

Refactoring your Java EE applications using Microservices and Containers

Arun Gupta
Vice President, Developer Advocacy
Couchbase
[@arungupta](https://twitter.com/arungupta), blog.arungupta.me
arun@couchbase.com



O'REILLY®

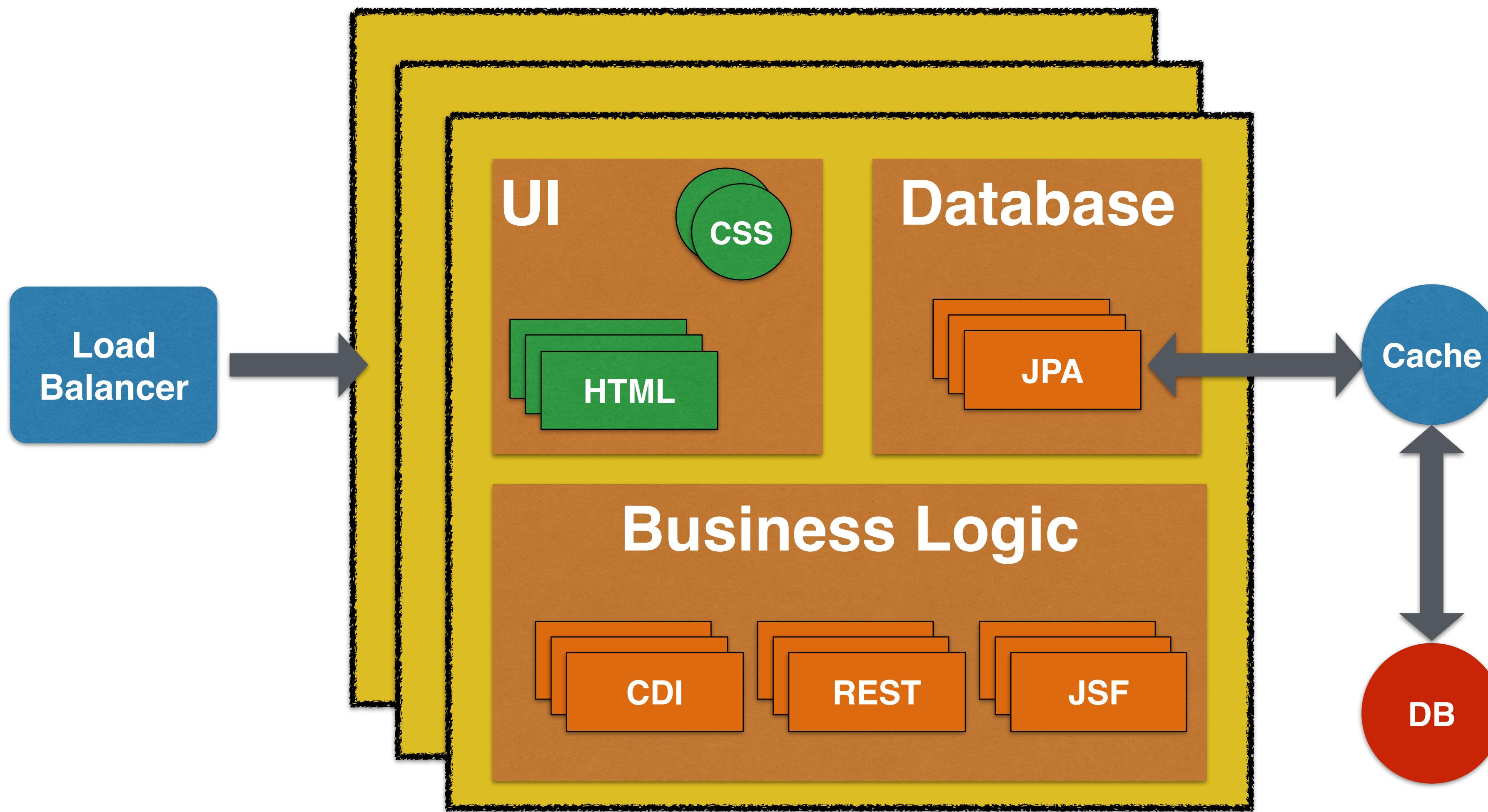


Minecraft Modding with Forge

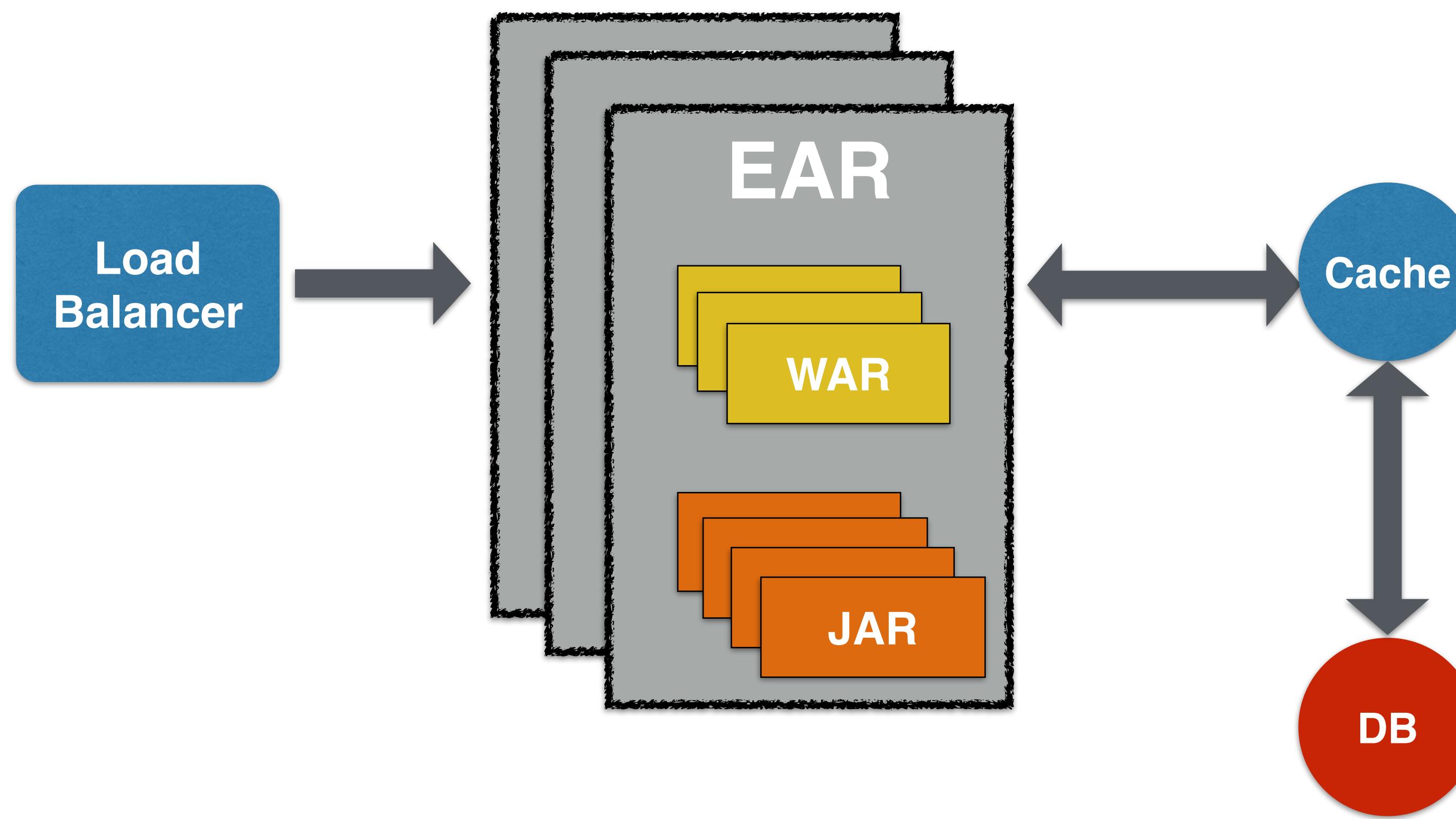
