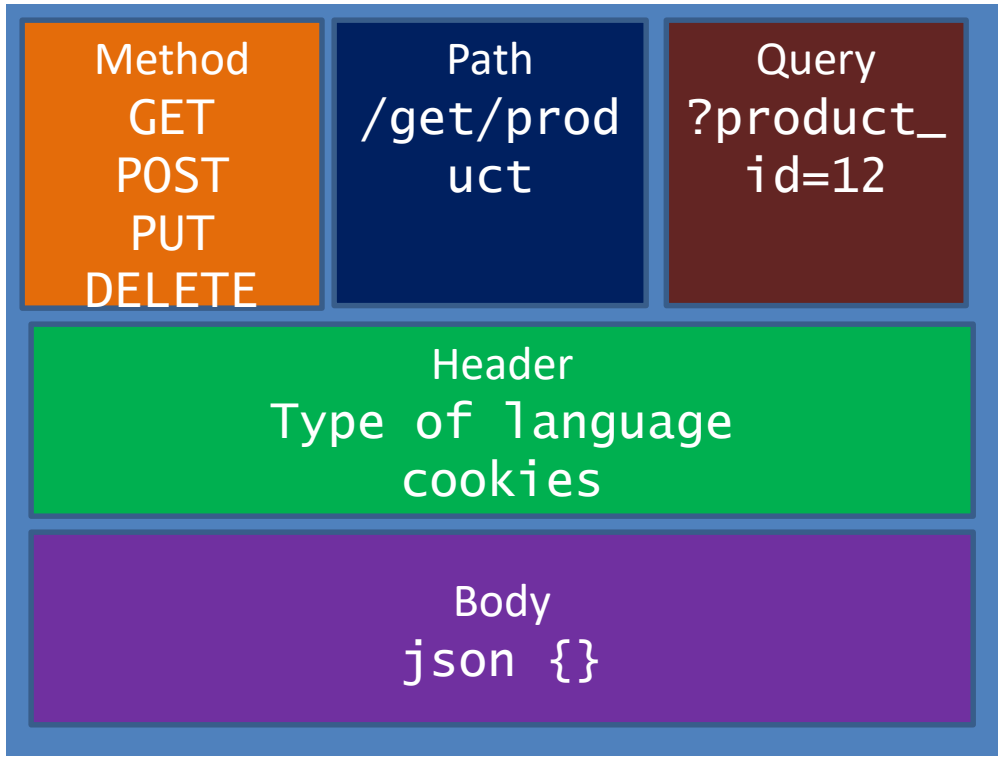
Analogy to a typical app

Insert	**C**	CREATE	POST
Select	**R**	READ	GET
Update	**U**	UPDATE	PUT
Delete	**D**	DELETE	DELETE

