

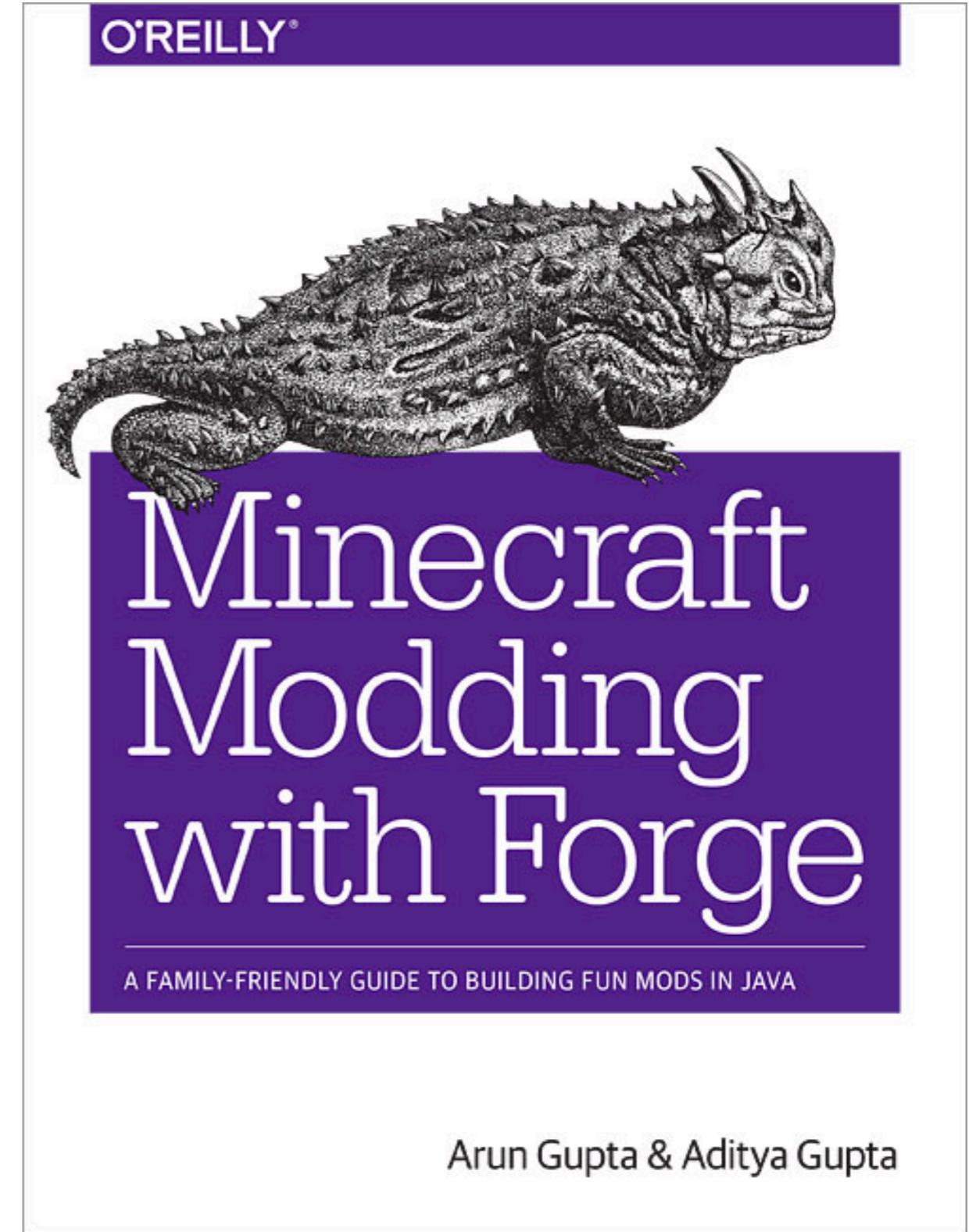
Refactoring your Java EE applications using Microservices and Containers