A FAMILY-FRIENDLY GUIDE TO BUILDING FUN MODS IN JAVA

Arun Gupta & Aditya Gupta

Monolith Application



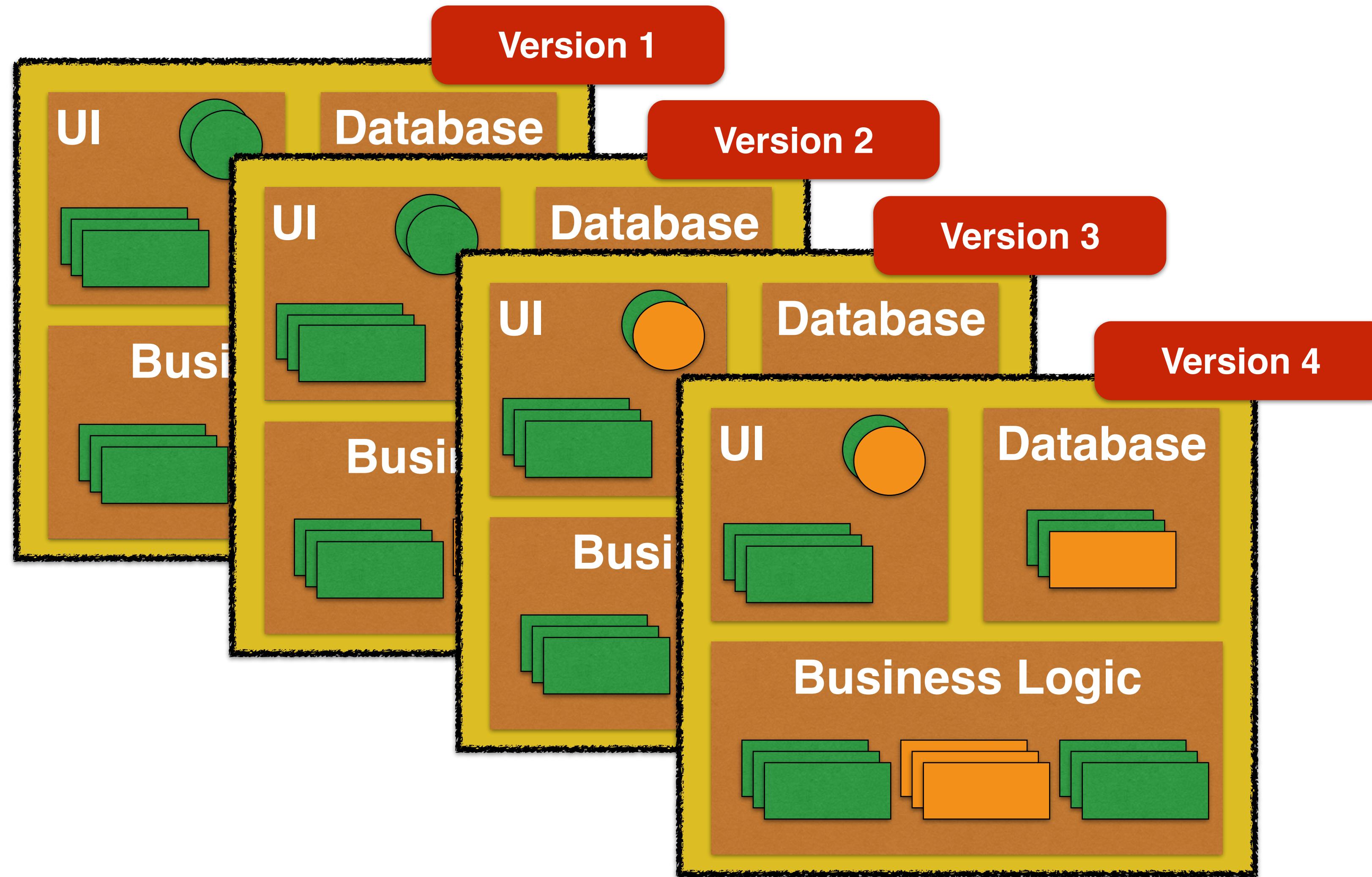
Monolith Application



Advantages of Monolith

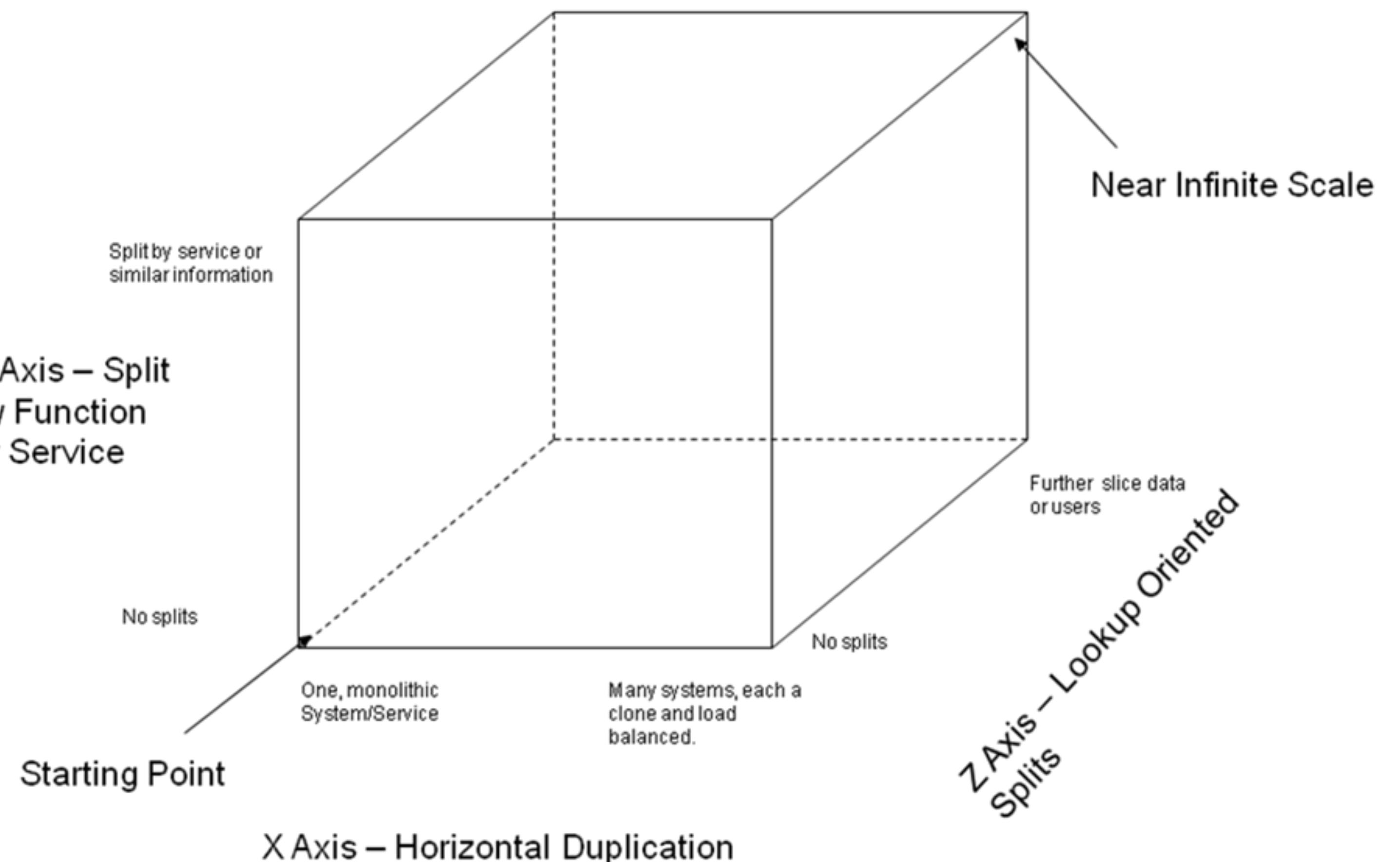
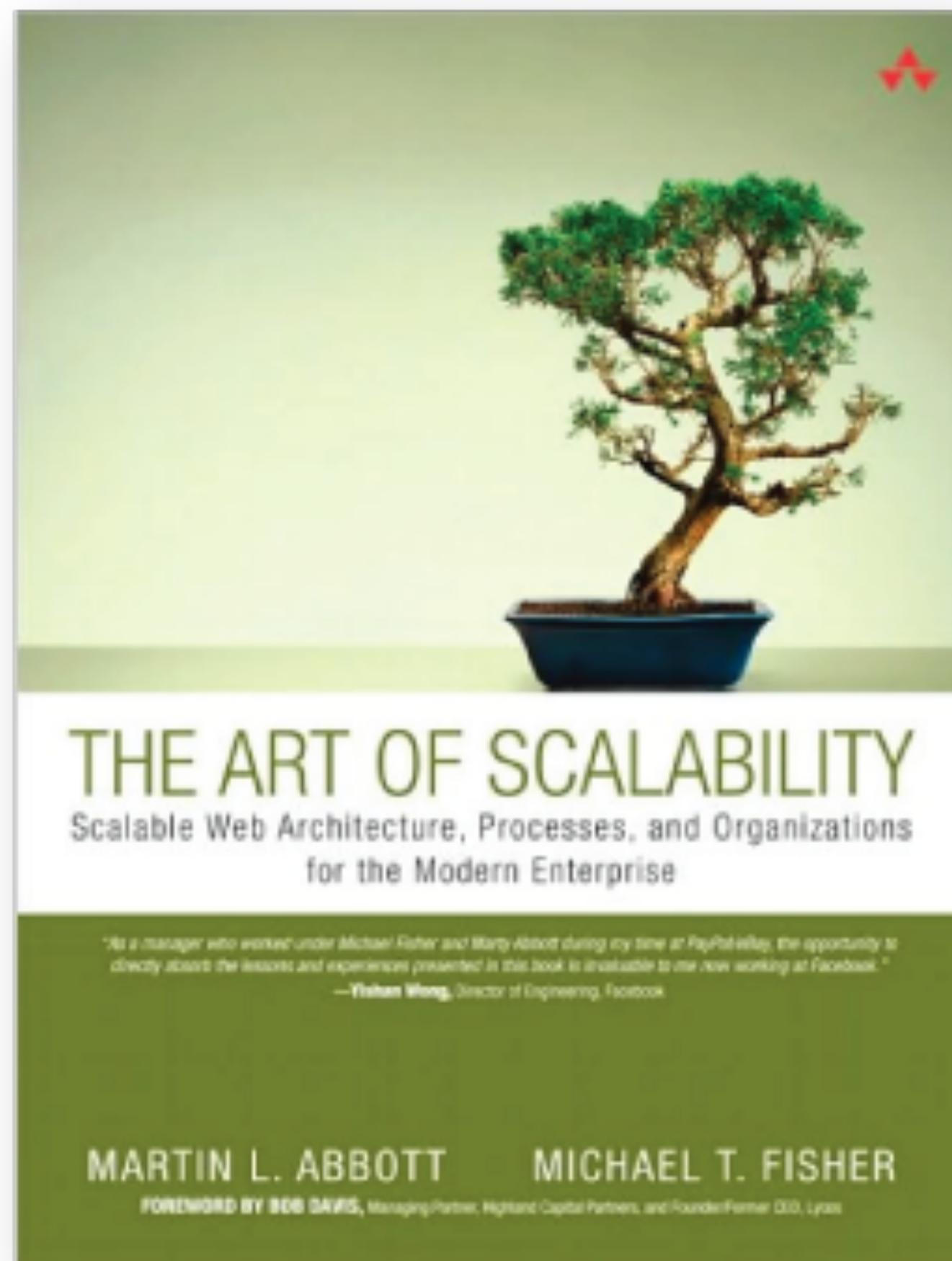
- Typically packaged in a single .ear
- Easy to test (all required services are up)
- Simple to develop