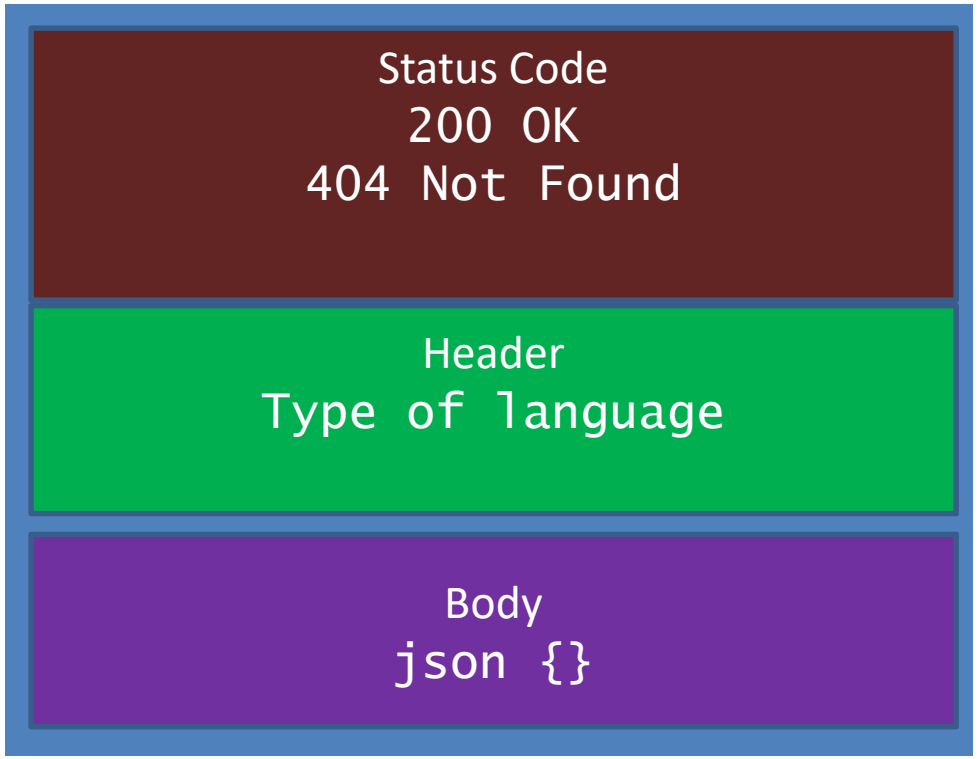
REST API

http

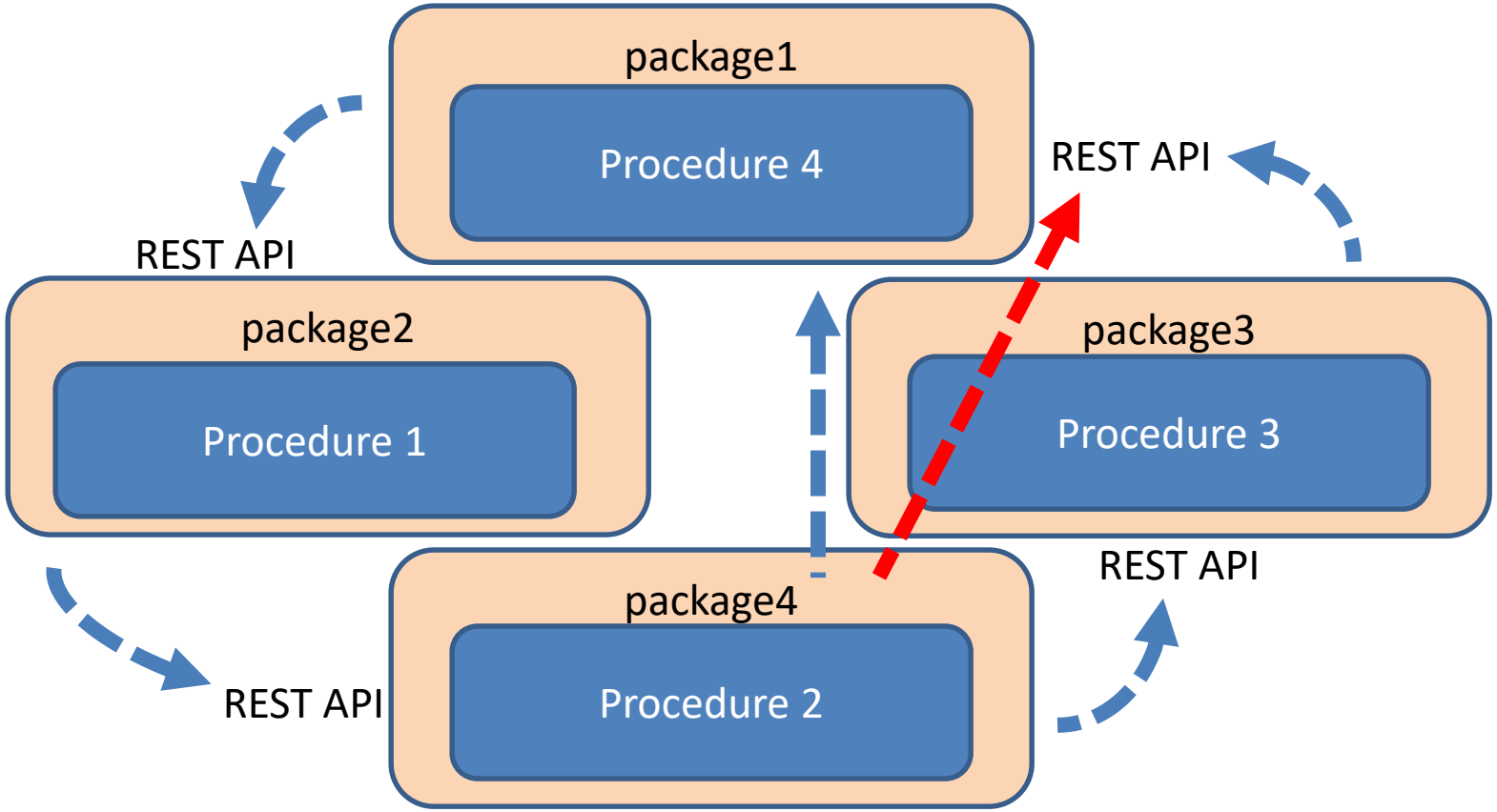
Structure of HTTP Requests

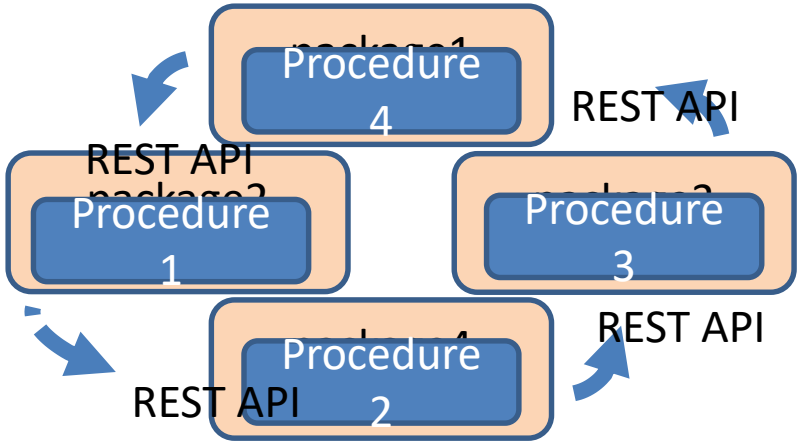


Request