Arun Gupta

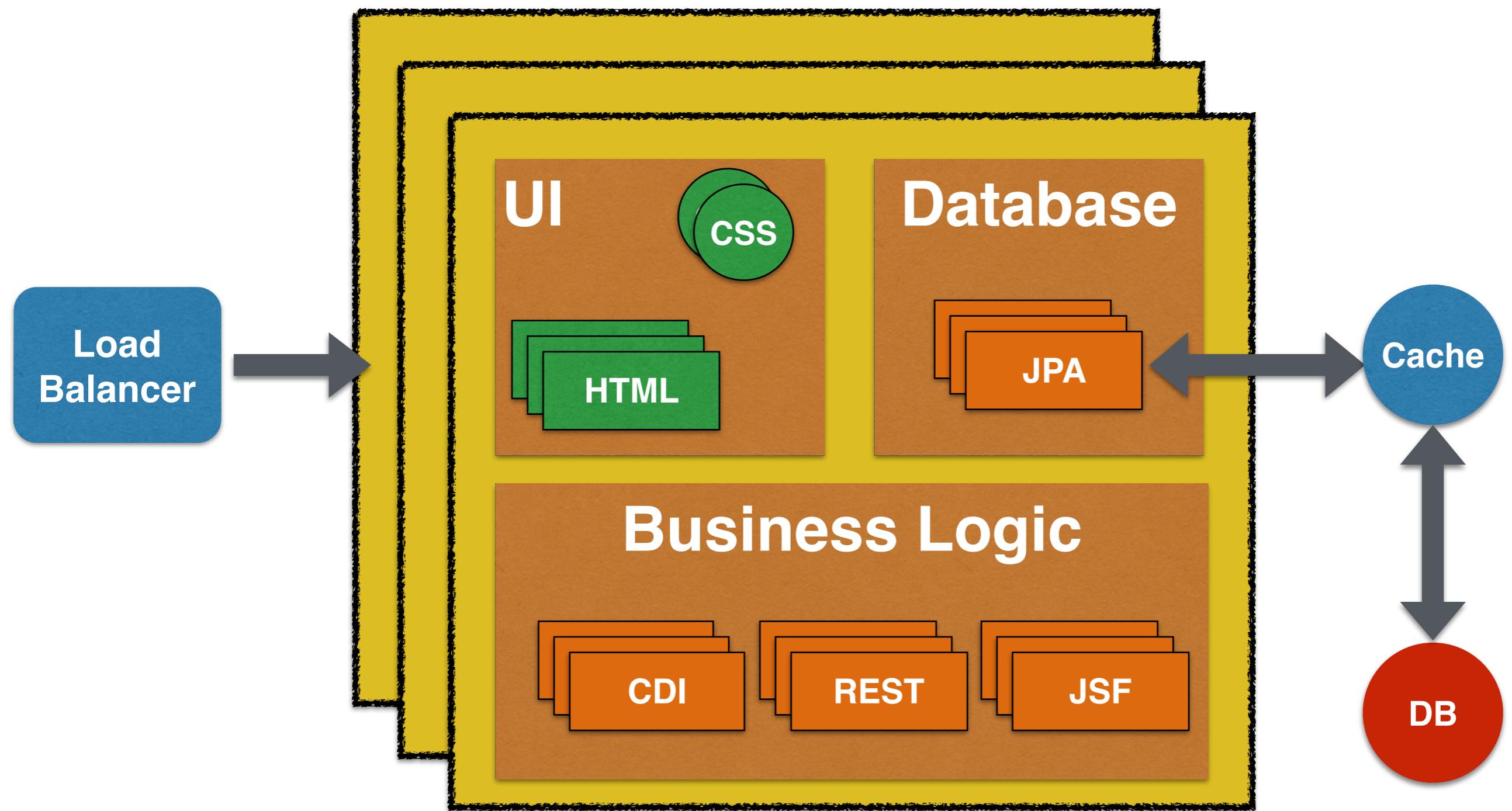
Vice President, Developer Advocacy

@arungupta, blog.arungupta.me

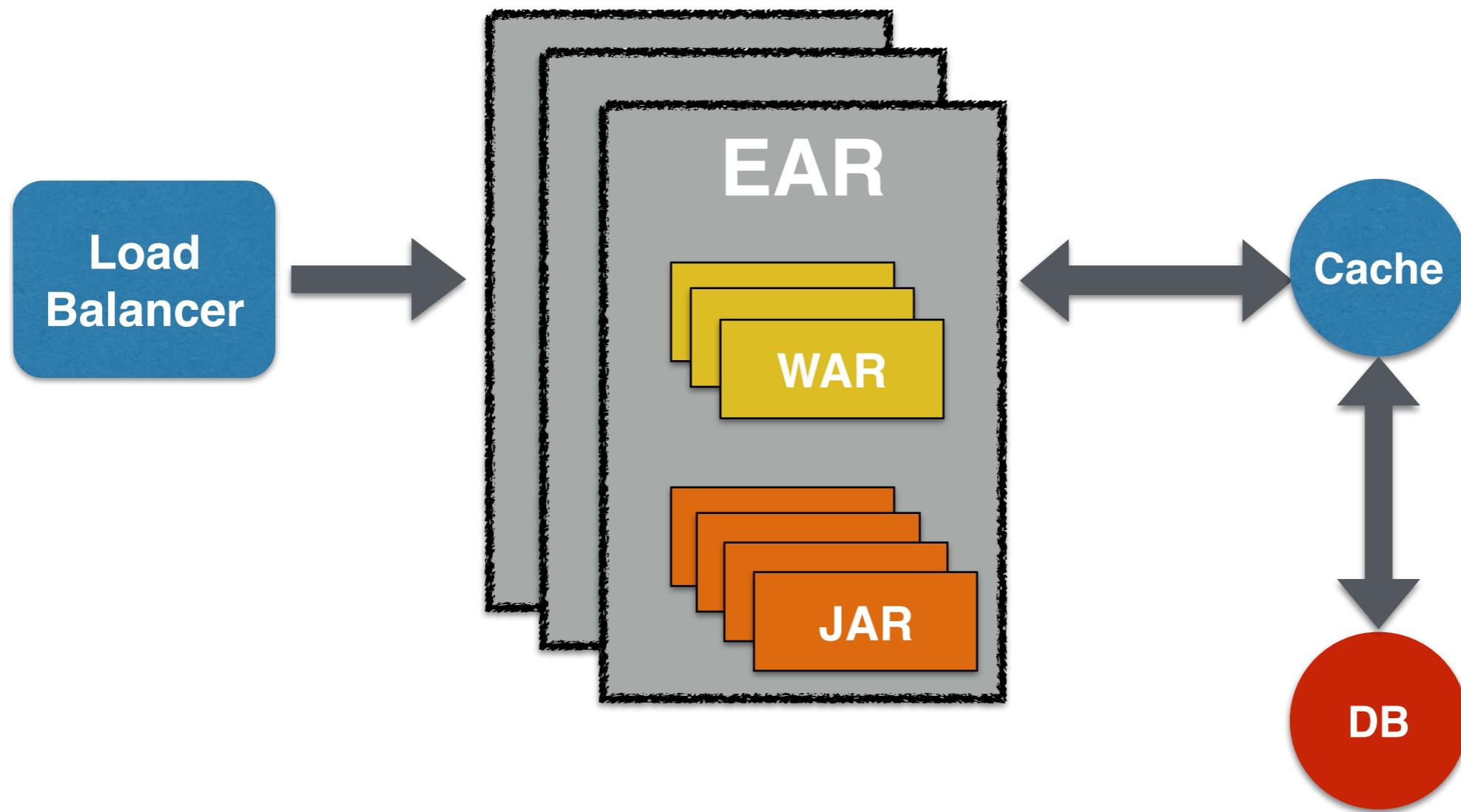
arun@couchbase.com



Monolith Application



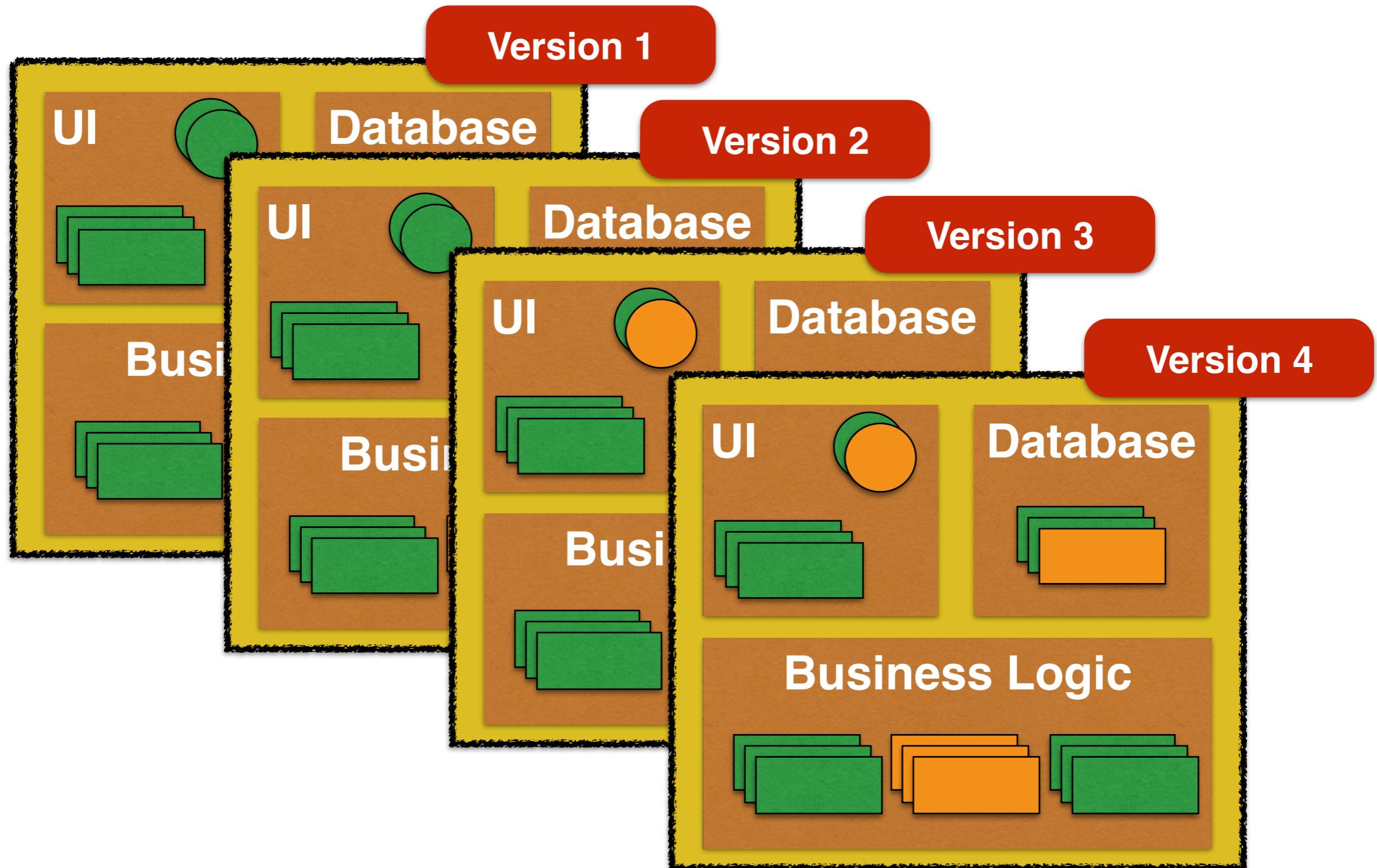
Monolith Application



Advantages of Monolith Application

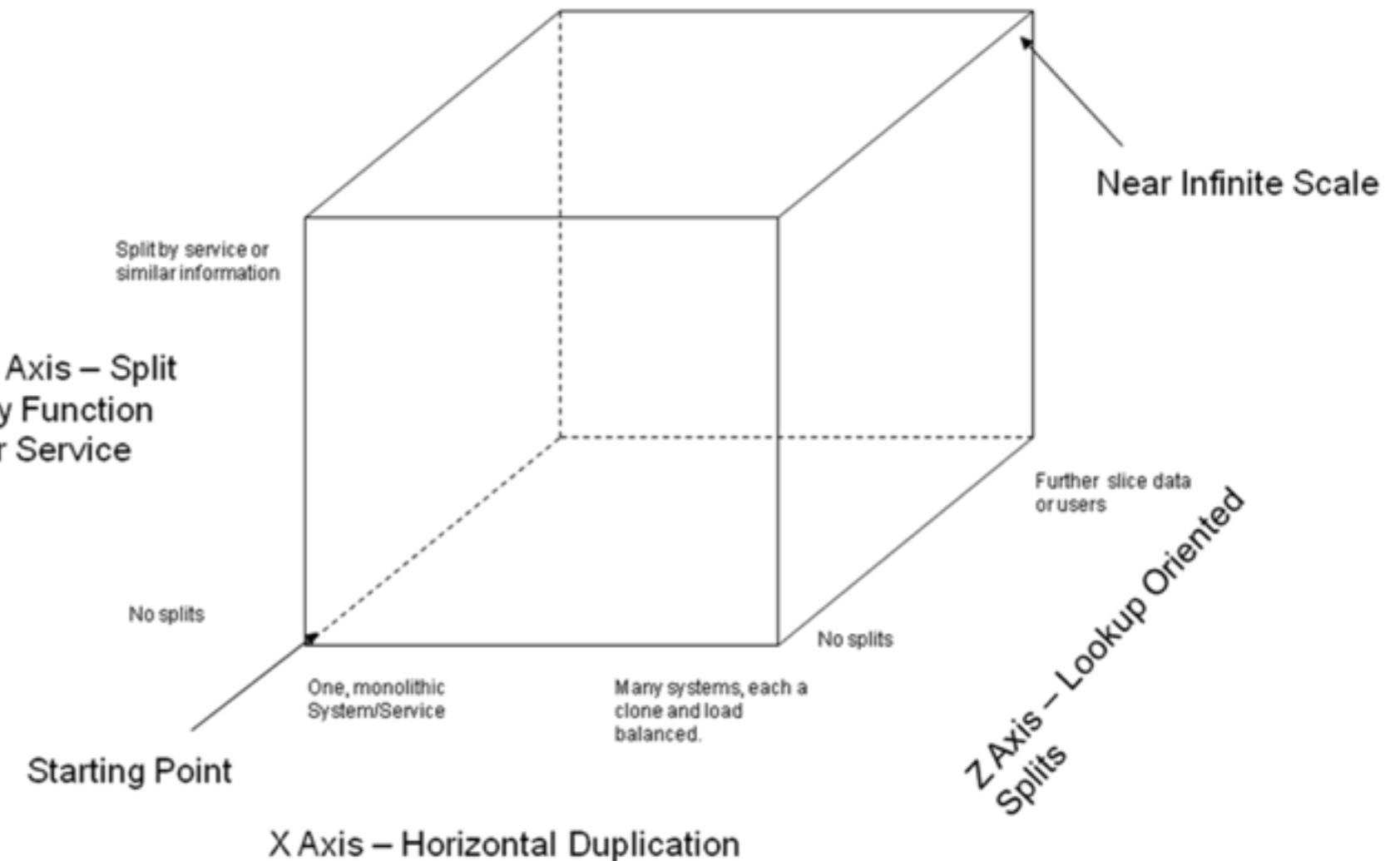
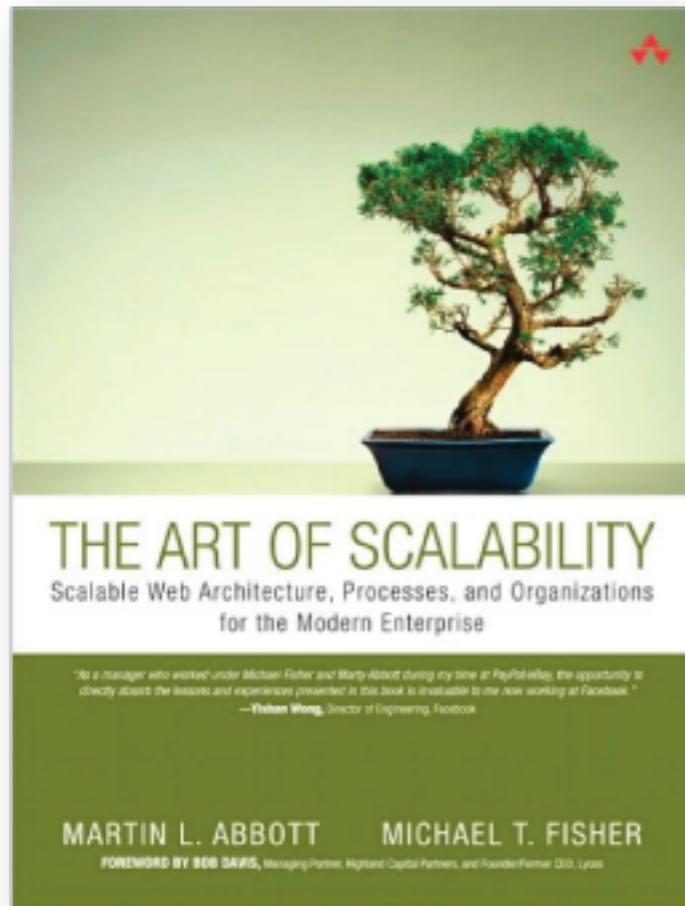
- Typically packaged in a single .ear
- Easy to test (all required services are up)
- Simple to develop

Monolith Application



Disadvantages of Monolith Application

- Difficult to deploy and maintain
- Obstacle to frequent deployments
- Dependency between unrelated features
- Makes it difficult to try out new technologies/framework

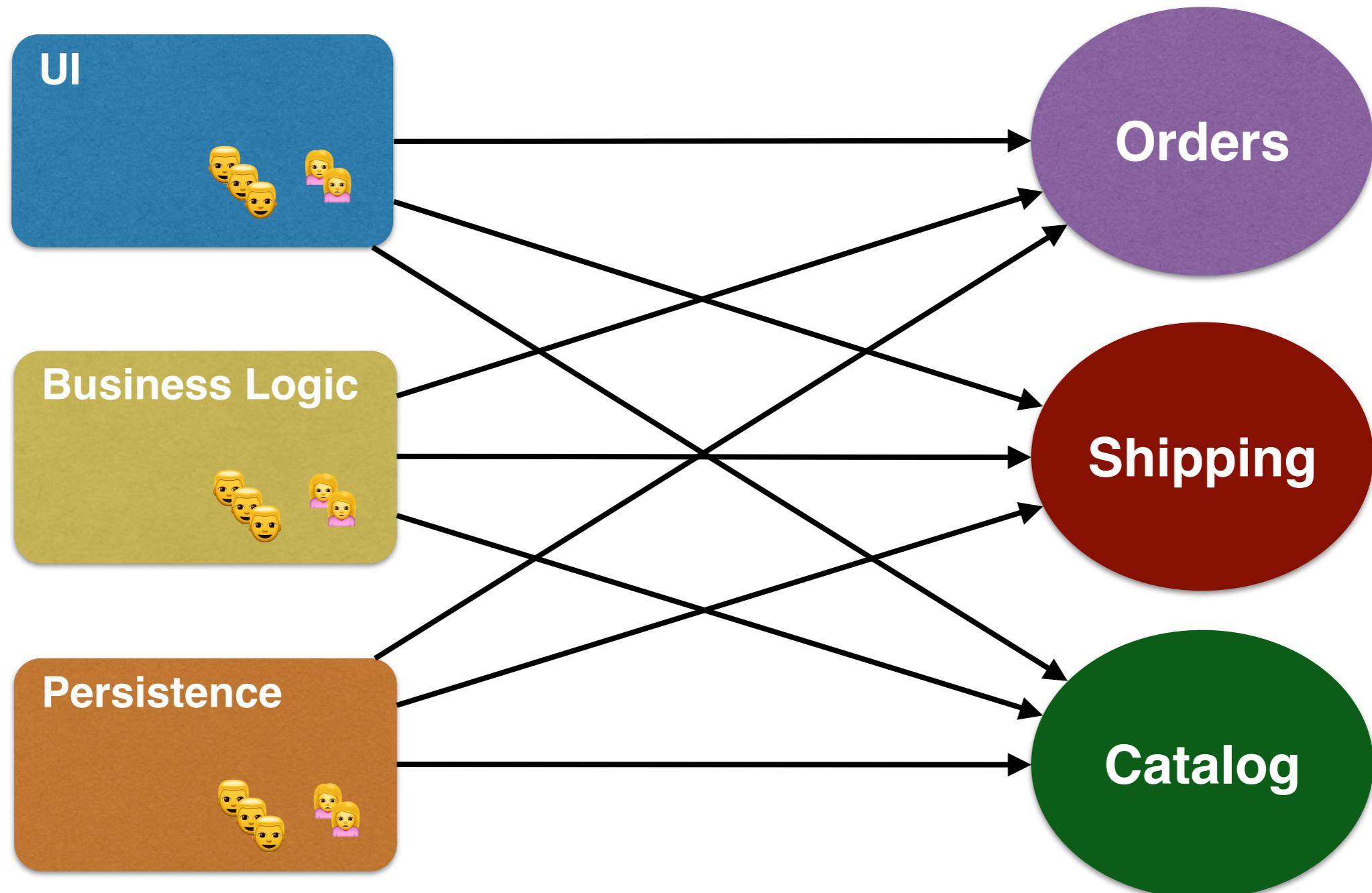


An ***architectural approach***, that emphasizes the ***decomposition of applications*** into ***single-purpose, loosely coupled*** services managed by ***cross-functional teams***, for delivering and maintaining ***complex software systems*** with the velocity and quality required by today's ***digital business***