Monolith Version Management



Disadvantages of Monolith

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

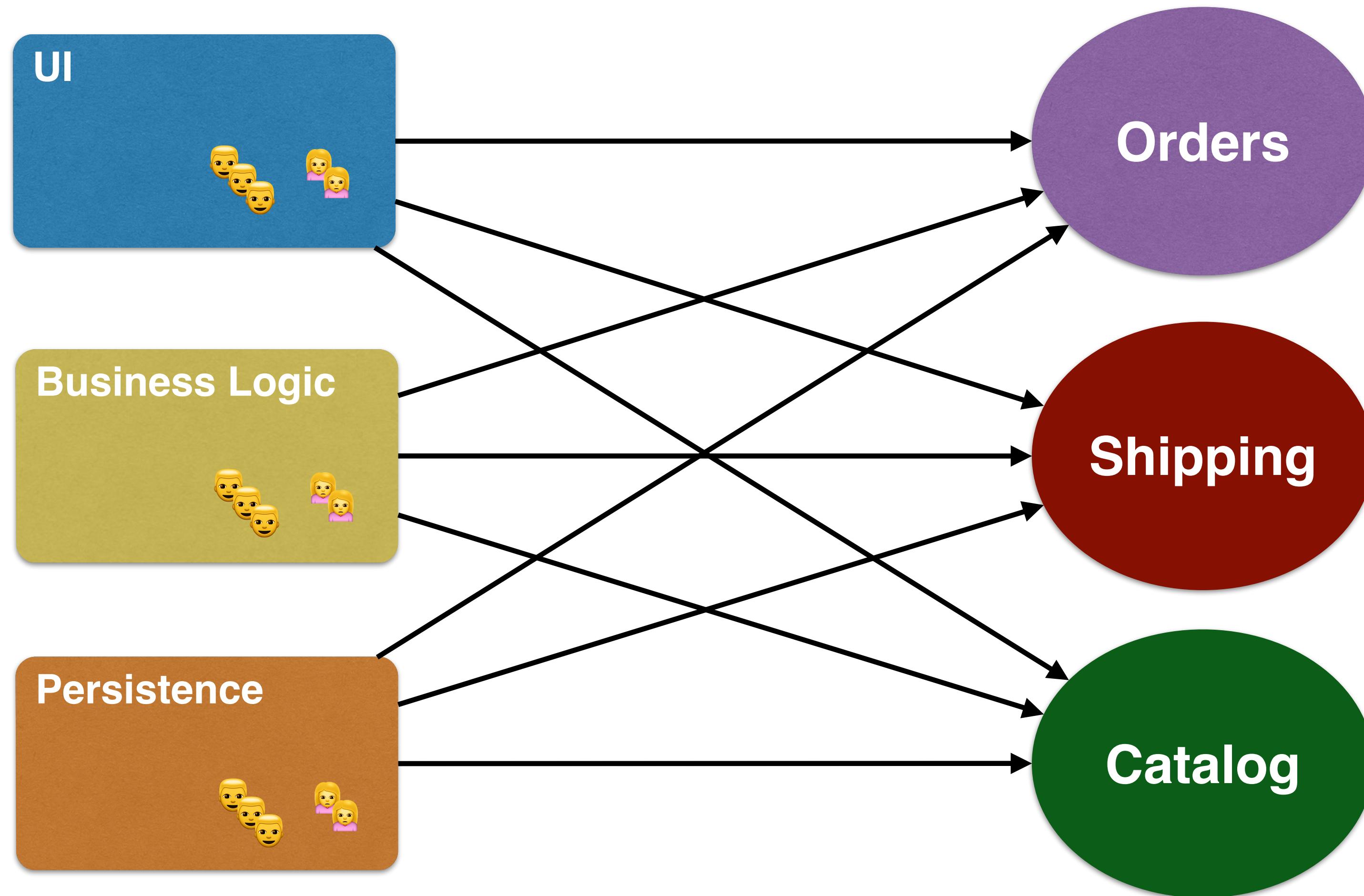


An ***architectural approach***, that emphasizes the ***decomposition of applications*** into ***single-purpose, loosely coupled*** services managed by ***cross-functional teams***, for delivering and maintaining ***complex software systems*** with the velocity and quality required by today's ***digital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

I DONT ALWAYS
BUILD EVERYTHING



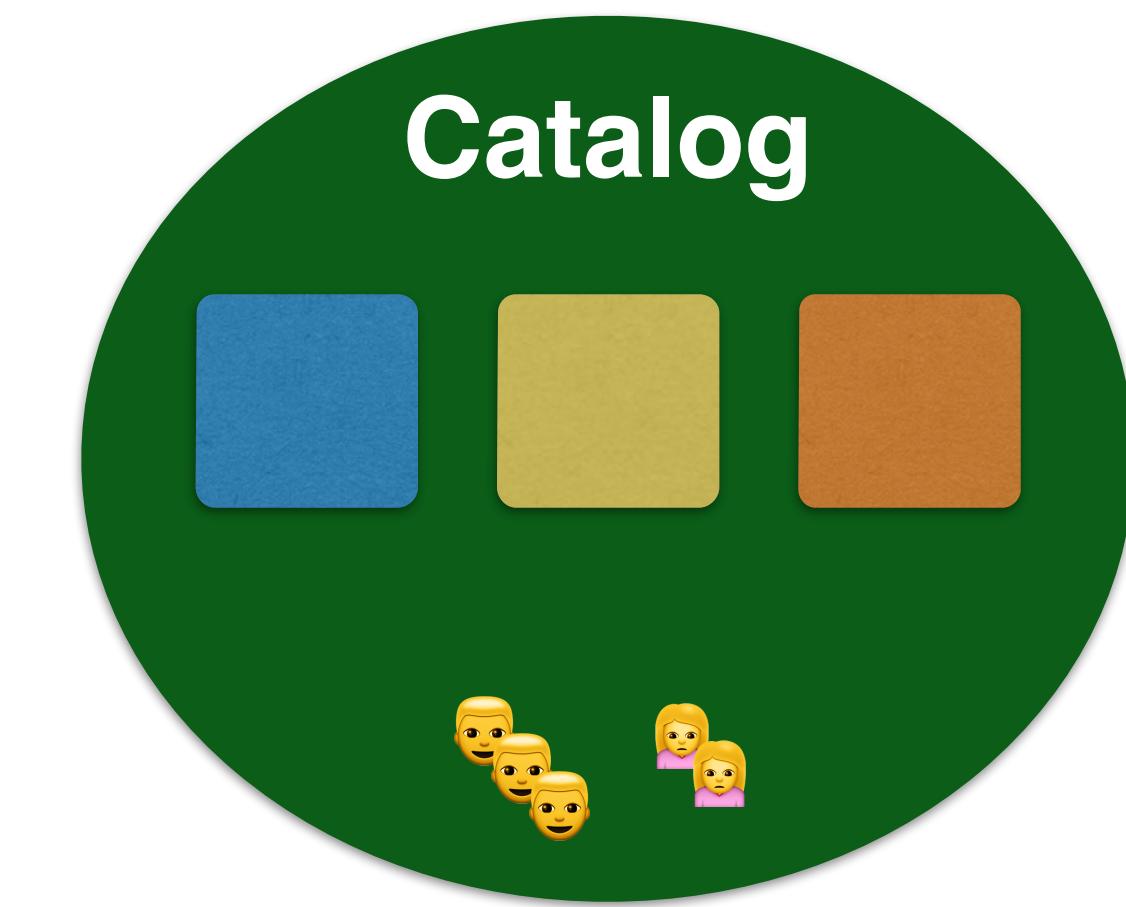
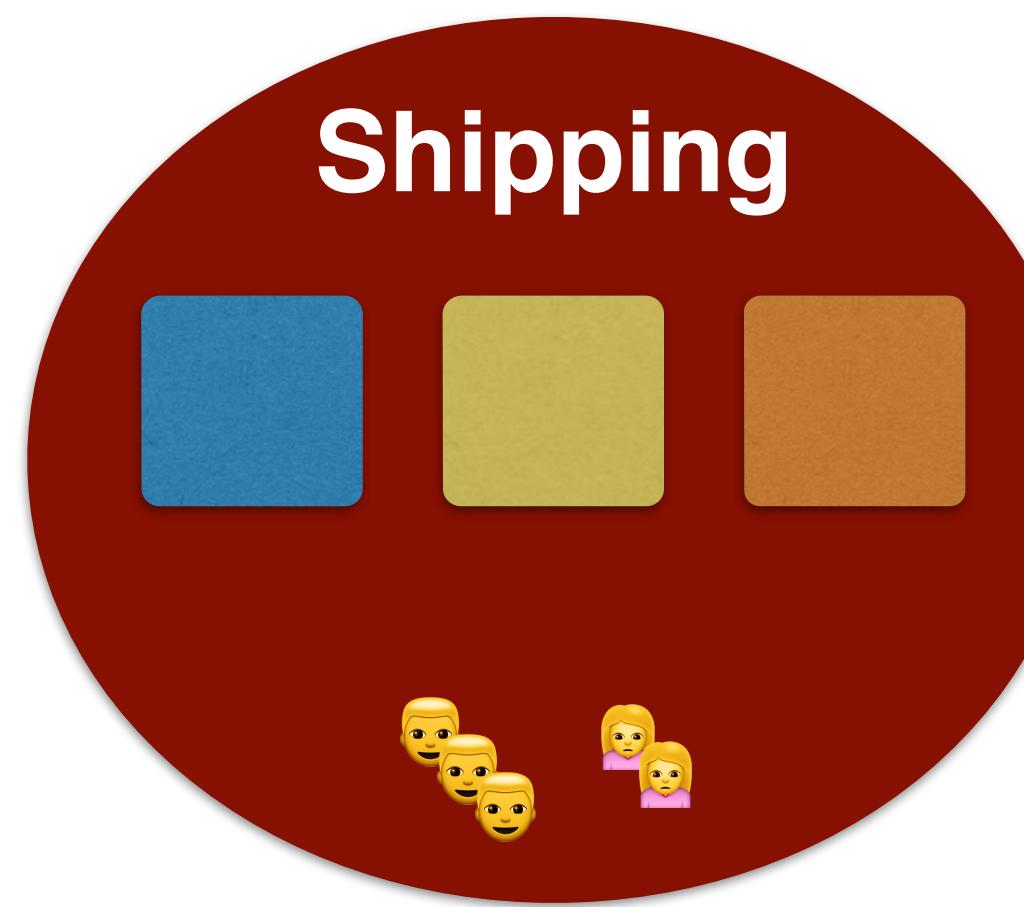
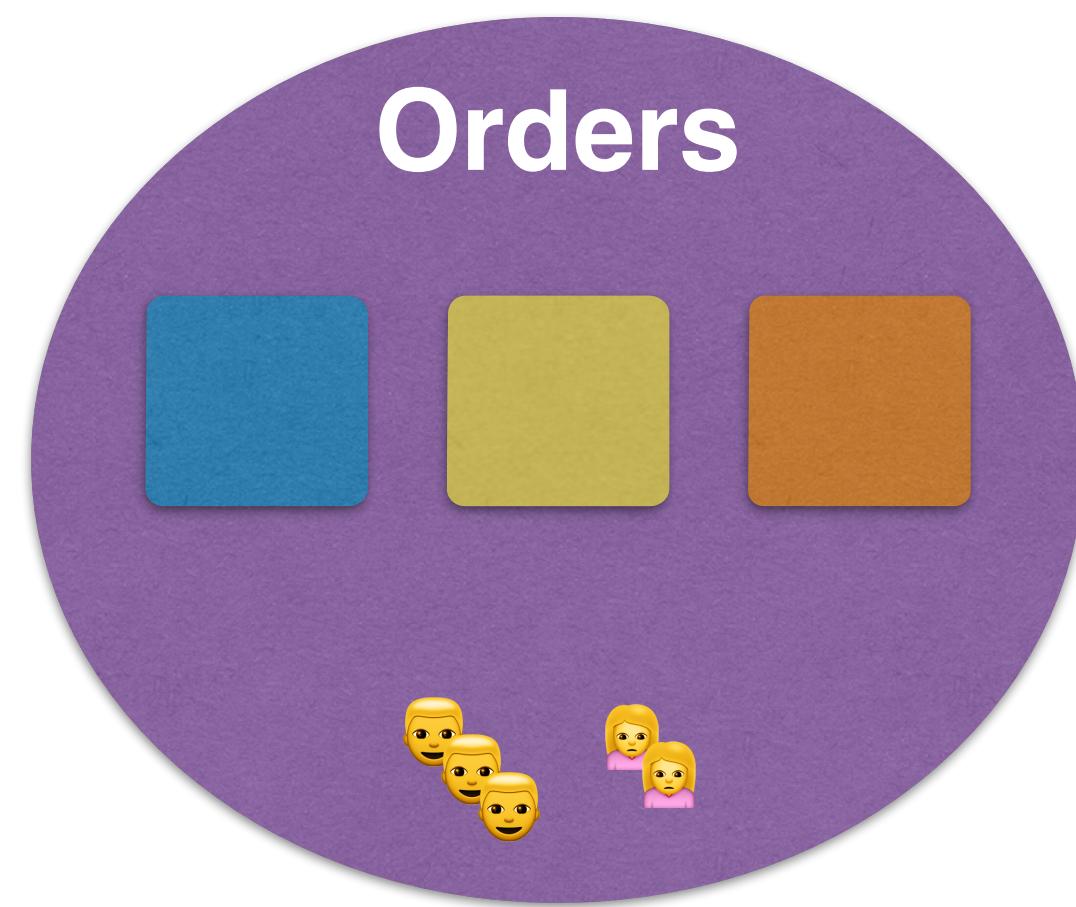
imgflip.com