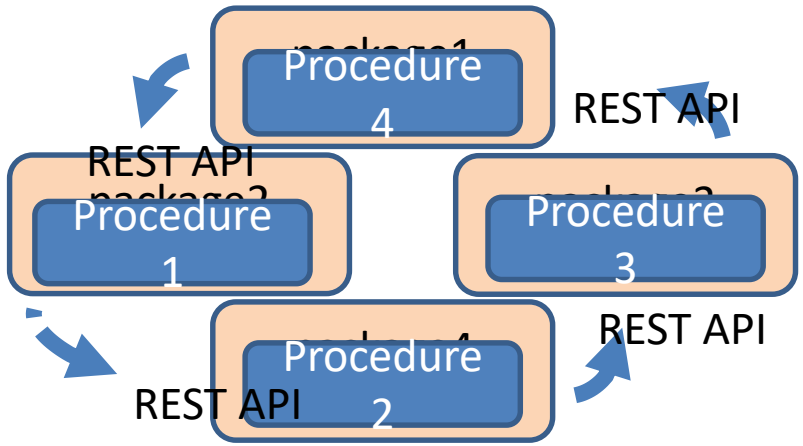
Response


package1
Procedure 4
REST API
REST API
package2
Procedure 1
package3
Procedure 3
REST API
package4
Procedure 2
REST API