I DONT ALWAYS
BUILD EVERYTHING



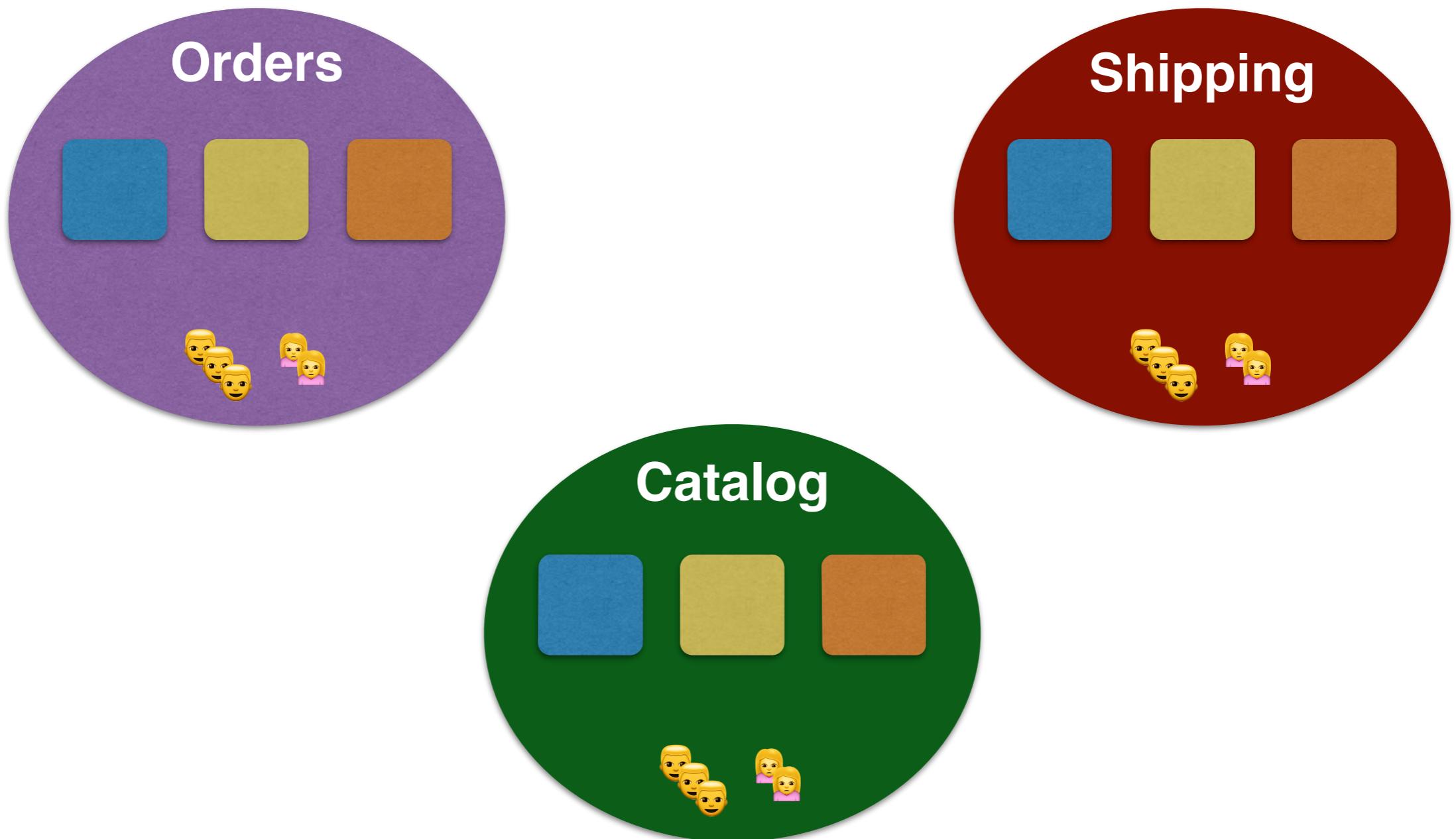
imgflip.com



*“Any **organization** that designs a system
(defined more broadly here than just information
systems) will inevitably produce a design whose
structure is a **copy of the organization's
communication structure.**”*

–Melvin Conway

Teams around business capability



Single Responsibility Principle

DO
1
THING

Explicitly Published interface



Independently replaceable and upgradeable





With great
power, comes great
responsibility



“you build it, you run it!”

Designed for failure



Fault tolerance is a requirement, not a feature



Characteristics



Scala



ORACLE
D A T A B A S E



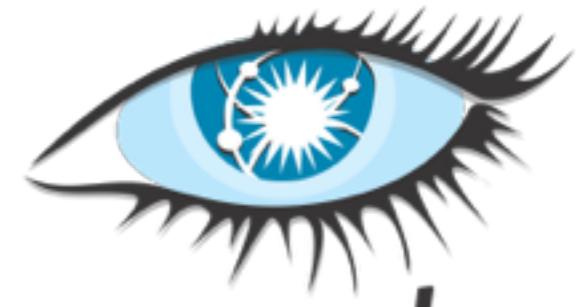
PostgreSQL



Couchbase



redis



cassandra

100% automated

WELLS FARGO

Bill Pay Overview Payments Payees eBills Reports Notices User Profile

Bill Pay Overview

Make Payment

Note: Delivery time for payment varies by payee. See number of business days in Send On column.

Payee	Pending Payment	Last Paid	Amount	Send On
AMERICAN EXPRESS	\$2,053.50	\$1,349.93	\$ []	mm/dd/yyyy 3 Business Days
BANK OF AMERICA <small>eBill</small> Receiving eBills View eBill	\$198.80	\$92.17	\$ []	mm/dd/yyyy 3 Business Days
BANK ONE / FIRST	\$55.00	\$55.00	\$ []	mm/dd/yyyy 3 Business Days
CHARLES SCHWAB			\$ []	mm/dd/yyyy 5 Business Days
CITIBANK VISA <small>eBill</small> Pending activation	\$63.50	\$198.80	\$ []	mm/dd/yyyy 5 Business Days
DIRECT TV <small>eBill</small> Activate eBills		\$63.50	\$ []	mm/dd/yyyy 3 Business Days
SBC-PACIFIC BELL		\$45.80	\$ []	mm/dd/yyyy 5 Business Days
SPRINT PCS <small>eBill</small> Activate eBills	\$49.78	\$63.50	\$ []	mm/dd/yyyy 5 Business Days
SFPUC-WATER DE			\$ []	mm/dd/yyyy 3 Business Days
WF HOME MORTGAGE		\$1,349.93	\$ []	mm/dd/yyyy 5 Business Days

Help

[Unviewed Notices](#) (2)
[Unpaid eBills](#) (3)
[Pending Payments](#) (6)

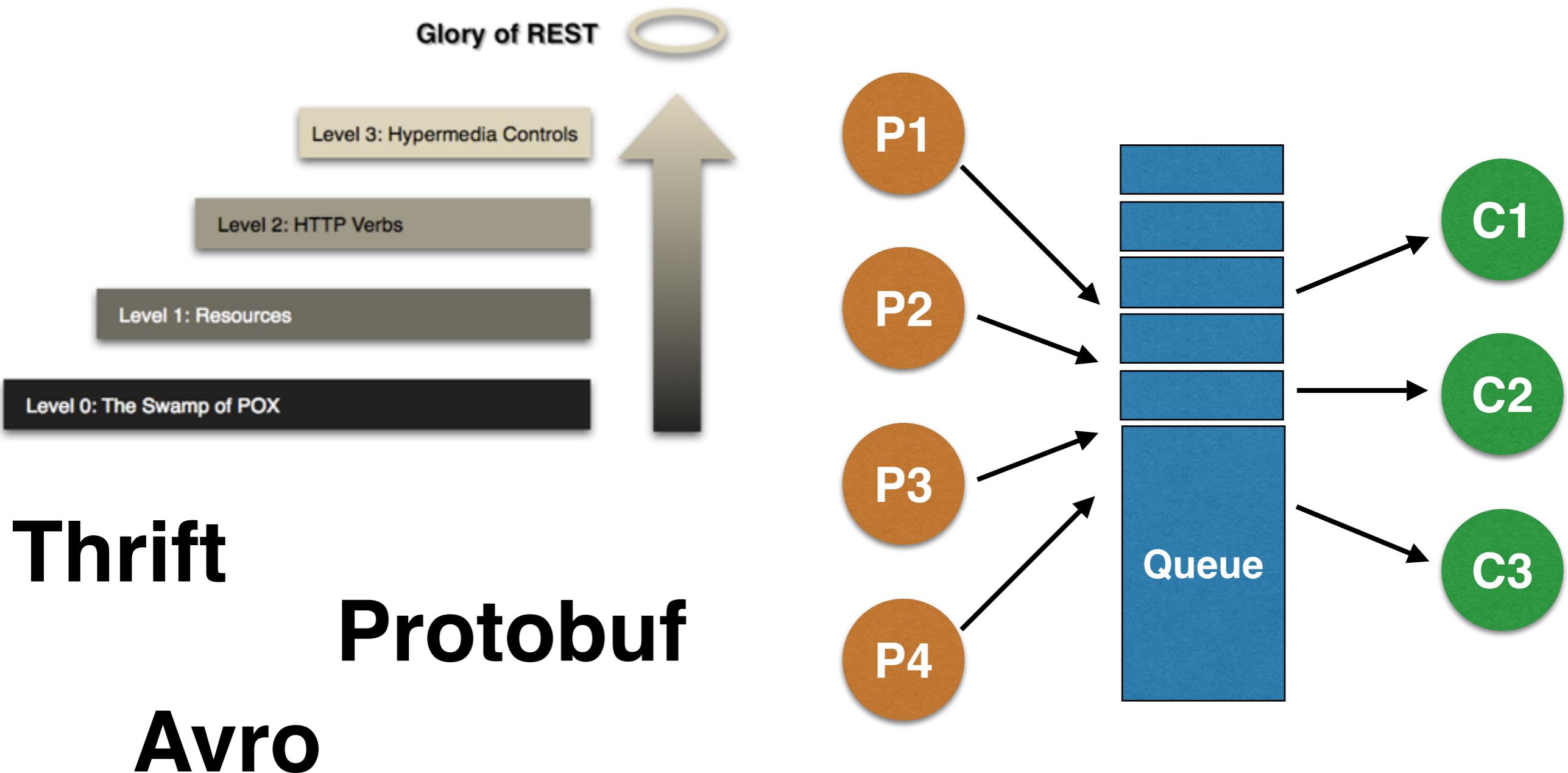
Make Payment

Home | Locations | Contact Us | Open Account | Sign Off
© 2001-2005 Wells Fargo. All rights reserved.

Sync or Async Messaging



Sync vs Async



“Smart endpoints Dumb pipes”



SOA

- SOA 2.0
- Hipster SOA
- SOA done right
- SOA++

SOA 2.0?



Arun Gupta
@arungupta

- Conway's Law
- Service Discovery
- Immutable VM

Microservices = SOA -ESB -SOAP -
Centralized governance/persistence -
Vendors +REST/HTTP +CI/CD +DevOps
+True Polyglot +Containers +PaaS WDYT?