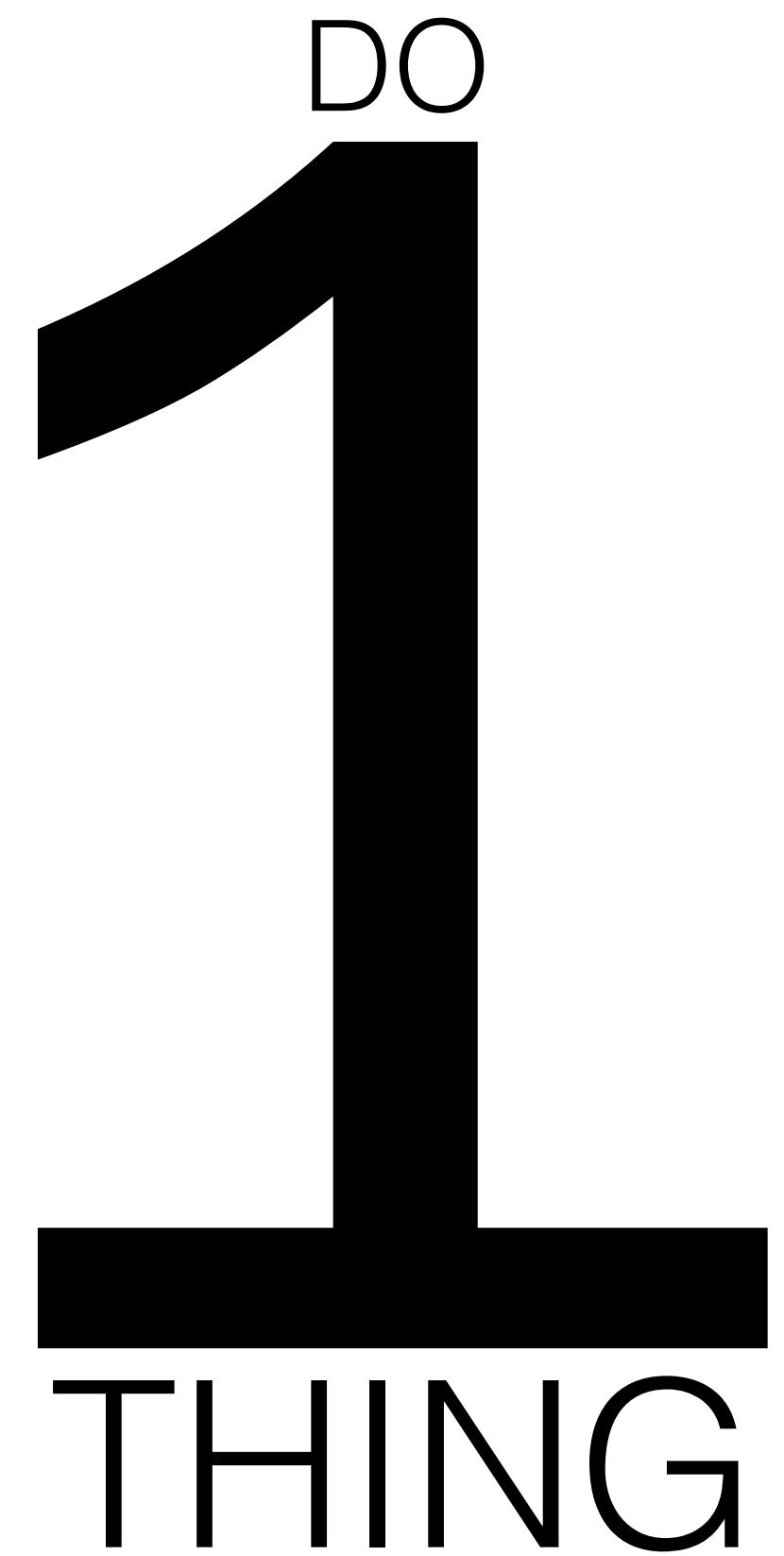
“Any ***organization that designs a system*** (defined more broadly here than just information systems) will inevitably produce a design whose structure is a ***copy of the organization's communication structure.***”

–Melvin Conway

Teams around business capability



Single Responsibility Principle



Explicitly Published interface



Independently replace and upgrade

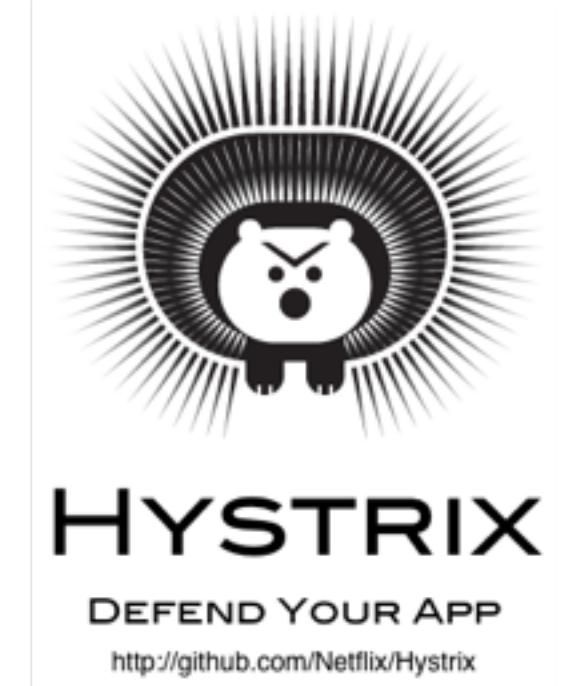




**With great power,
comes
great responsibility**

“you build it, you run it!”

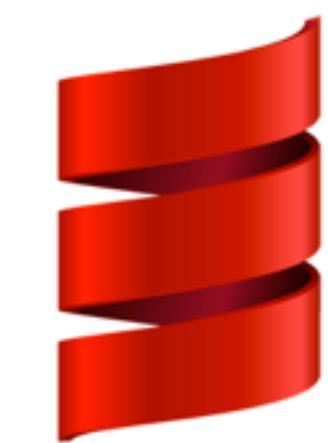
Designed for failure



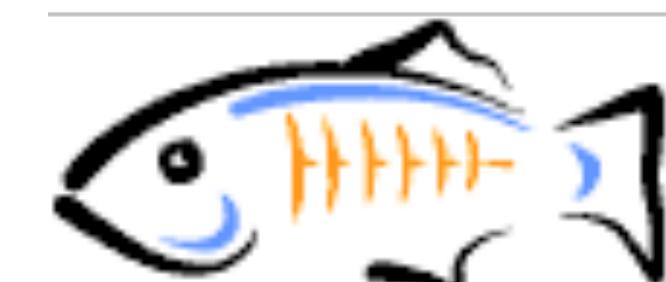
Fault tolerance is a requirement, not a feature



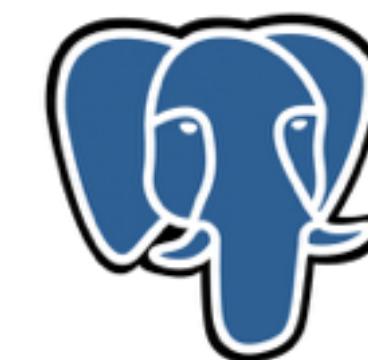
Characteristics



Scala



Couchbase



PostgreSQL



redis



100% automated

Sign Off | Home | Locations | Contact Us | Open Account

WELLS FARGO 

Accounts Bill Pay Transfers Brokerage Account Services Messages & Alerts

Bill Pay Overview Payments Payees eBills Reports Notices User Profile

Bill Pay Overview [? Help](#)

Make Payment

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

Payee Add a Payee	Pending Payment	Last Paid	Amount	Send On
AMERICAN EXPRESS	\$2,053.50 06/30/2004	\$1,349.93 05/24/2004	\$ <input type="text"/> mm/dd/yyyy 3 Business Days	
BANK OF AMERICA  Receiving eBills View eBill	\$198.80 06/30/2004	\$92.17 05/25/2004	\$ <input type="text"/> mm/dd/yyyy 3 Business Days	
BANK ONE / FIRST	\$55.00 06/25/2004	\$55.00 05/22/2004†	\$ <input type="text"/> mm/dd/yyyy 3 Business Days	
CHARLES SCHWAB			\$ <input type="text"/> mm/dd/yyyy 5 Business Days	
CITIBANK VISA  Pending activation	\$63.50 06/30/2004*	\$198.80 05/25/2004*	\$ <input type="text"/> mm/dd/yyyy 5 Business Days	
DIRECT TV  Activate eBills		\$63.50 05/25/2004	\$ <input type="text"/> mm/dd/yyyy 3 Business Days	
SBC-PACIFIC BELL		\$45.80 05/25/2004	\$ <input type="text"/> mm/dd/yyyy 5 Business Days	
SPRINT PCS  Activate eBills	\$49.78 06/30/2004	\$63.50 06/25/2004	\$ <input type="text"/> mm/dd/yyyy 5 Business Days	
SFPUC-WATER DE			\$ <input type="text"/> mm/dd/yyyy 3 Business Days	
WF HOME MORTGAGE		\$1,349.93 05/25/2004	\$ <input type="text"/> mm/dd/yyyy 5 Business Days	