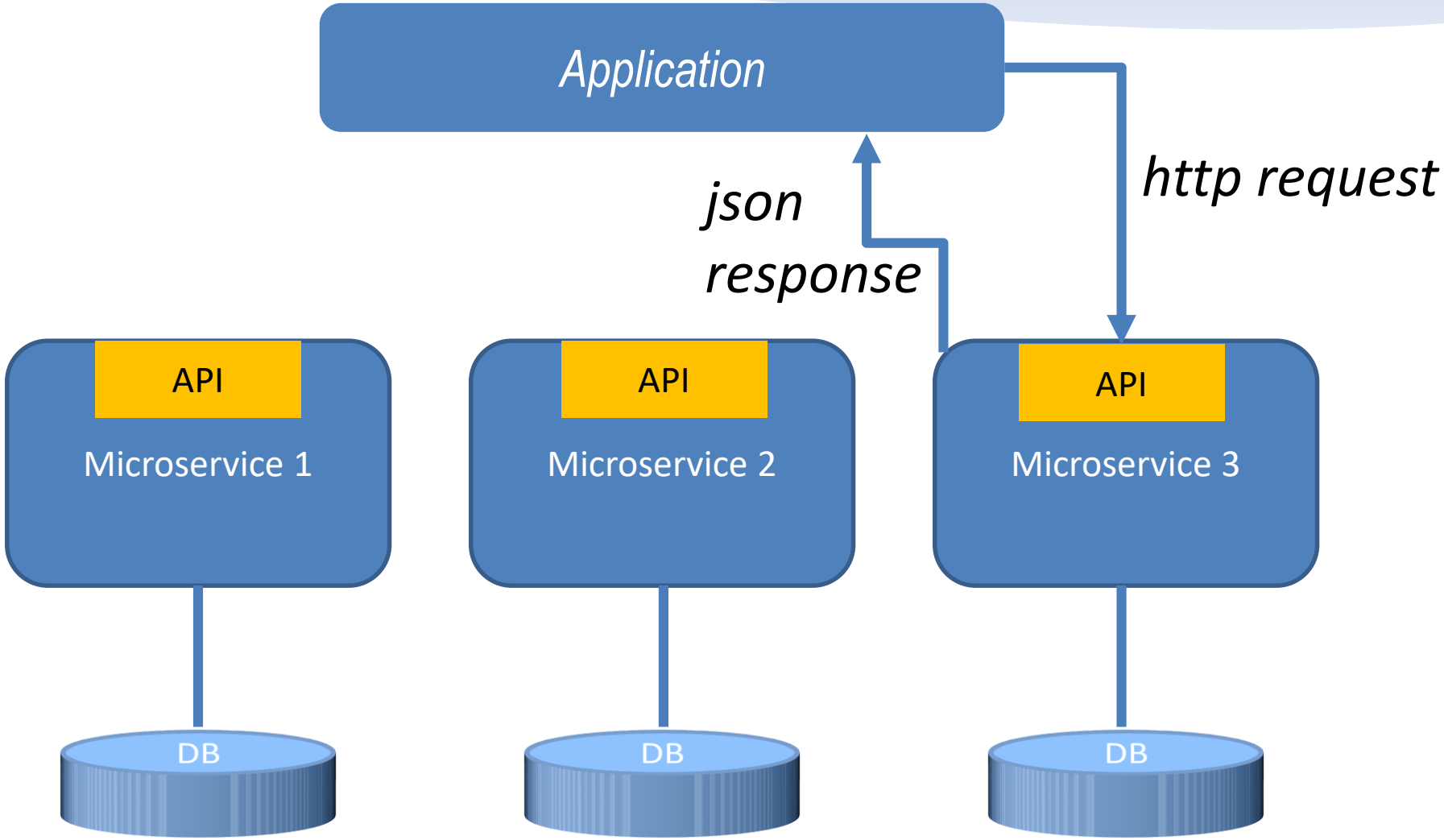
Microservice

service–oriented
architecture
composed of
loosely coupled
elements
that have
bounded contexts