RETWEETS FAVORITES
72 **63**



5:07 PM - 27 May 2015

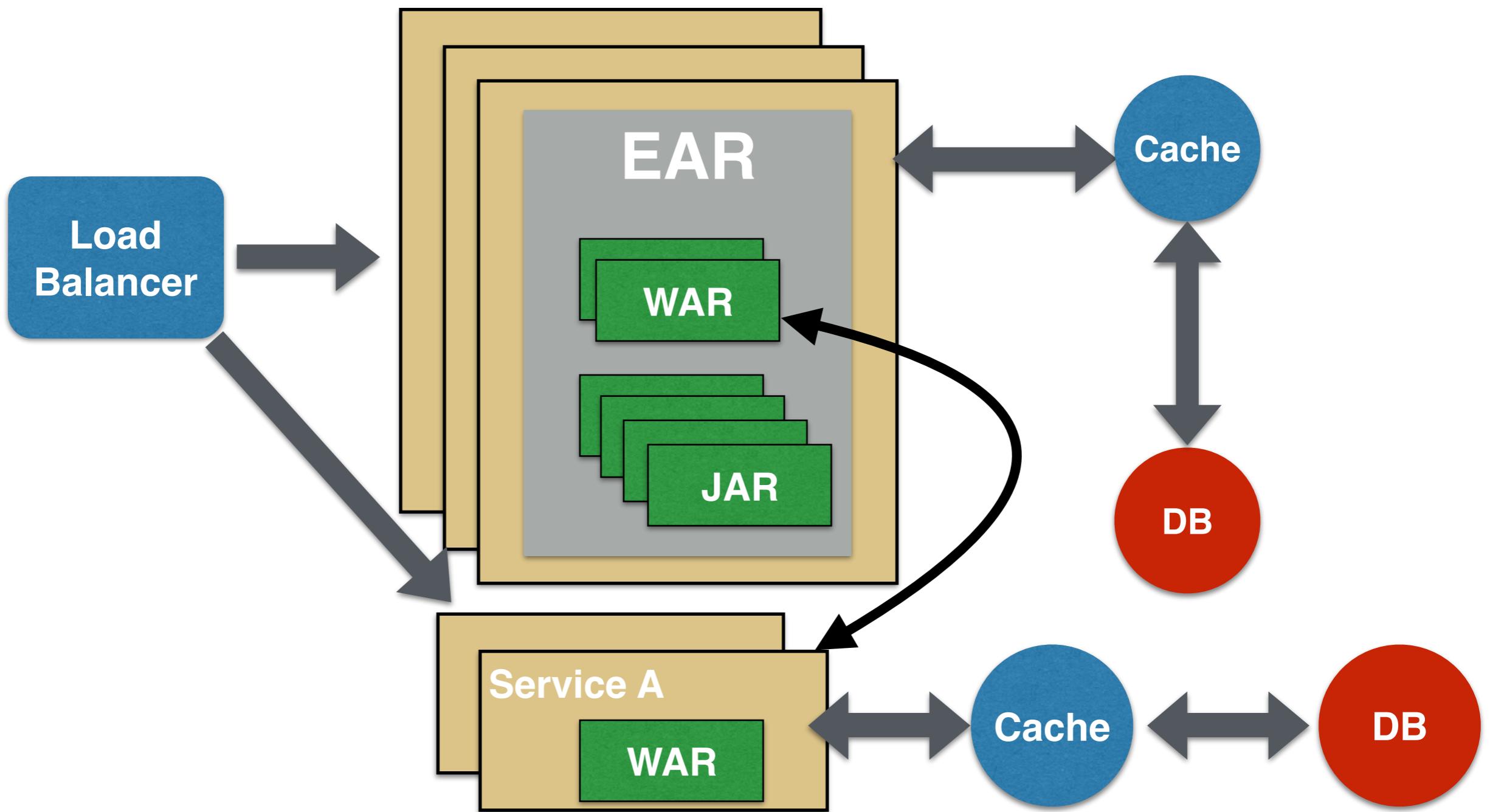
**AM I THE ONLY
ONE**

**WHO DID NOT
UNDERSTAND**

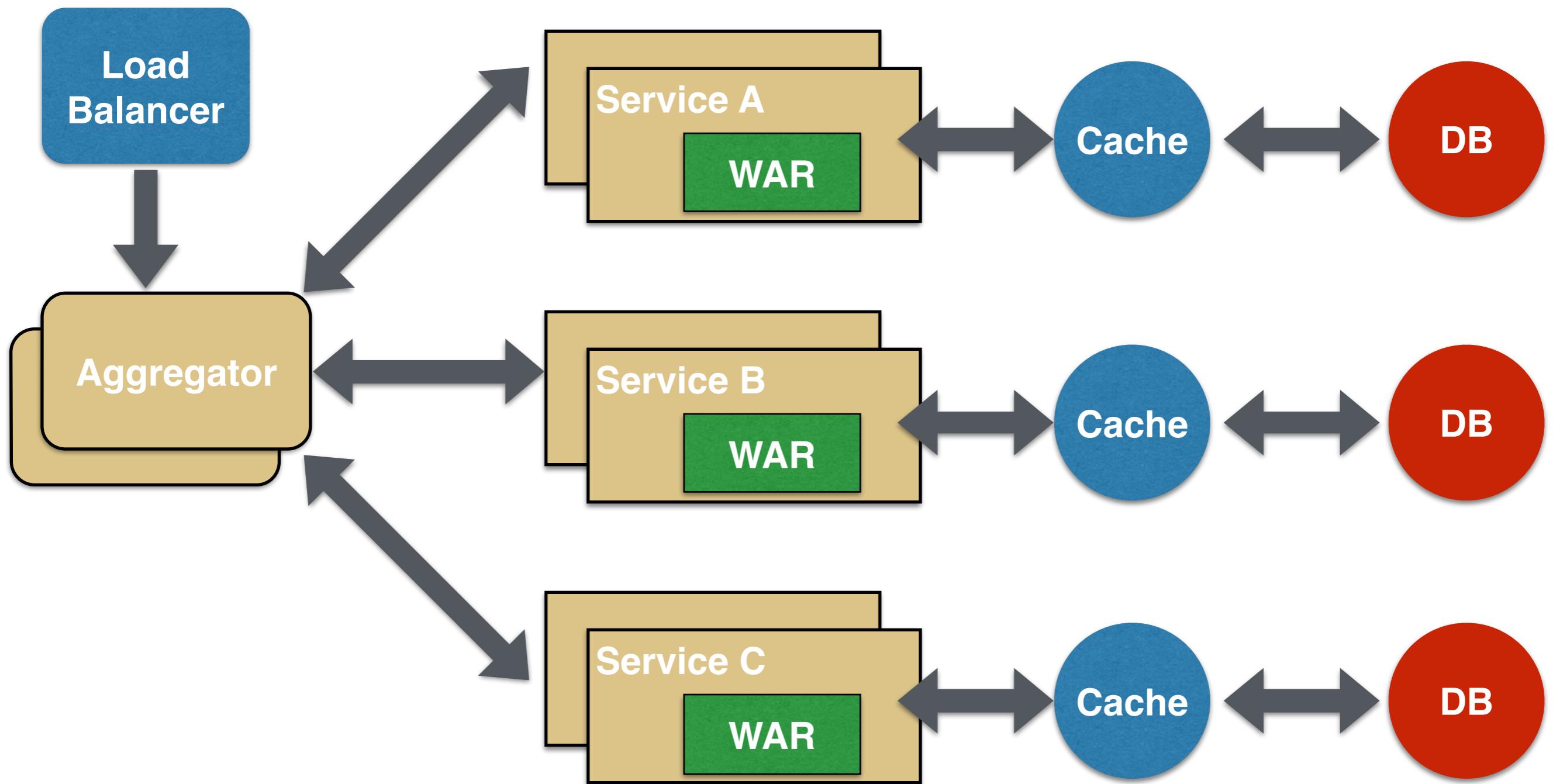
Strategies for decomposing

- Verb or usecase - e.g. Checkout UI
- Noun - e.g. Catalog product service
- Bounded context
- Single Responsible Principle - e.g. Unix utilities