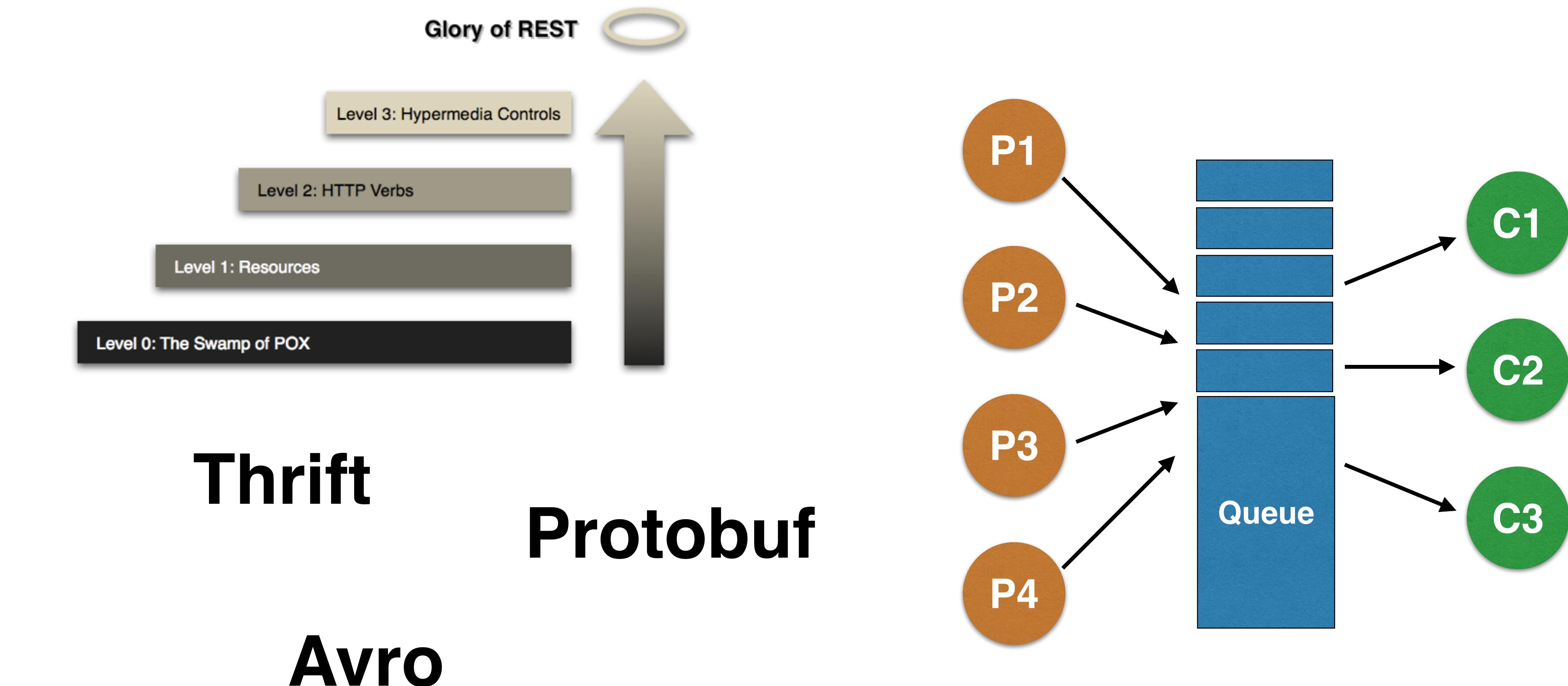
[Make Payment](#)

[Home](#) | [Locations](#) | [Contact Us](#) | [Open Account](#) | [Sign Off](#)
© 2001-2005 Wells Fargo. All rights reserved.

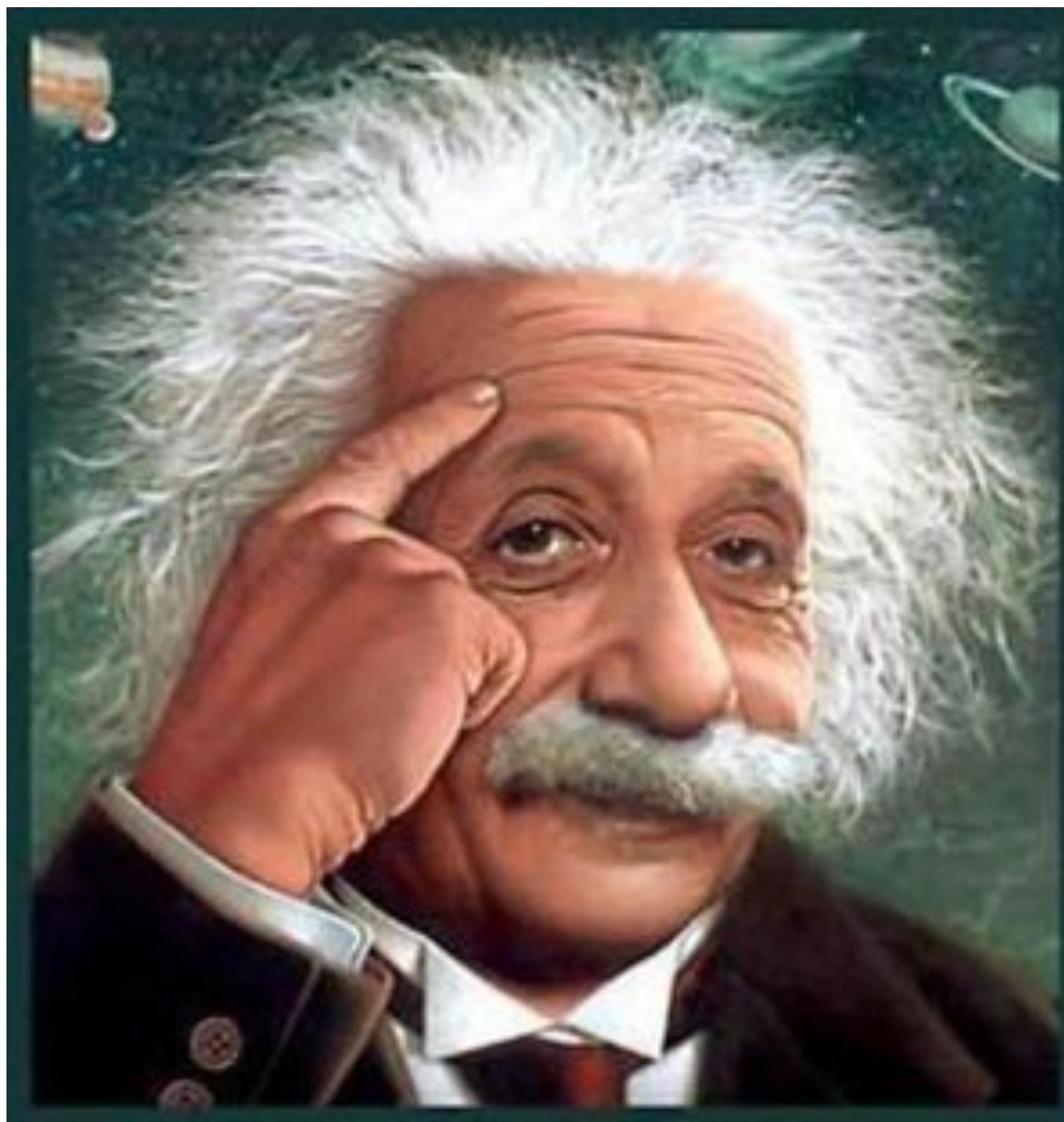
Sync or Async Messaging



Sync vs Async



“Smart endpoints, Dumb pipes”



SOA

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

SOA 2.0?