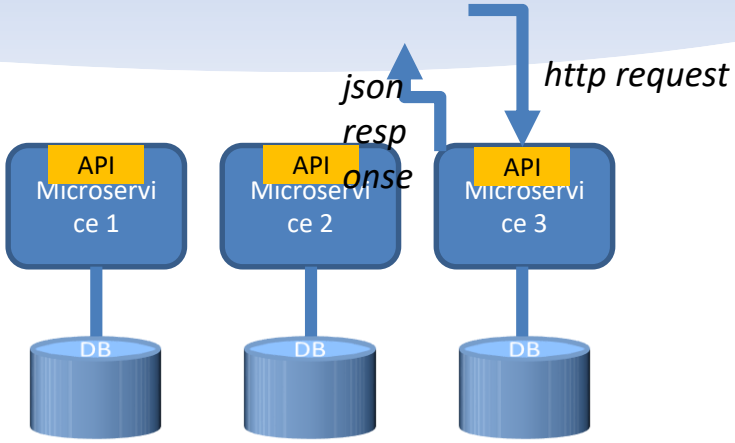


Do One Thing Only, and Well


WENGER


Application
http request
json
response
API
Microservice 1
API
Microservice 2
API
Microservice 3
DB
DB
DB

Putting it Together



Python for the actual microservices code

HTTP processing: Flask

```
python –m pip install Flask
```

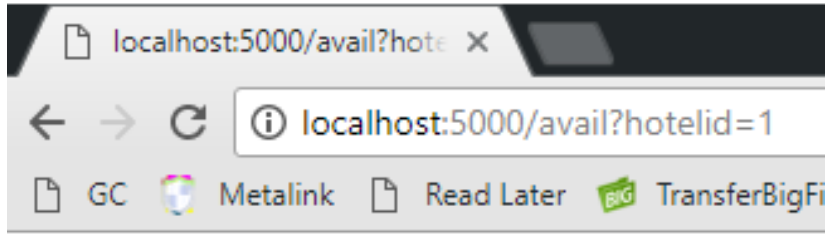
Database Connectivity: cx_Oracle

```
python –m pip install cx_Oracle
```

Python Code

```
from flask import Flask, redirect, url_for, request, render_template
app = Flask(__name__)

hotelavail={
            1:
            {'Room1':10,'Room2':5,'Room3':7},
            2:
            {'Room1':9,'Room2':2,'Room3':0},
            3:
            {'Room1':6,'Room2':11,'Room3':3}
            }

@app.route('/avail', methods= ['GET'])
def avail():
    if request.method == 'GET':
        hotelid = int(request.args.get('hotelid'))
        return render_template('list_avail.html',dict=hotelavail[hotelid])
    else:
        Null

if __name__ == '__main__':
    app.run(debug=1)
```

C:\> python hotel.py



Room Type	Available
Room1	10
Room2	5
Room3	7

Code to Get from Database

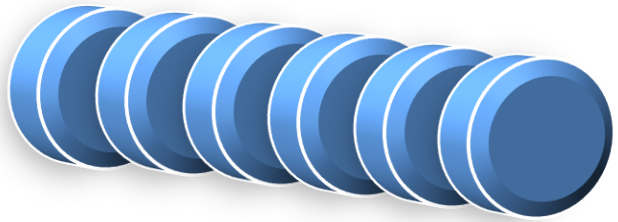
```
from flask import Flask, redirect, url_for, request, render_template
import cx_Oracle as cxo
app = Flask(__name__)

def getHotelAvail(hotelid):
    conn = cxo.connect('hr','hr','10.12.8.43:1521/orcl')
    c1 = conn.cursor()
    c1.prepare('select room_type, avail from avail where hotel_id = :hotelid')
    c1.execute(None,{'hotelid':int(hotelid)})
    hotelavail = dict(c1.fetchall())
    return hotelavail

@app.route('/avail', methods= ['POST','GET'])
def avail():
    if request.method == 'GET':
        hotelid = int(request.args.get('hotelid'))
        hotelavail = getHotelAvail(hotelid)
        return render_template('list_avail.html',dict=hotelavail)
    else:
        return Null

if __name__ == '__main__':
    app.run(debug=1)
```