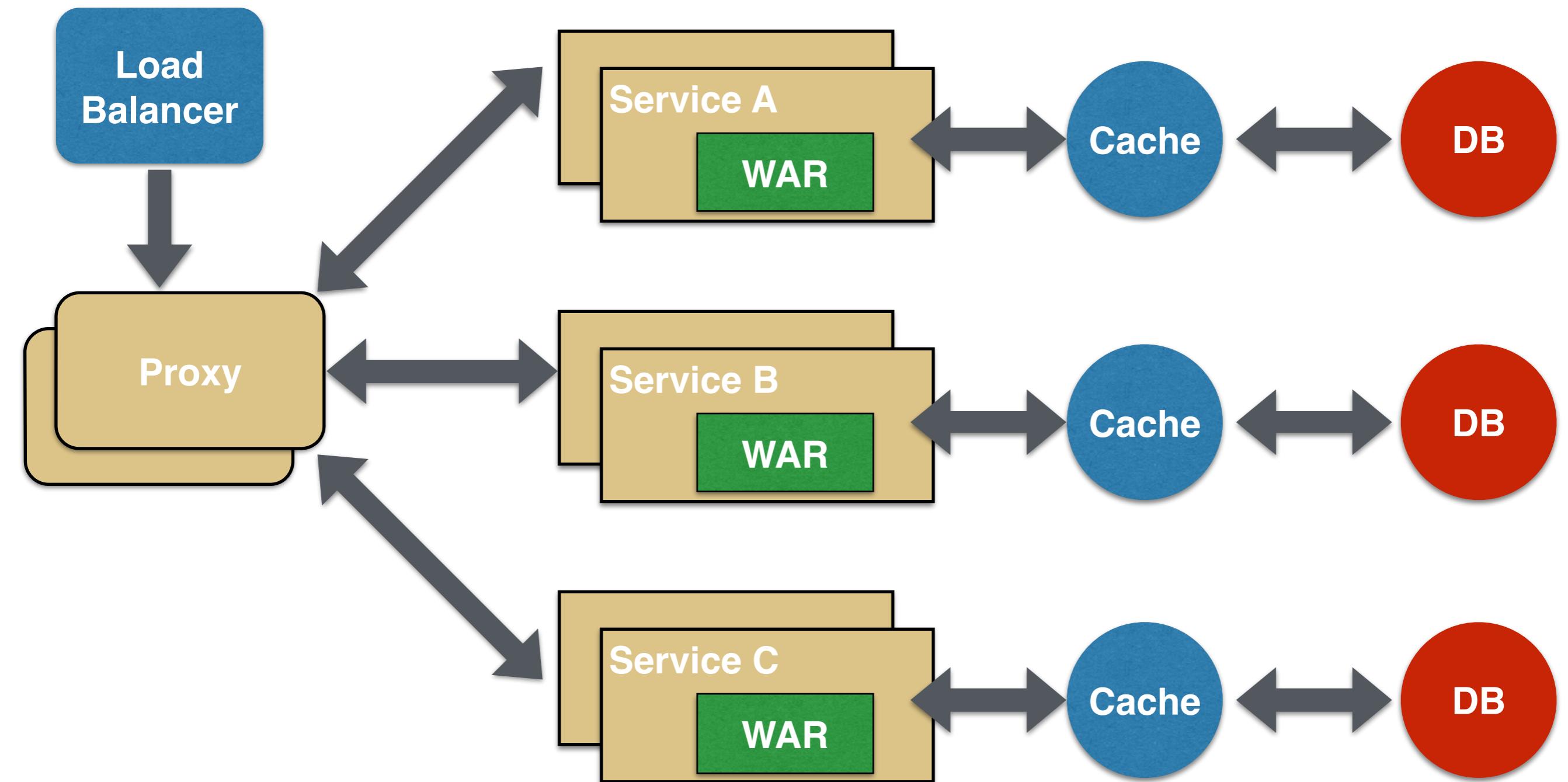
Towards microservices



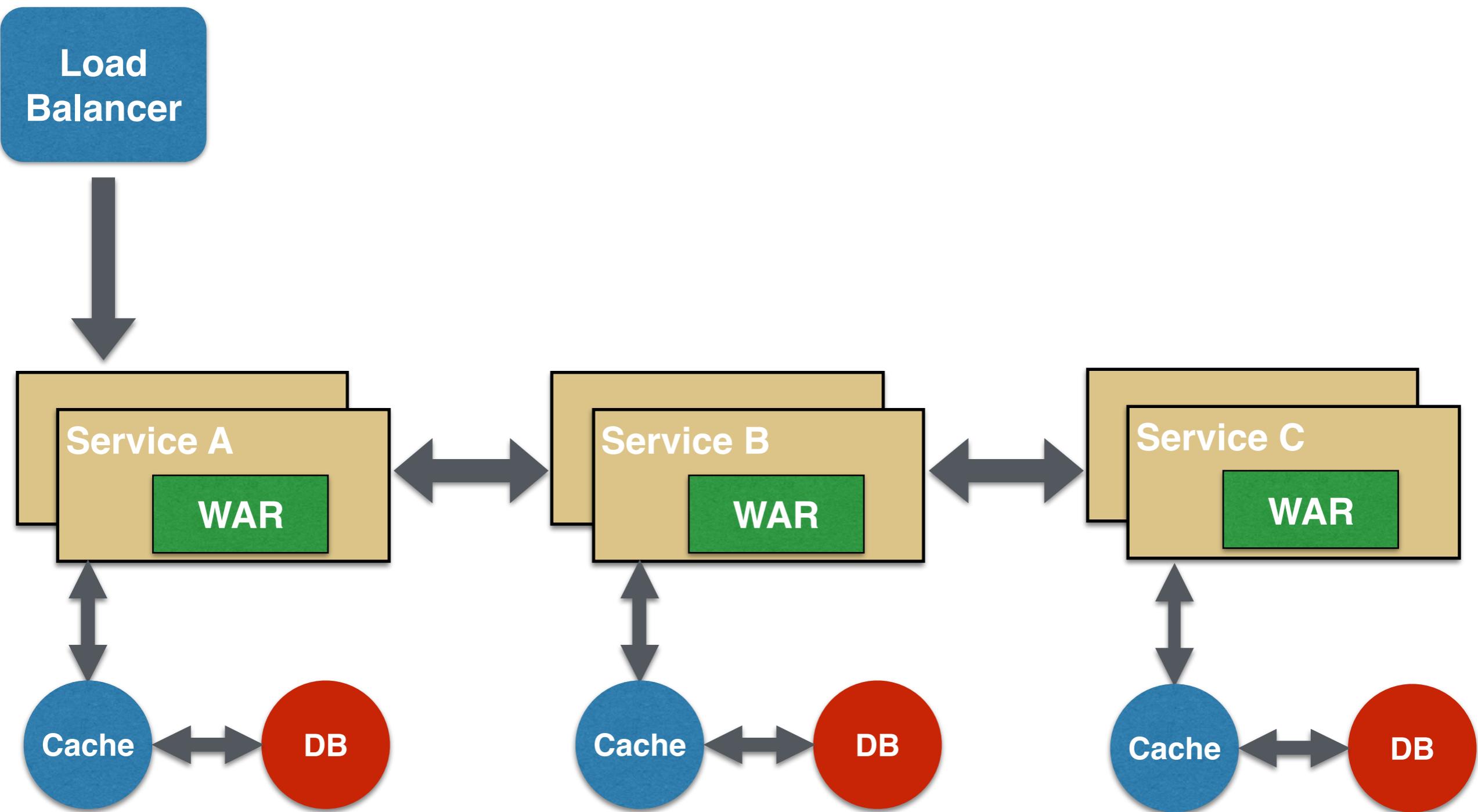
Aggregator Pattern #1



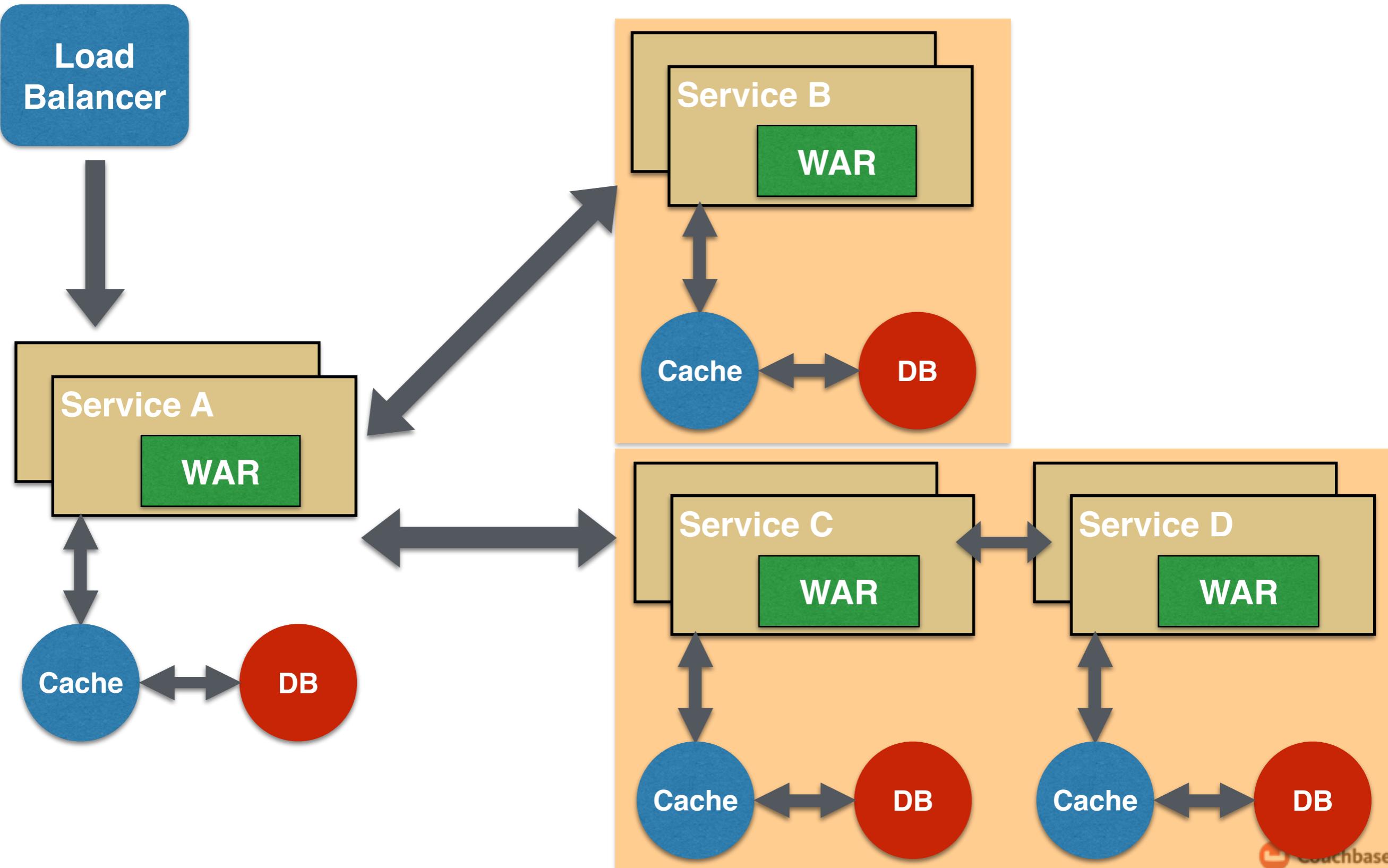
Proxy Pattern #2



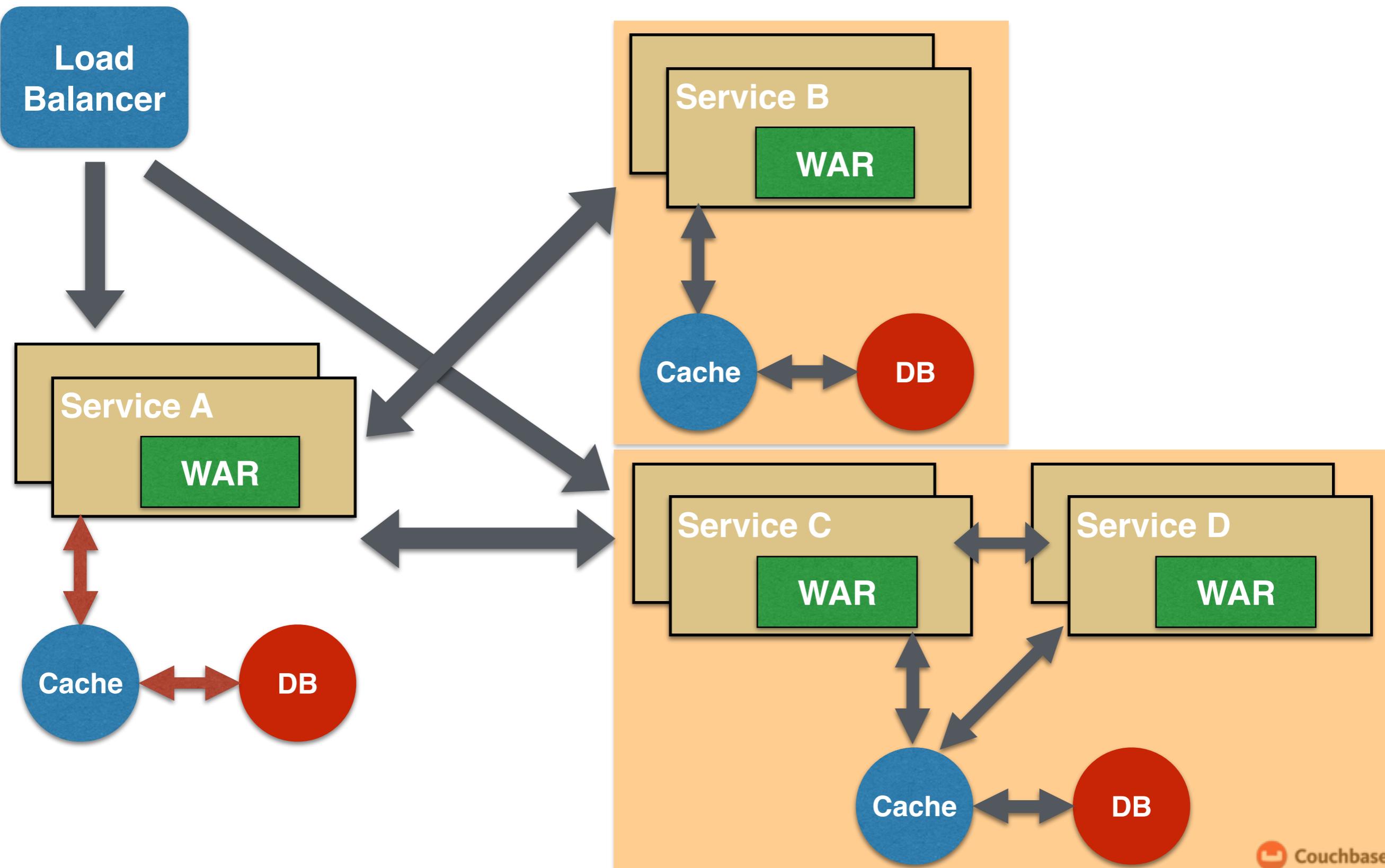
Chained Pattern #3



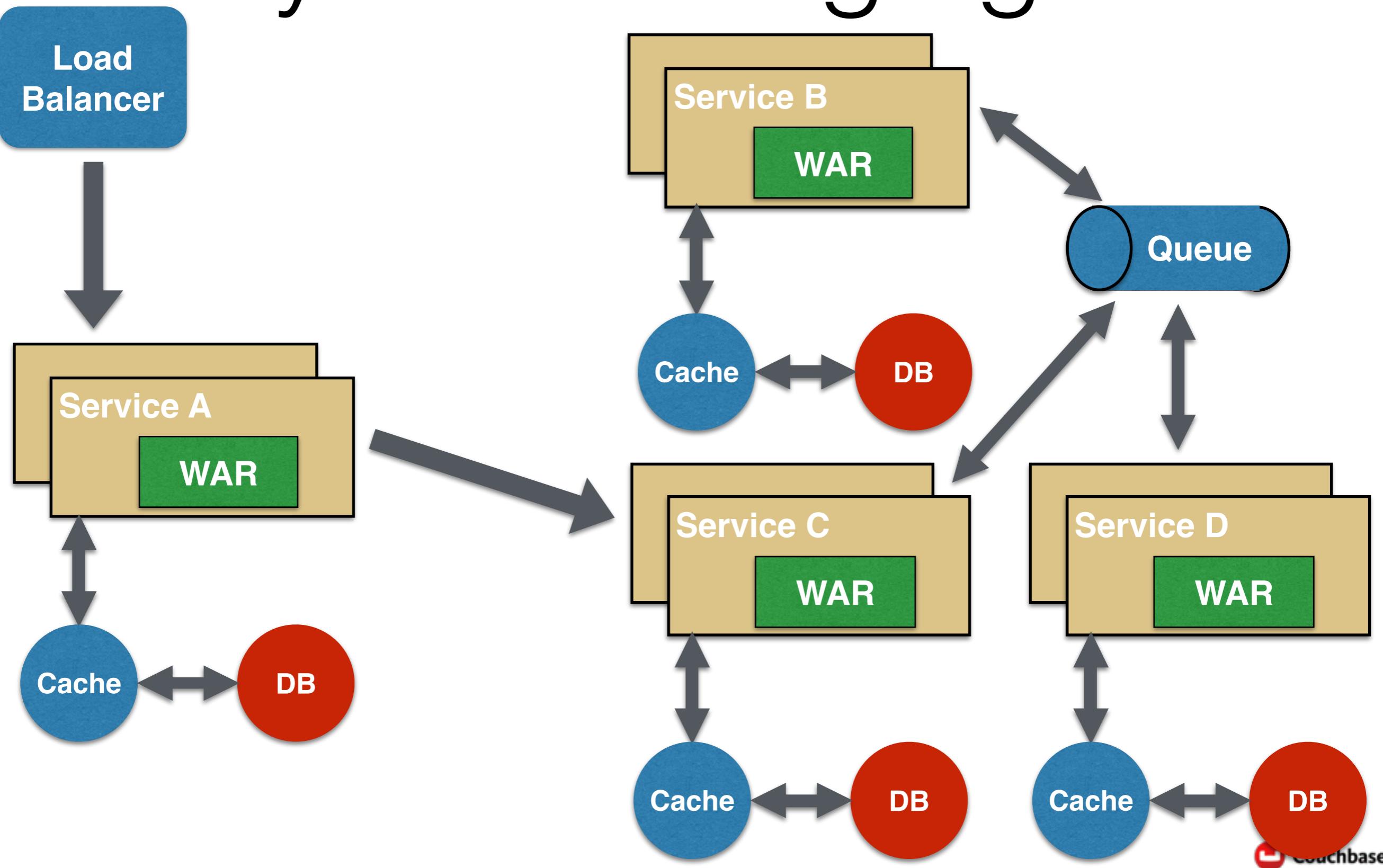
Branch Pattern #4



Shared Resources #5



Async Messaging #5





Refcard #215

Getting Started With Microservices

Design Patterns for Decomposing the Monolith

by Arun Gupta

Still re-deploying your entire application for one small update? Microservices deploy modular updates and increase the speed of application deployments.