Arun Gupta
@arungupta

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



RETWEETS

72

FAVORITES

63



5:07 PM - 27 May 2015

- Conway's Law
- Service Discovery
- Immutable VM

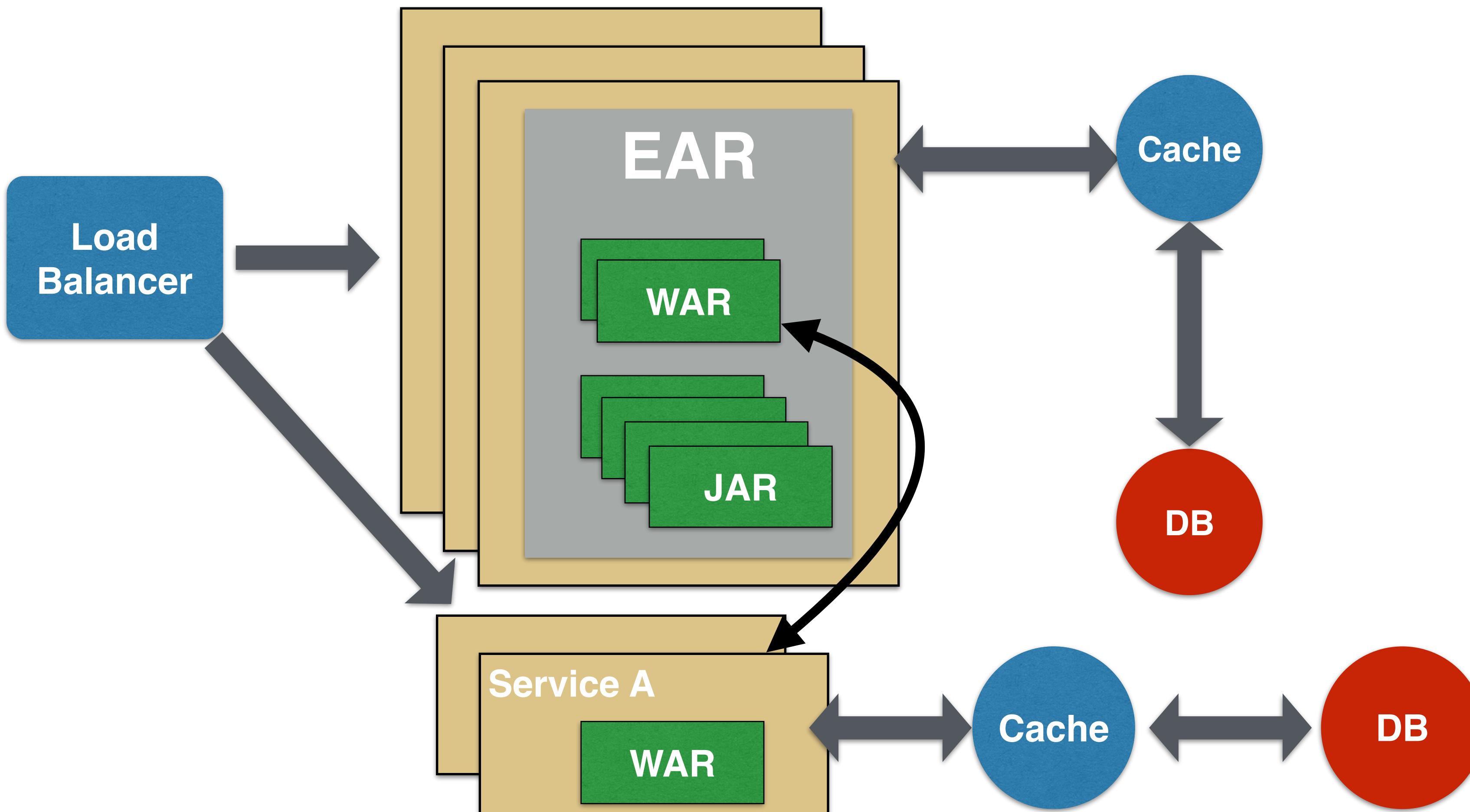
**AM I THE ONLY
ONE**

**WHO DID NOT
UNDERSTAND**

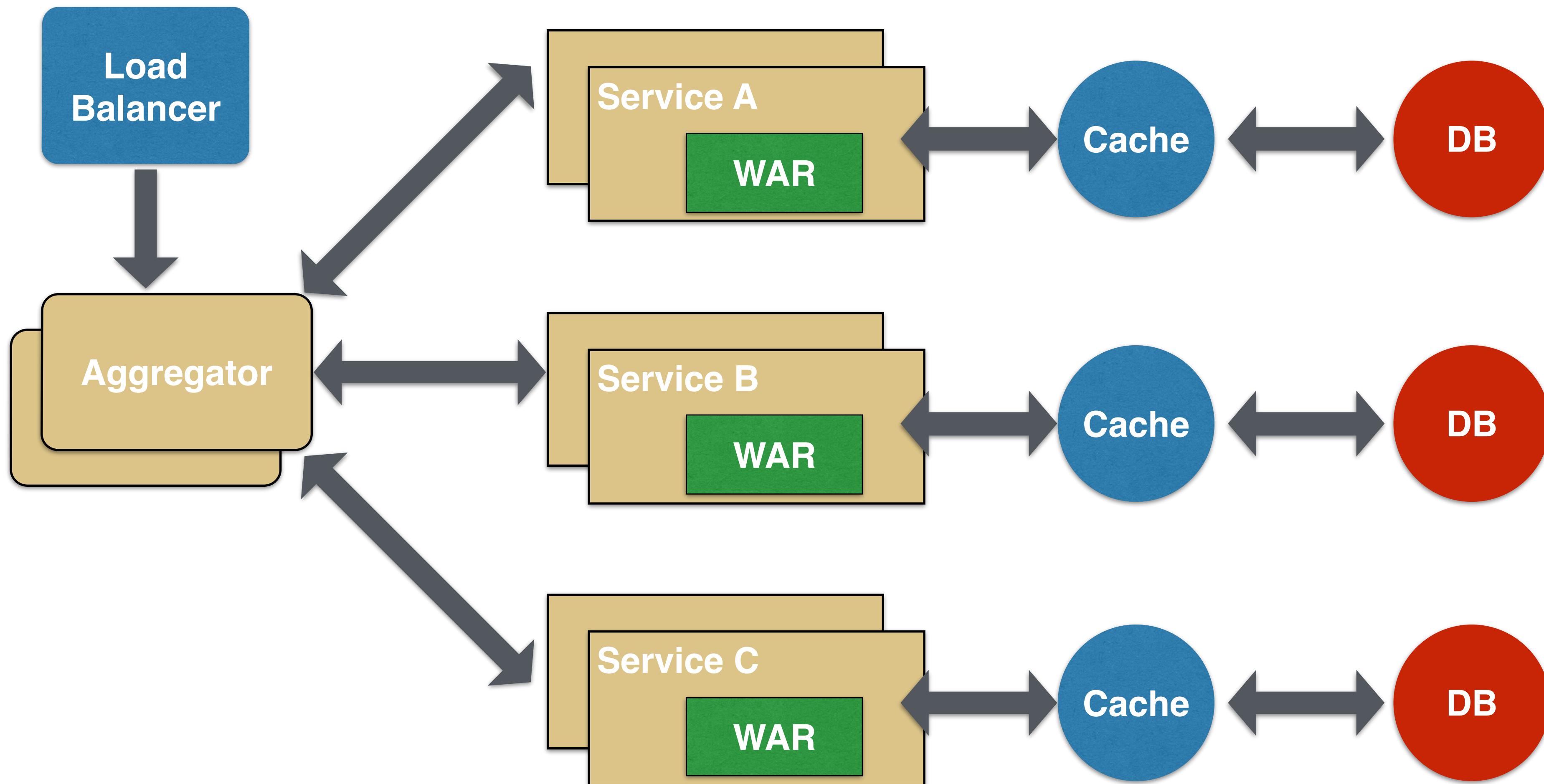
Strategies for decomposing

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

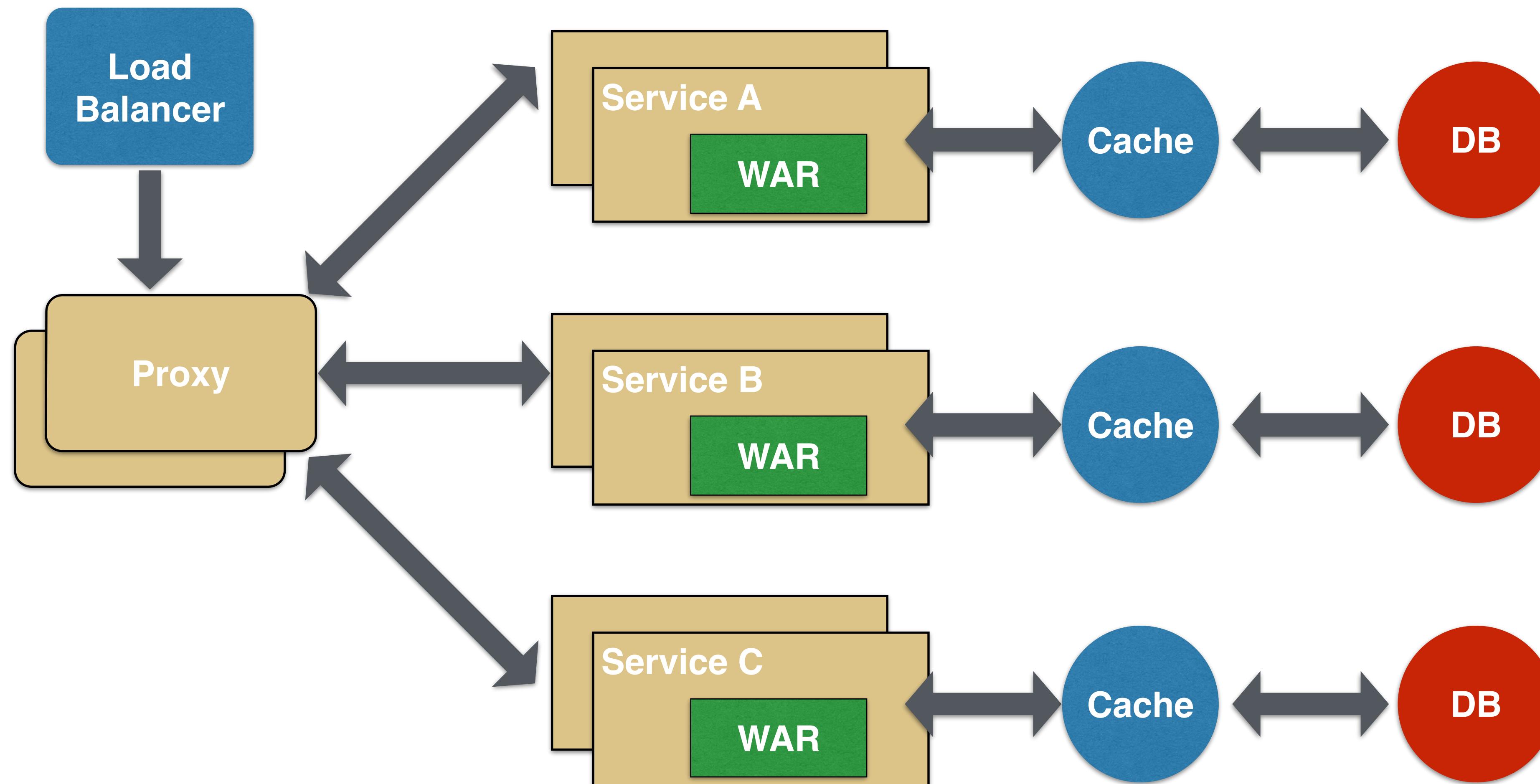
Towards microservices



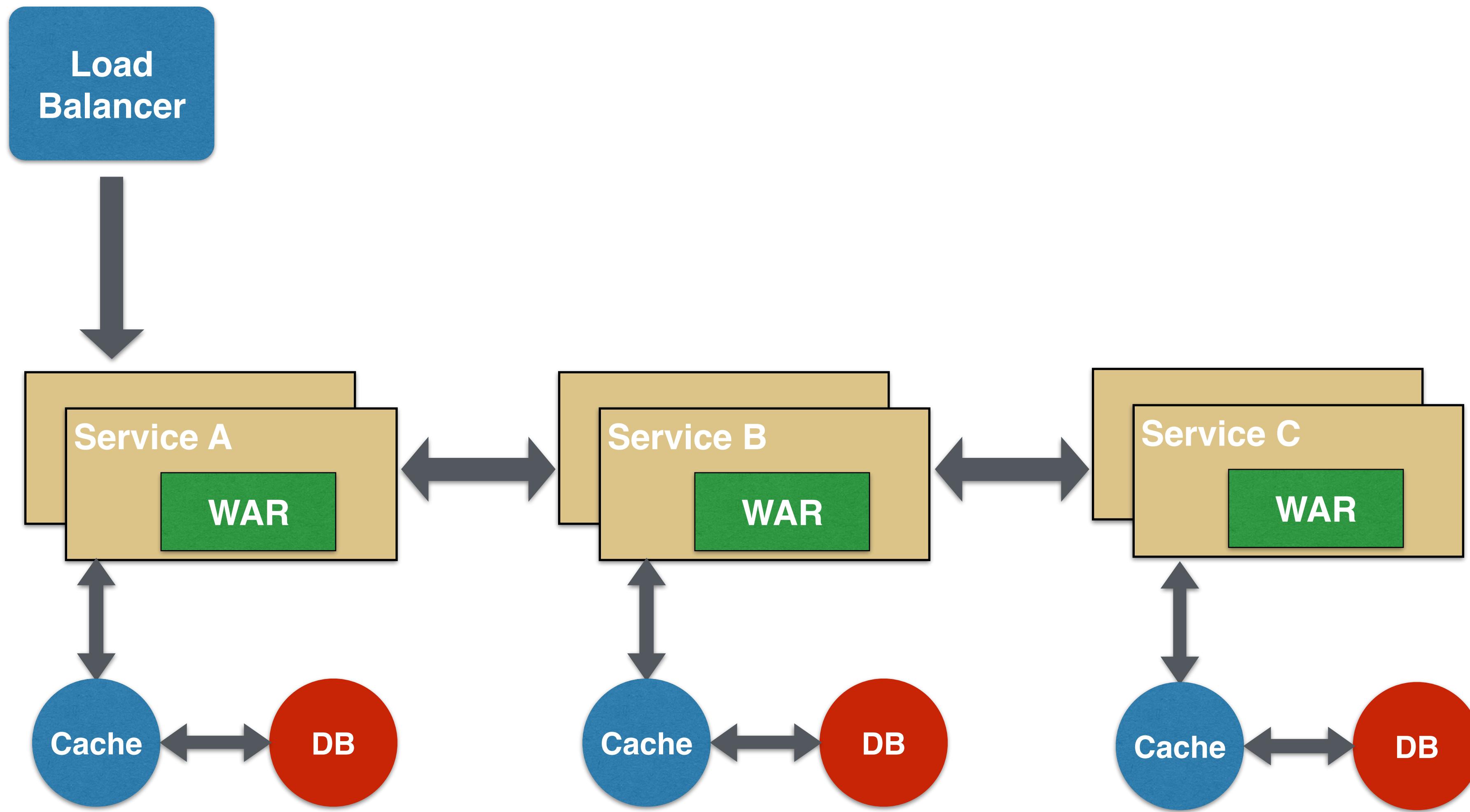
Aggregator Pattern #1



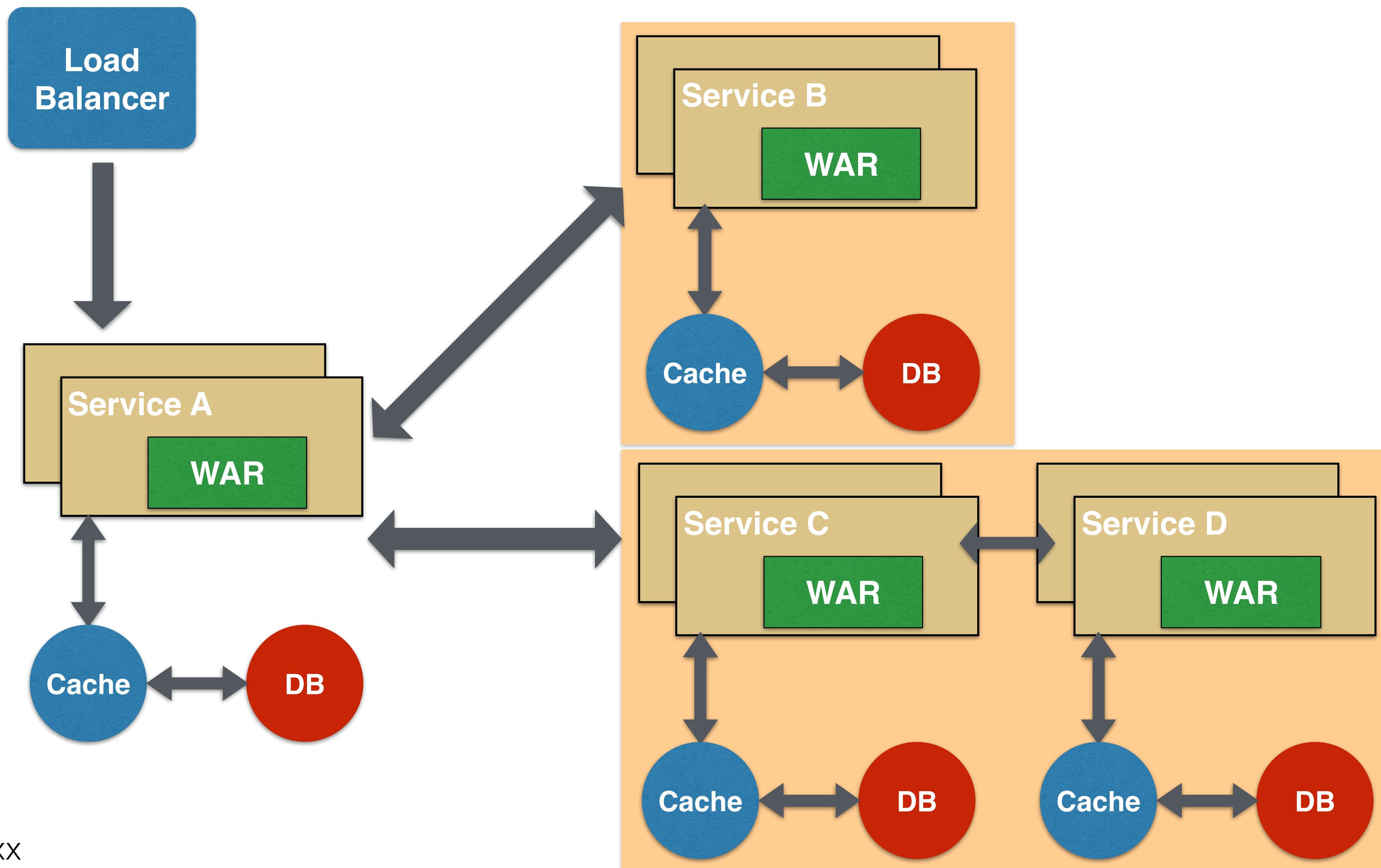