Python for PL/SQL Professionals

http://bit.ly/python4plsql

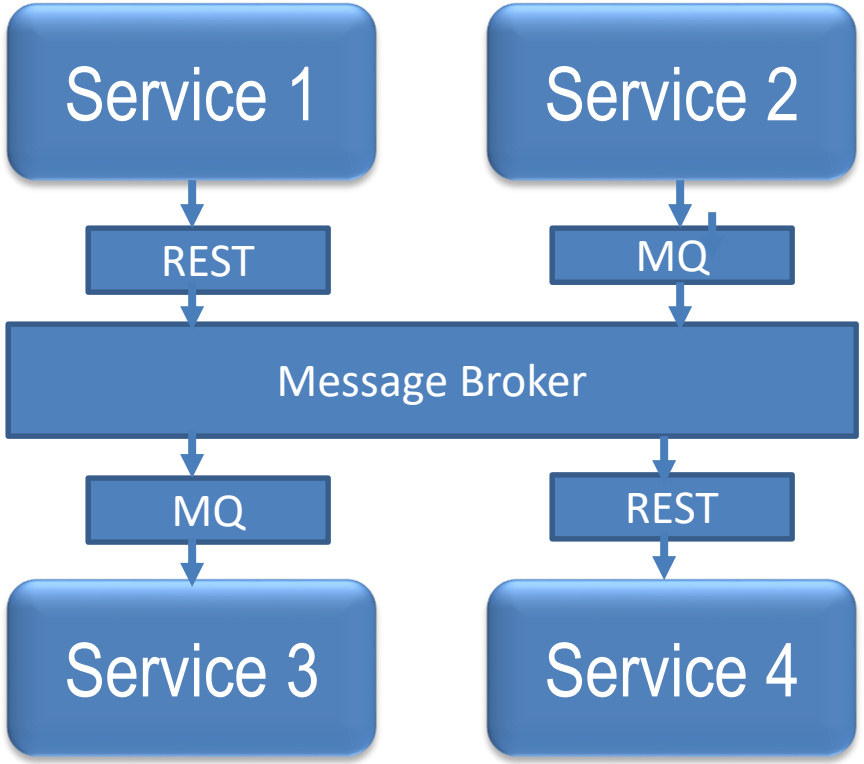
How is it different from Service Oriented Architecture