Free PDF

 DOWNLOAD

 SAVE

 12.2k

<https://dzone.com/refcardz/getting-started-with-microservices>

Design Principles for Monoliths

- DDD
- SoC using MVC
- High cohesion, low coupling
- DRY
- CoC
- YAGNI

SAY MICROSERVICE



ONE MORE TIME

memegenerator.net

Advantages of microservices

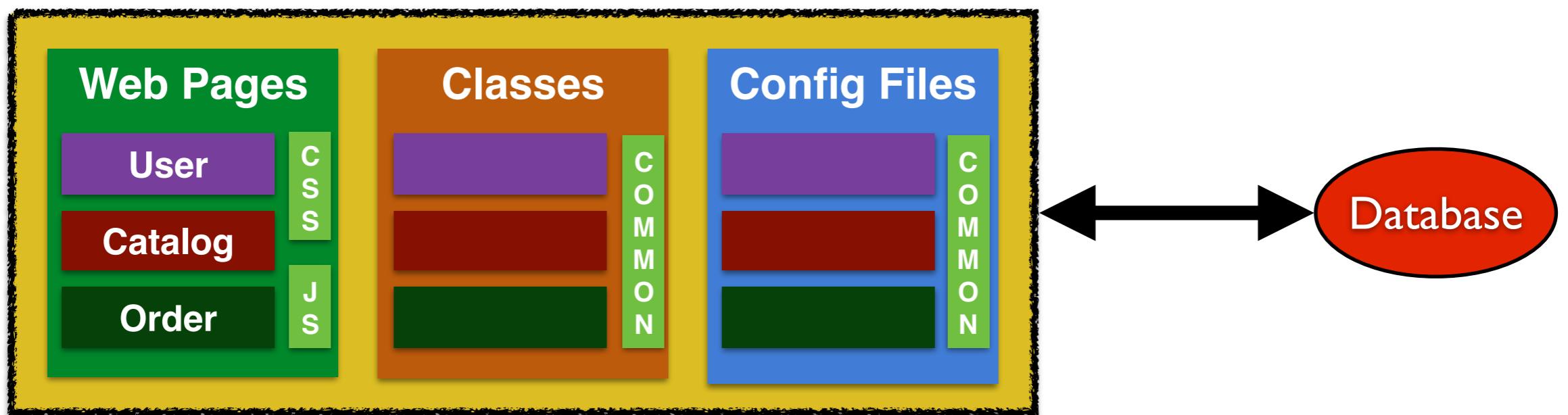
- Easier to develop, understand, maintain
- Starts faster than a monolith, speeds up deployments
- Local change can be easily deployed, great enabler of CD
- Each service can scale on X- and Z-axis
- Improves fault isolation
- Eliminates any long-term commitment to a technology stack
- Freedom of choice of technology, tools, frameworks

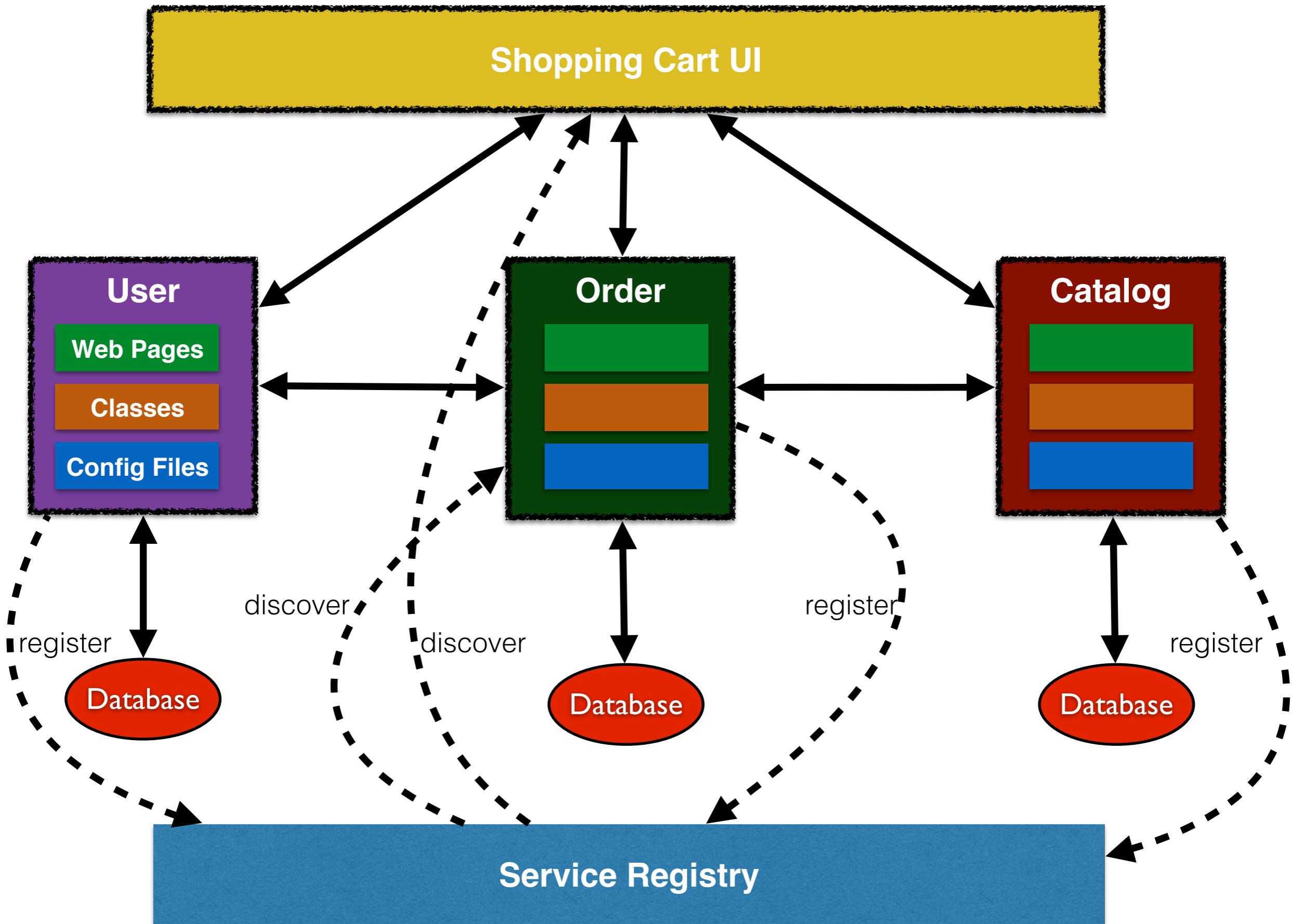
“If you can't build a [well-structured] monolith, what makes you think microservices are the answer?”

http://www.codingthearchitecture.com/2014/07/06/distributed_big_balls_of_mud.html

“If your monolith is a big ball of mud, your microservice will be a bag of dirt”

Arun Gupta





Monolith vs Microservice

	Monolith	Microservice
Archives	1	5 (Contracts, Order, User, Catalog, Web)
Web pages	8	8
Config Files	4 (persistence.xml, web.xml, load.sql, template.xhtml)	12 (3 per archive)
Classes	12	26 (Service registration/discovery, Application)
Archive Size	24 KB	~52 KB total

Service Registry/Discovery

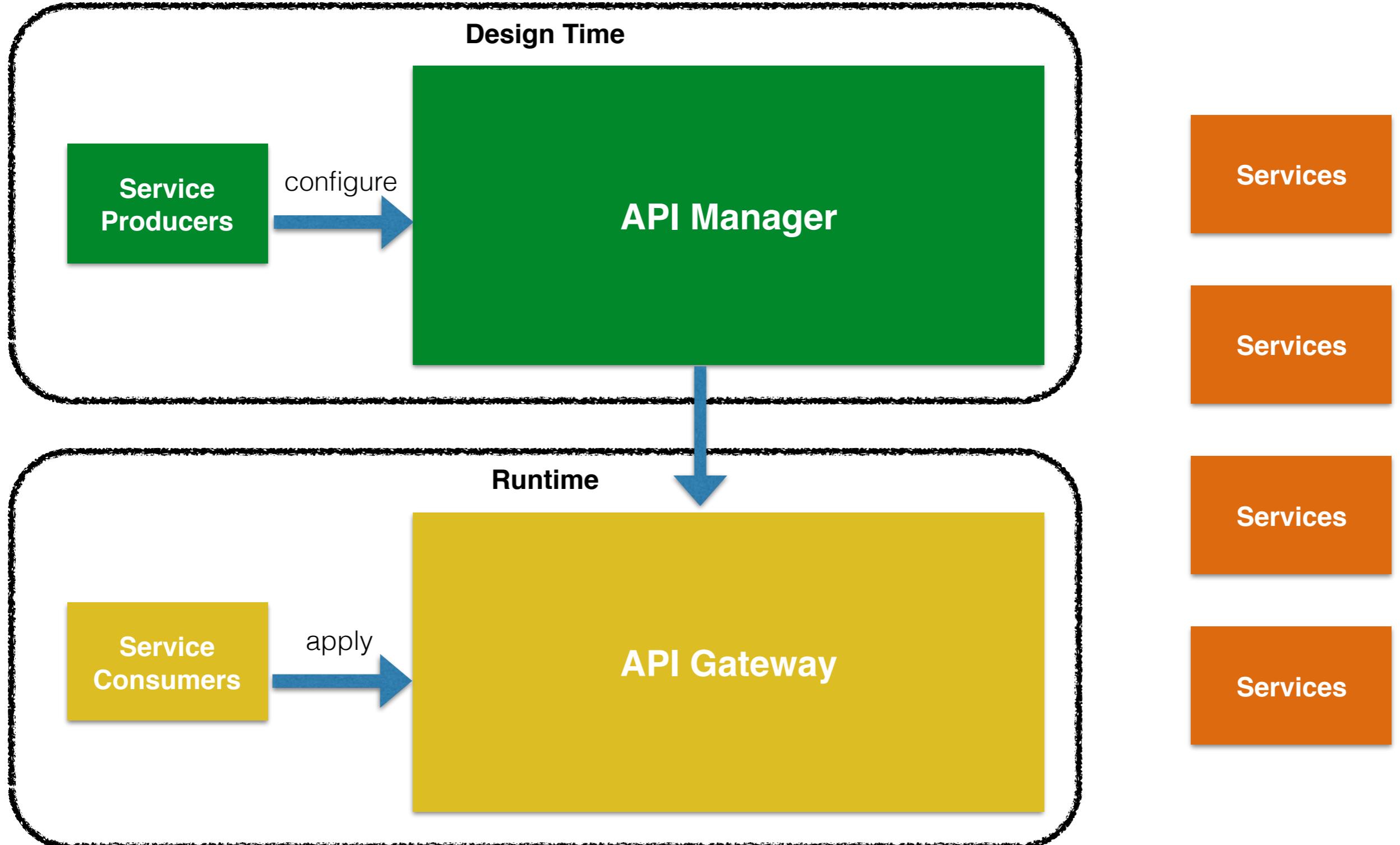
- ZooKeeper and Curator
- Snoop
- ...
- Kubernetes
- etcd
- Consul
- OSGi

Design Considerations

- UI and Full stack
 - Client-side composition (JavaScript?)
 - Server-side HTML generation (JSF?)
 - One service, one UI
- REST Services
- Event sequencing instead of 2PC
- API Management

API Management

- Centralized governance policy configuration
- Tracking of APIs and consumers of those APIs
- Easy sharing and discovery of APIs
- Leveraging common policy configuration across different APIs
 - Security, Caching, Rate limiting, Metrics, Billing, ...

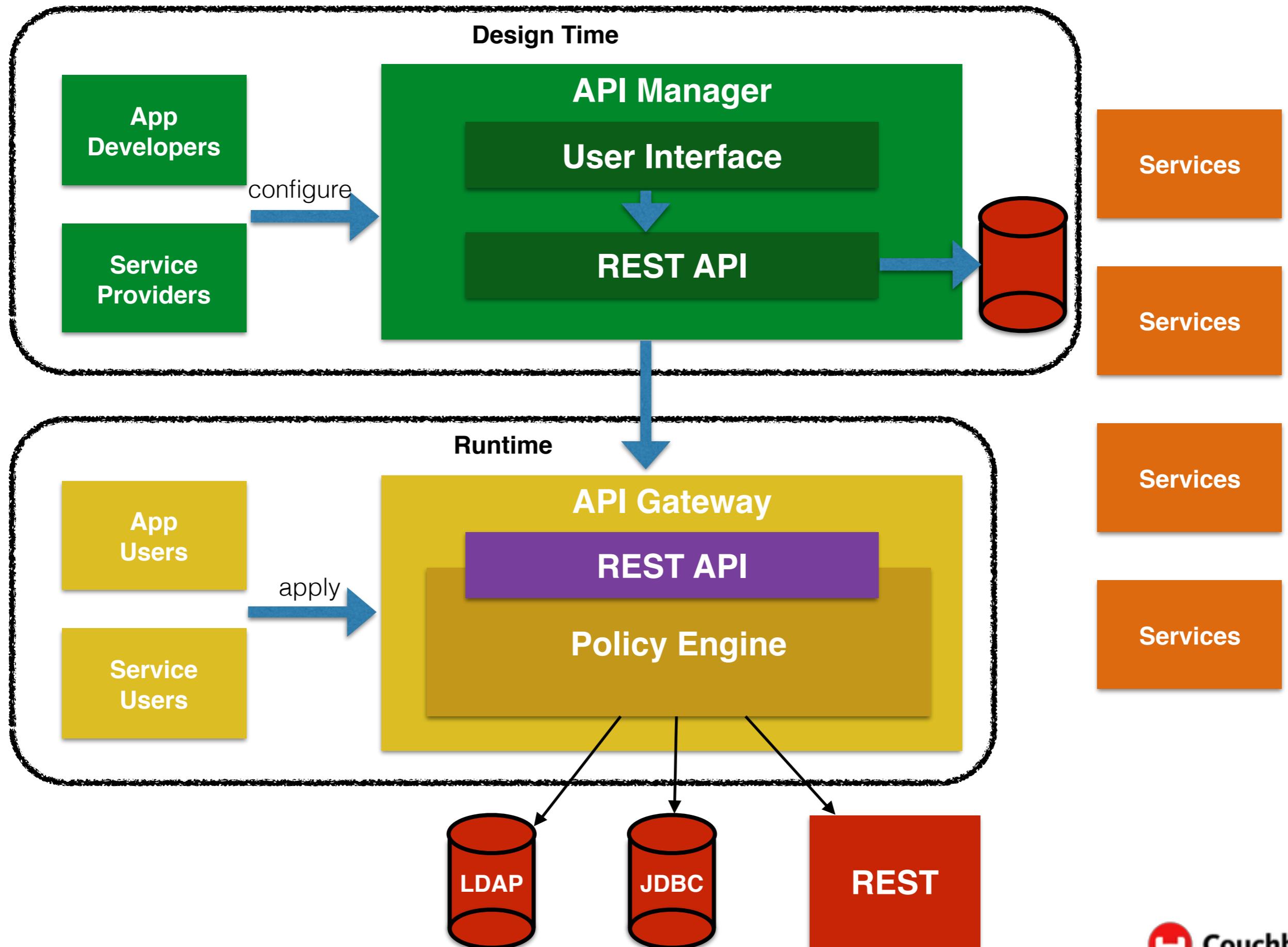


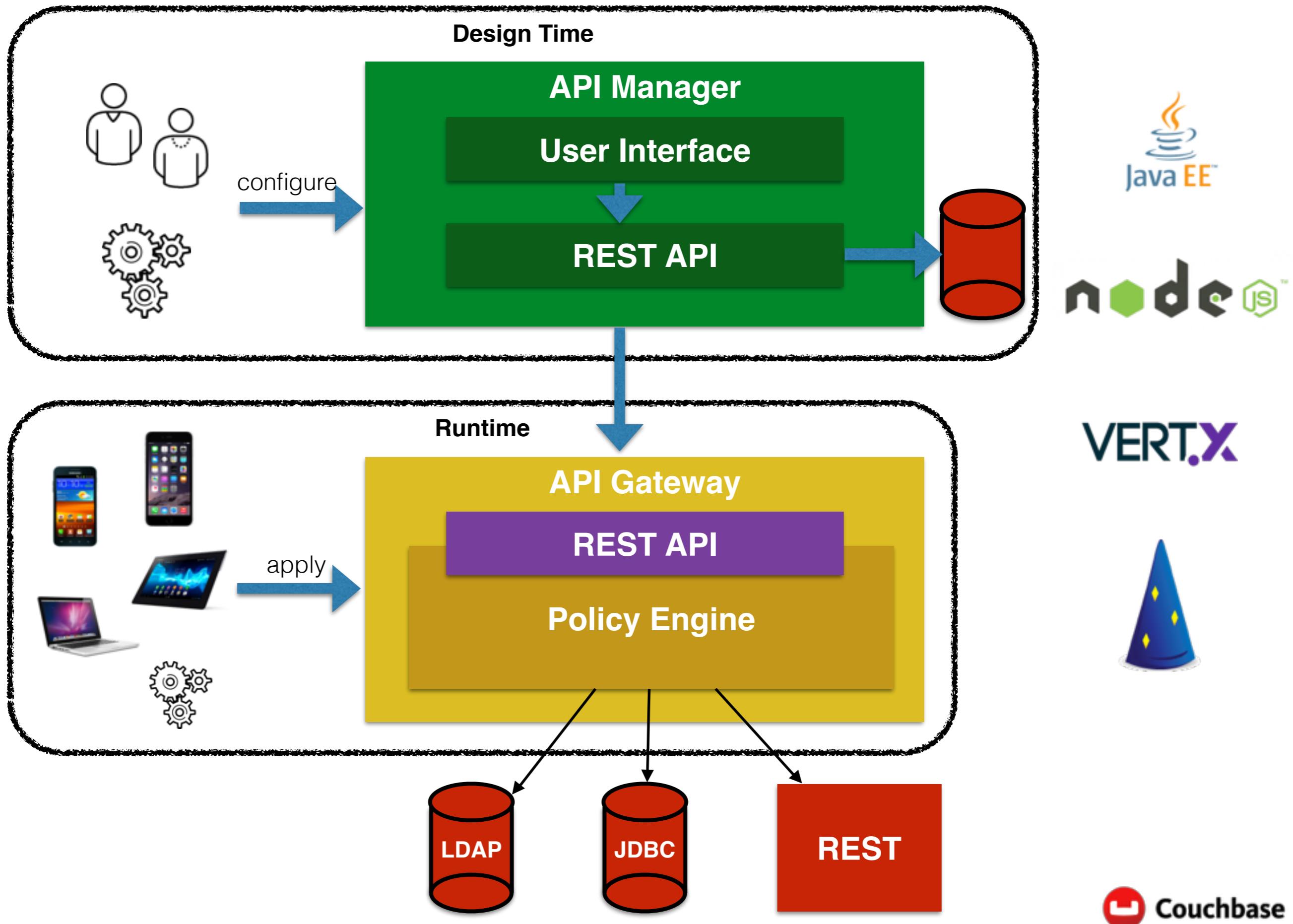
API Manager

- Configure and manage APIs
 - **Organization:** Top-level container concept
 - **Policies:** Unit of work executed at runtime to implement API governance
 - **Plans:** Set of policies that define a Service, e.g. Gold, Silver, ...
 - **Services:** External API being managed
 - **Applications:** Consumer of an API
 - **Contracts:** Link between Application and Service through a Plan
 - **Policy Chains:** Ordered sequence of policies

API Gateway

- Apply policies configured during management
 - Publish a Service
 - Register an Application (with Service Contracts)
 - Retire a Service
 - Unregister an Application



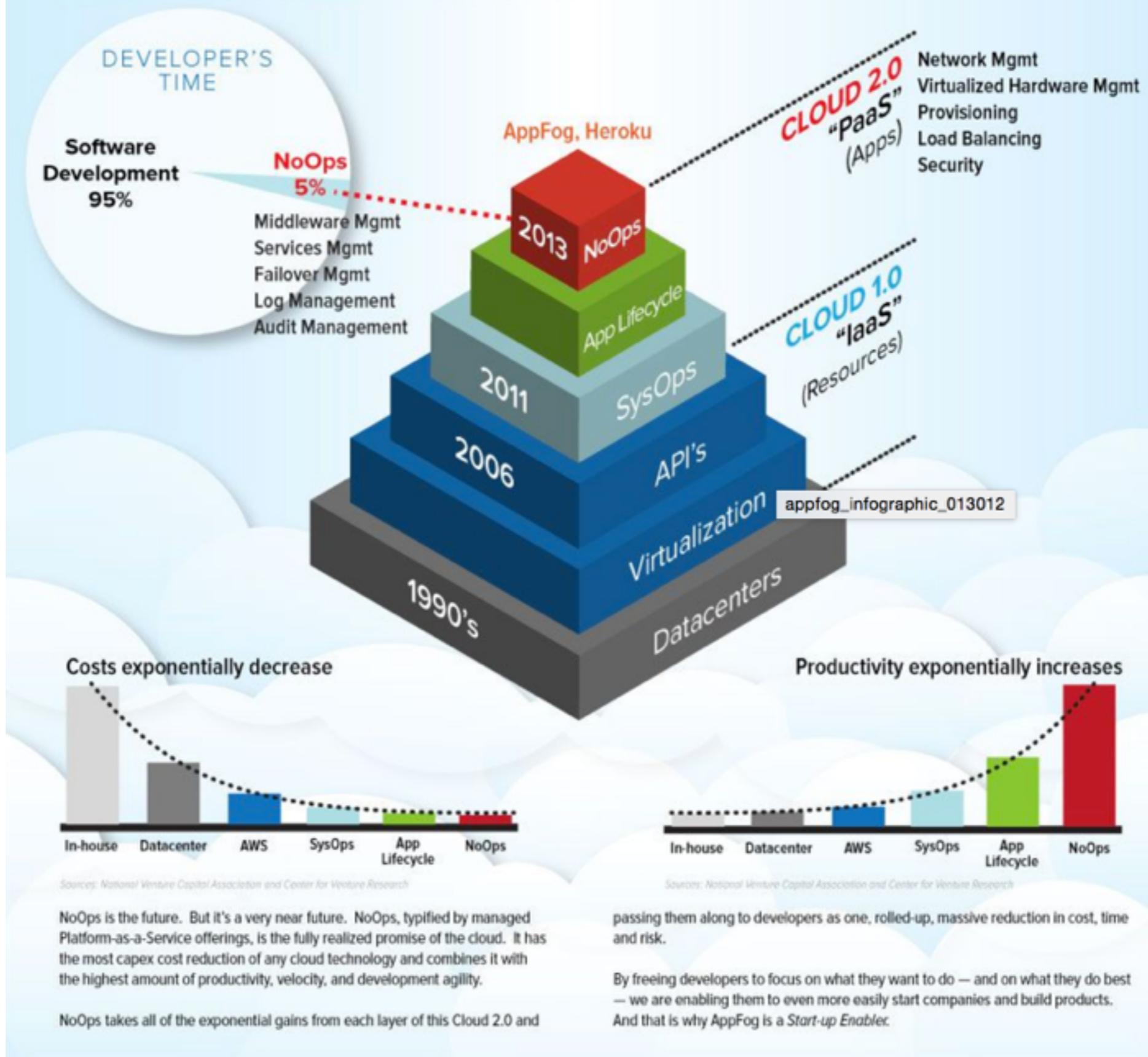


NoOps

- Service replication (Kubernetes)
- Dependency resolution (Nexus)
- Failover (Circuit Breaker)
- Resiliency (Circuit Breaker)
- Service monitoring, alerts and events (ELK)

2013: A bright NoOps future

So where does this all lead? The end-game is NoOps. Where building and running an app is purely a developer process — and where developers are not having to spend time doing Ops work.



Data Strategy

- Private tables-, schema-, or database-per-service
- No sharing tables
 - Only one service writes to a table(s), read-only by others
- No transactions across databases
- Logical transactions in application

Event Sourcing

- State of the application is defined by a sequence of events

store_id	store_version	subject_id	event_type	event
13b710cd-bd86-6ad1-f936-4be6e4926b7f	1428928602343		CurrencyWasSet	{"value": "GBP"}
13b710cd-bd86-6ad1-f936-4be6e4926b7f	1428928619507		TermsAndConditionsEnabled	{"value": true}
13b710cd-bd86-6ad1-f936-4be6e4926b7f	1428928622014		TermsAndConditionsWasSet	{"value": "aaaaaaa"}
13b710cd-bd86-6ad1-f936-4be6e4926b7f	1428928643605		CurrencyWasSet	{"value": "USD"}
13b710cd-bd86-6ad1-f936-4be6e4926b7f	1428928651105		CurrencyWasSet	{"value": "GBP"}
13b710cd-bd86-6ad1-f936-4be6e4926b7f	1428928803500		RefundCancellationPolicyEnabled	{"value": true}
13b710cd-bd86-6ad1-f936-4be6e4926b7f	1428928806991		RefundCancellationPolicyWasSet	{"value": "hhhhhhh"}
13b710cd-bd86-6ad1-f936-4be6e4926b7f	1428928809555		TermsAndConditionsWasSet	{"value": "kkkkkkk"}
13b710cd-bd86-6ad1-f936-4be6e4926b7f	1428928832951		TermsAndConditionsWasSet	{}
13b710cd-bd86-6ad1-f936-4be6e4926b7f	1428928849827		RefundCancellationPolicyWasSet	{"value": "kkkkkkk"}

CQRS

- Traditional: CRUD using same data model
- Command Query Responsibility Segregation
 - Different model to update and read information
 - **Command**: change the state of system, do not return value
 - **Query**: return state, do not change state