SOA

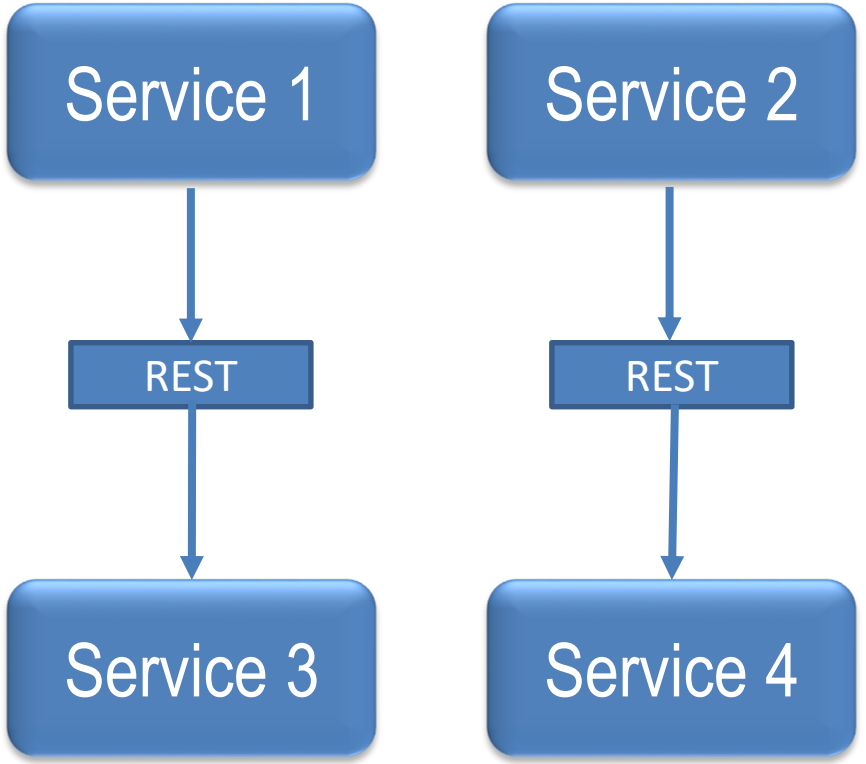


Microservices

SOA



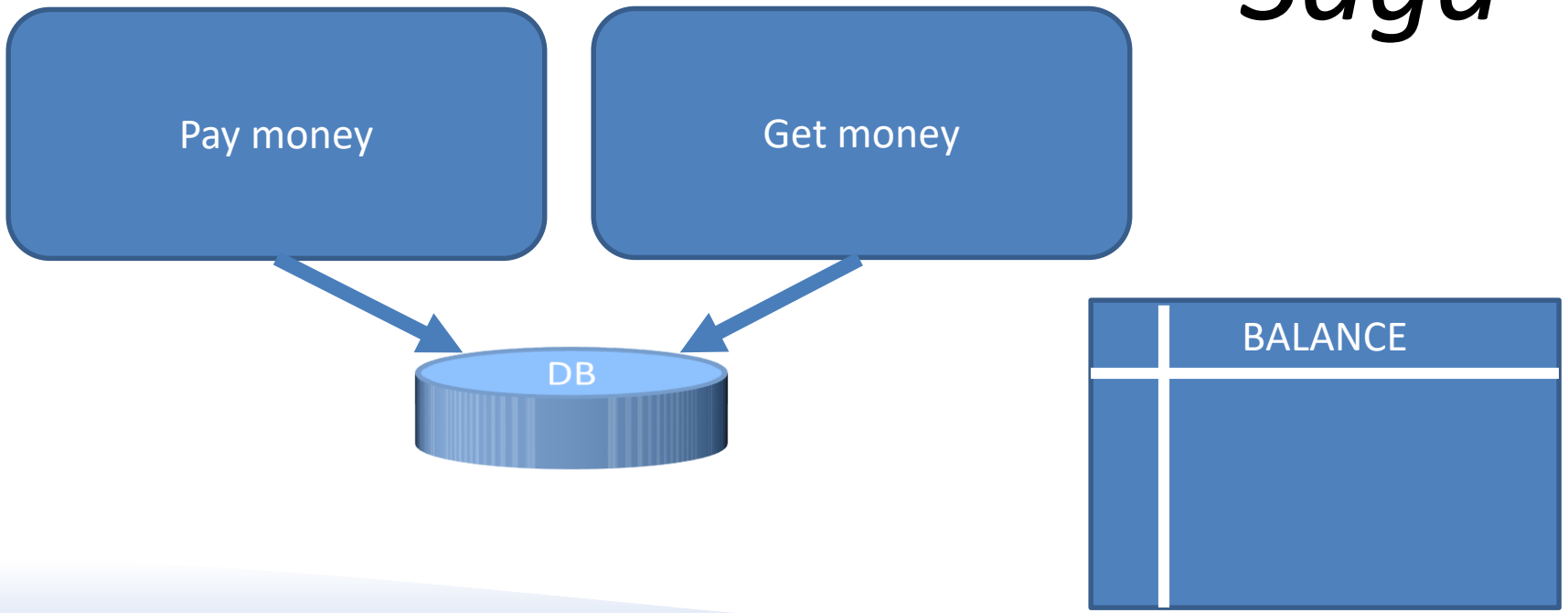
Microservices



When it's NOT good

- When you don't need to scale
- When the database transactions are important across the services

Saga



Thank You!

Blog: arup.blogspot.com
Tweeter: @ArupNanda
Facebook.com/ArupKNanda
Google Plus: +ArupNanda