CQRS

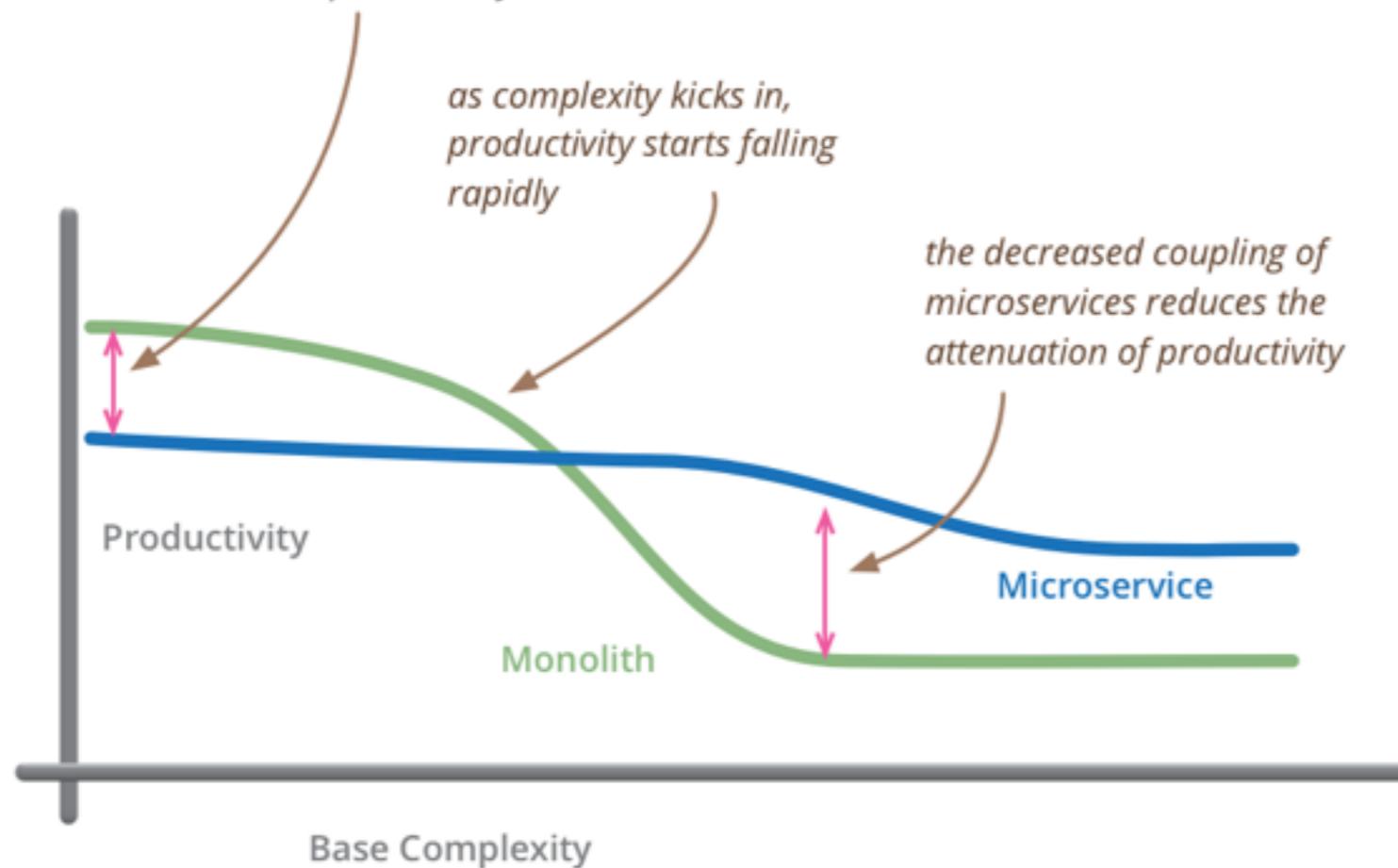


Drawbacks of microservices

- Additional complexity of distributed systems
- Significant operational complexity, need high-level of automation
- Rollout plan to coordinate deployments
- Slower ROI, to begin with

Microservice Premium

for less-complex systems, the extra baggage required to manage microservices reduces productivity

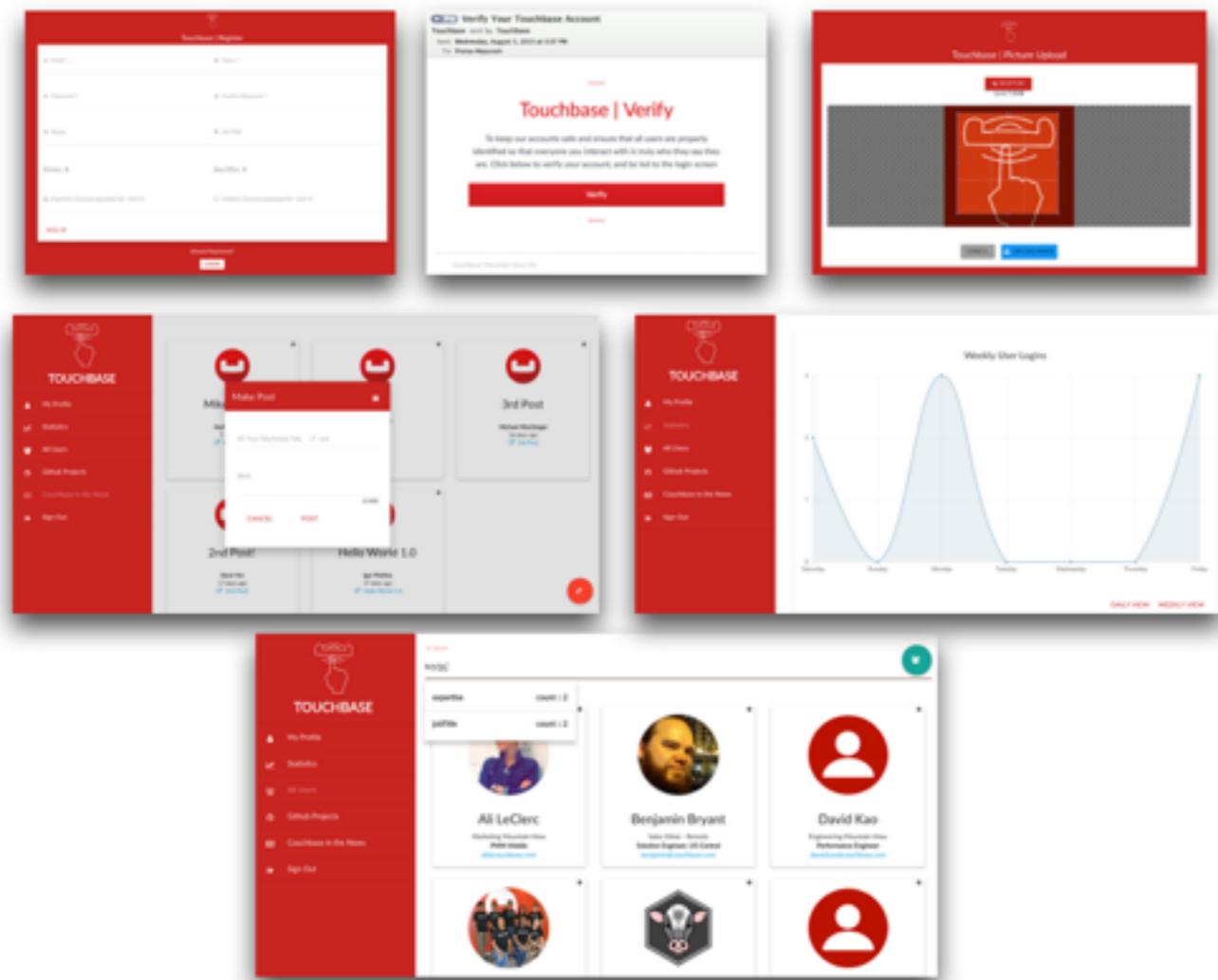


“don’t even consider microservices unless you have a system that’s too complex to manage as a monolith”

but remember the skill of the team will outweigh any monolith/microservice choice

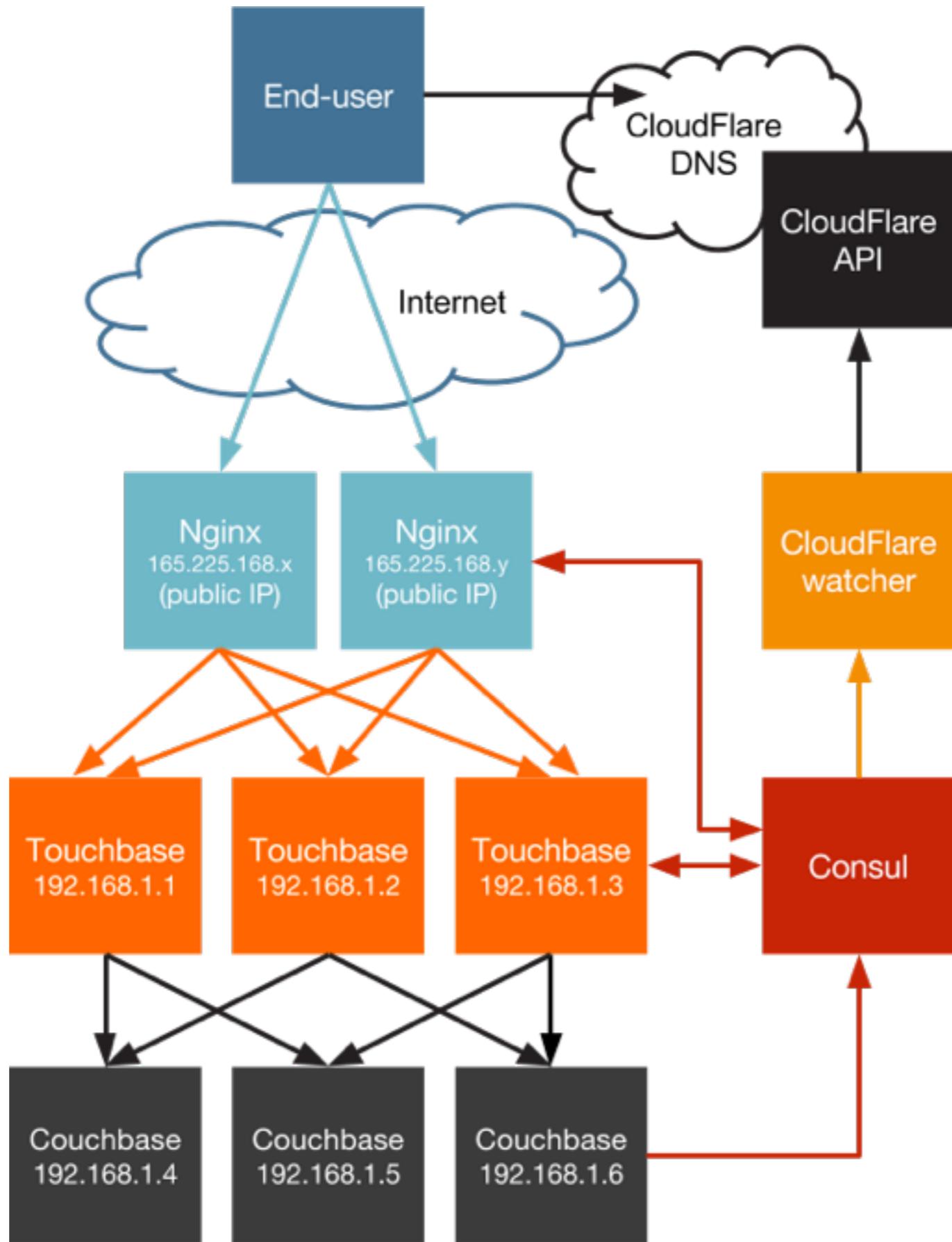
Touchbase

- Social network platform built using CEAN
- <https://github.com/couchbaselabs/touchbase/>



ContainerBuddy

- Help containerize existing applications
 - Shim written using Go
- Registers the application with Consul
 - Sends TTL health check
- Checks for changes in dependent/upstream services



```
1 # Consul as a service discovery tier
2 consul:
3     image: program/consul:latest
4     command: -server -bootstrap -ui-dir /ui
5     restart: always
6     mem_limit: 128m
7     ports:
8         - 53
9         - 8300
10        - 8301
11        - 8302
12        - 8400
13        - 8500
14     dns:
15         - 127.0.0.1
16
17 # Manually bootstrap the first instance, then...
18 # Scale this tier and each additional container/instance will
19 # self-configure as a member of the cluster
20 couchbase:
21     image: misterbisson/triton-couchbase:enterprise-4.0.0-1
22     restart: always
23     links:
24         - consul
25     mem_limit: 4g
26     ports:
27         - 8091
28         - 8092
29         - 8093
30         - 11207
31         - 11210
32         - 11211
33         - 18091
34         - 18092
35     environment:
36         - CONSUL_HOST=http://consul:8500
37         - COUCHBASE_SERVICE_NAME=couchbase
38         - COUCHBASE_USER=Administrator
39         - COUCHBASE_PASS=password
40
41
42     # the main application
43 touchbase:
44     image: 0x74696d/triton-touchbase
45     links:
46         - consul
47     mem_limit: 1g
48     ports:
49         - 3000
50     restart: always
51     command: >
52         /opt/containerbuddy/containerbuddy
53         -config file:///opt/containerbuddy/touchbase.json
54         /usr/local/bin/run-touchbase.sh
55
56     # Nginx as a load-balancing tier and reverse proxy
57 nginx:
58     image: 0x74696d/triton-touchbase-demo-nginx
59     mem_limit: 512m
60     ports:
61         - 80
62     links:
63         - consul:consul
64     restart: always
65     environment:
66         - CONTAINERBUDDY=file:///opt/containerbuddy/nginx.json
67     command: >
68         /opt/containerbuddy/containerbuddy
69         nginx -g "daemon off;"
70
71     # Support dynamic DNS for the load balancing tier
72     # https://www.joyent.com/blog/automatic-dns-updates-with-containerbuddy
73 cloudfflare:
74     image: 0x74696d/triton-cloudflare
75     mem_limit: 128m
76     links:
77         - consul:consul
78     restart: always
79     env_file: .env
80     command: >
81         /opt/containerbuddy/containerbuddy
82         -config file:///opt/containerbuddy/cloudflare.json
```

References

- github.com/arun-gupta/microservices
- github.com/javaee-samples/docker-java
- dzone.com/refcardz/getting-started-with-microservices