Proxy Pattern #2



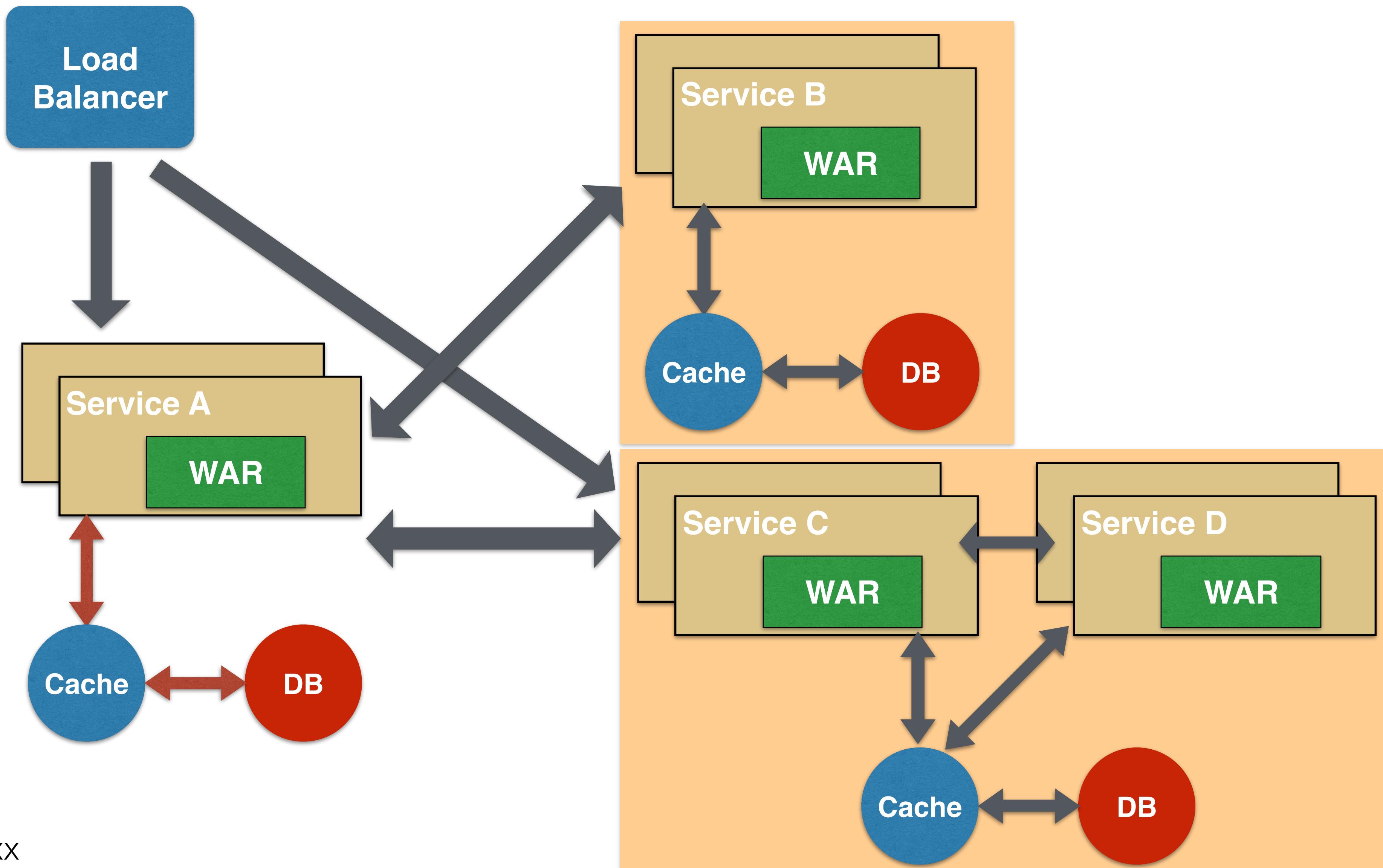
Chained Pattern #3



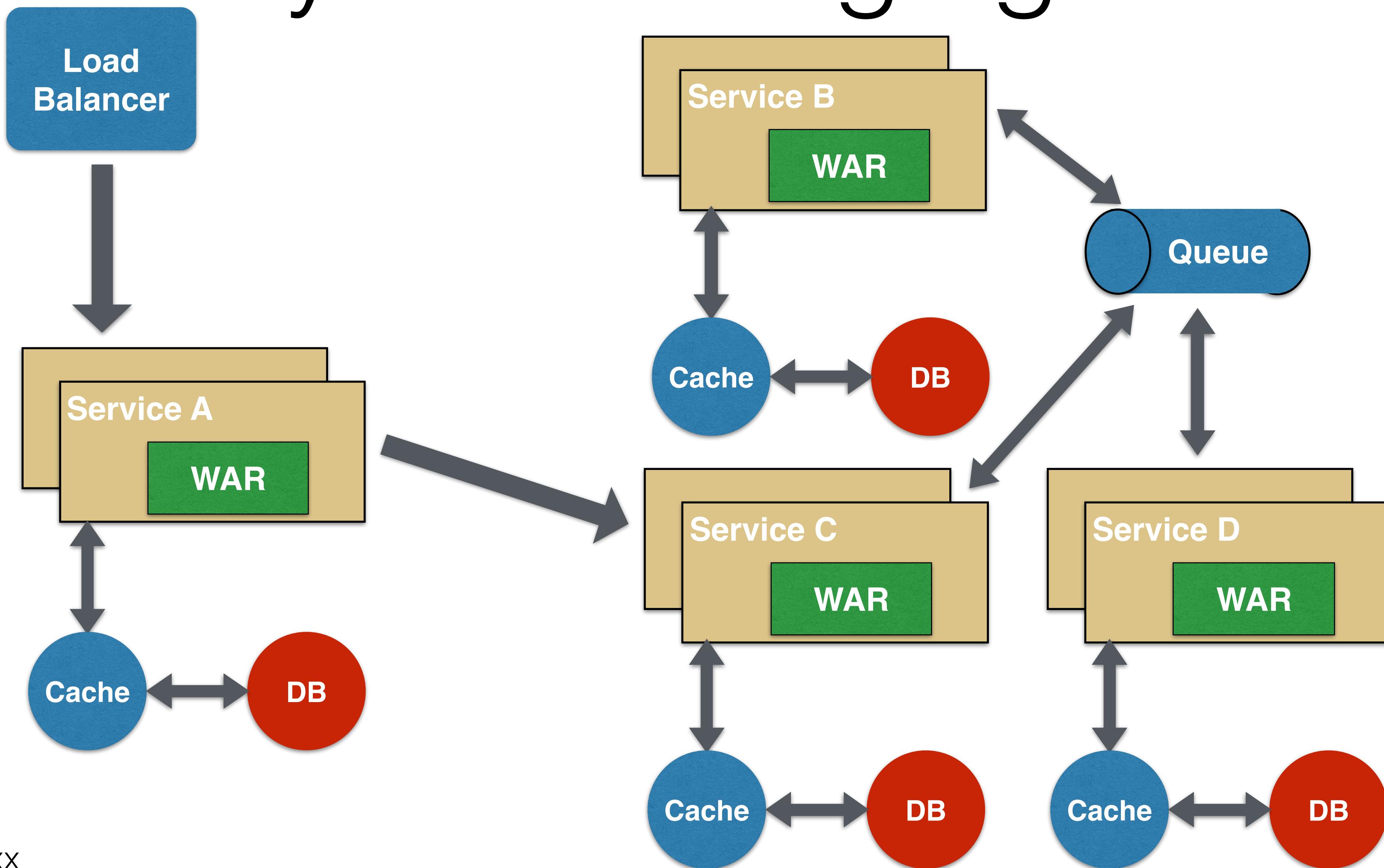
Branch Pattern #4

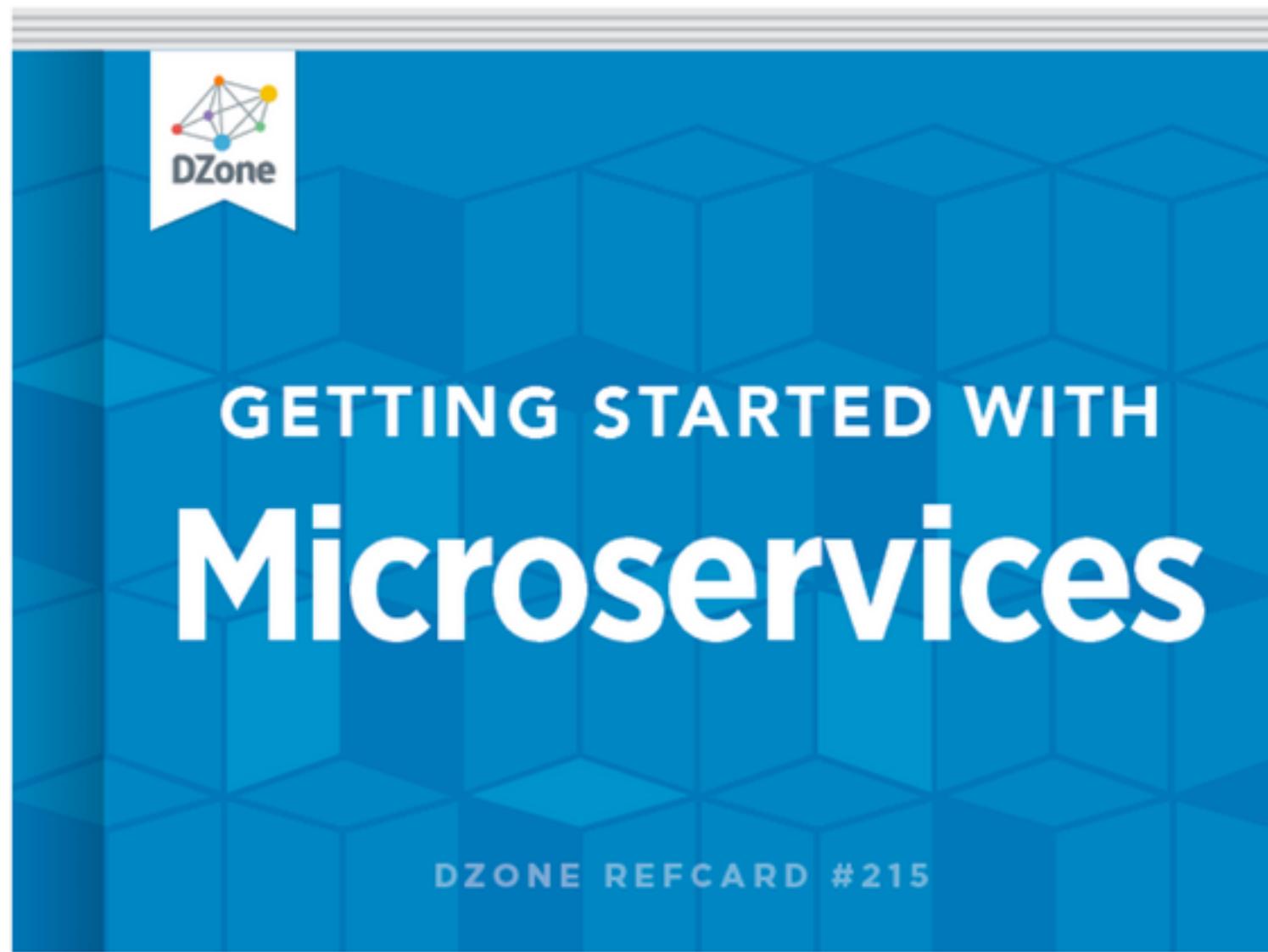


Shared Resources #5



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

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

SAY MICROSERVICE

ONE MORE TIME

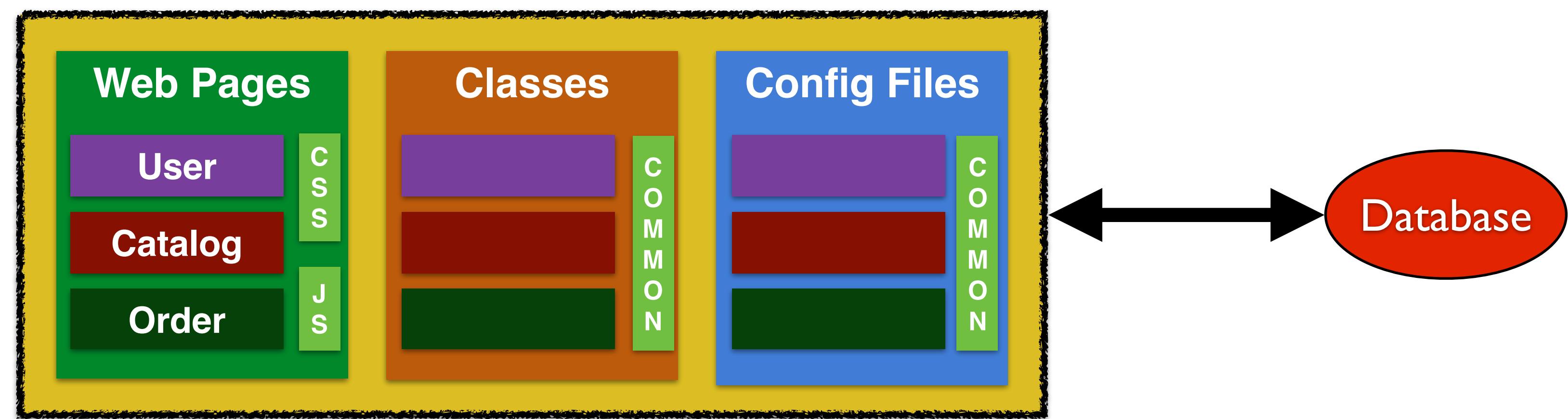
memegenerator.net

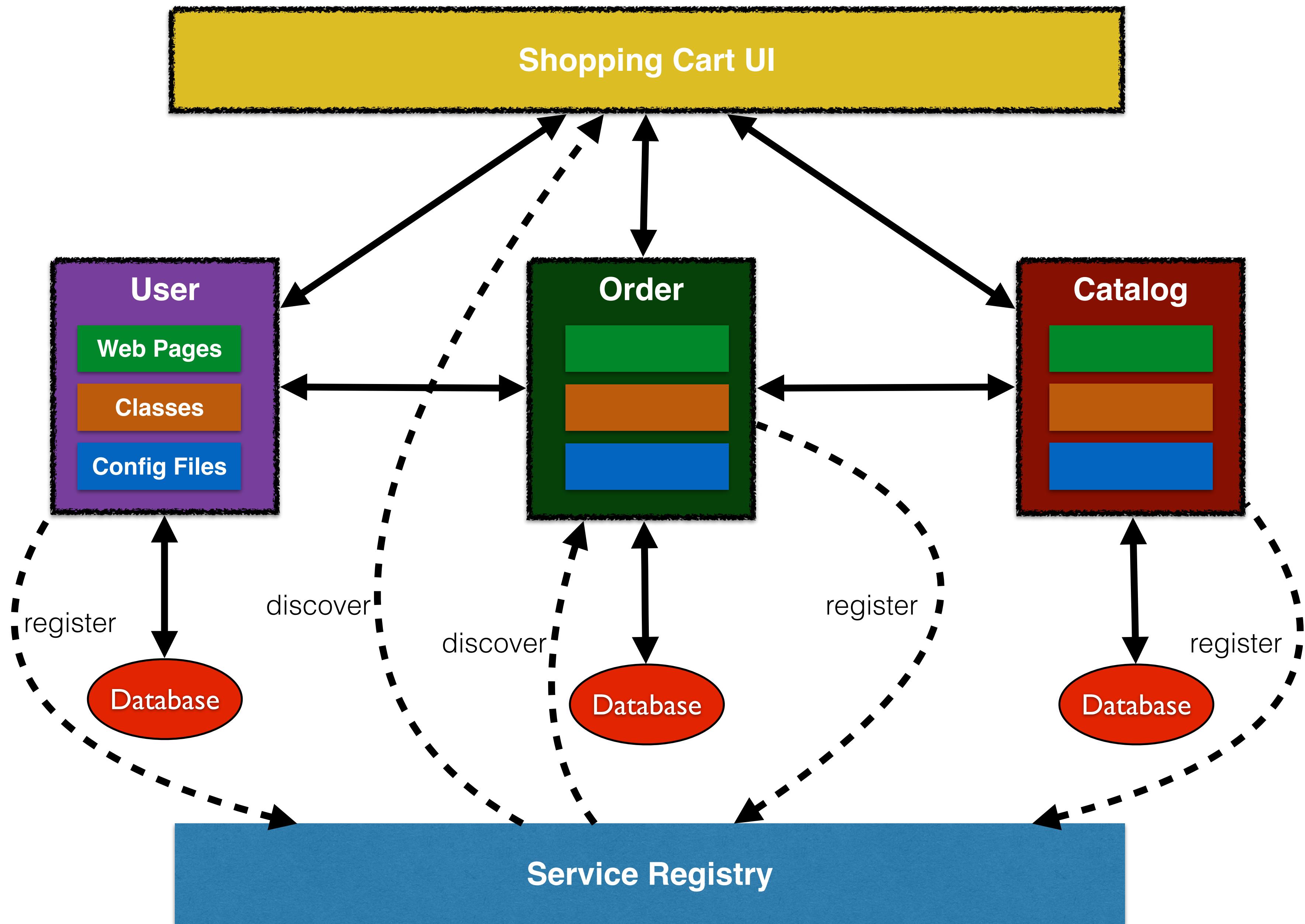
“If you can't build a [well-structured] monolith, what makes you think microservices are the answer?”

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

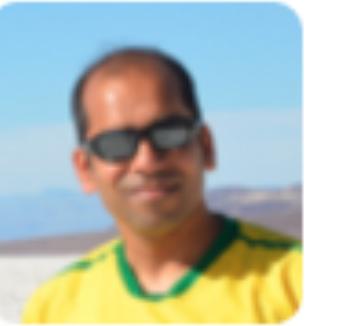
- UI and Full stack
 - Client-side composition (JavaScript?)
 - Server-side HTML generation (JSF?)
 - One service, one UI
- REST Services
- Event sequencing instead of 2PC
- API Management

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, ...

NoOps

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



Arun Gupta
@arungupta

Containers not necessary for #microservices,
#microservices does not mean you must use
containers. Similarities, but can exist w/o
other!

RETWEETS
20

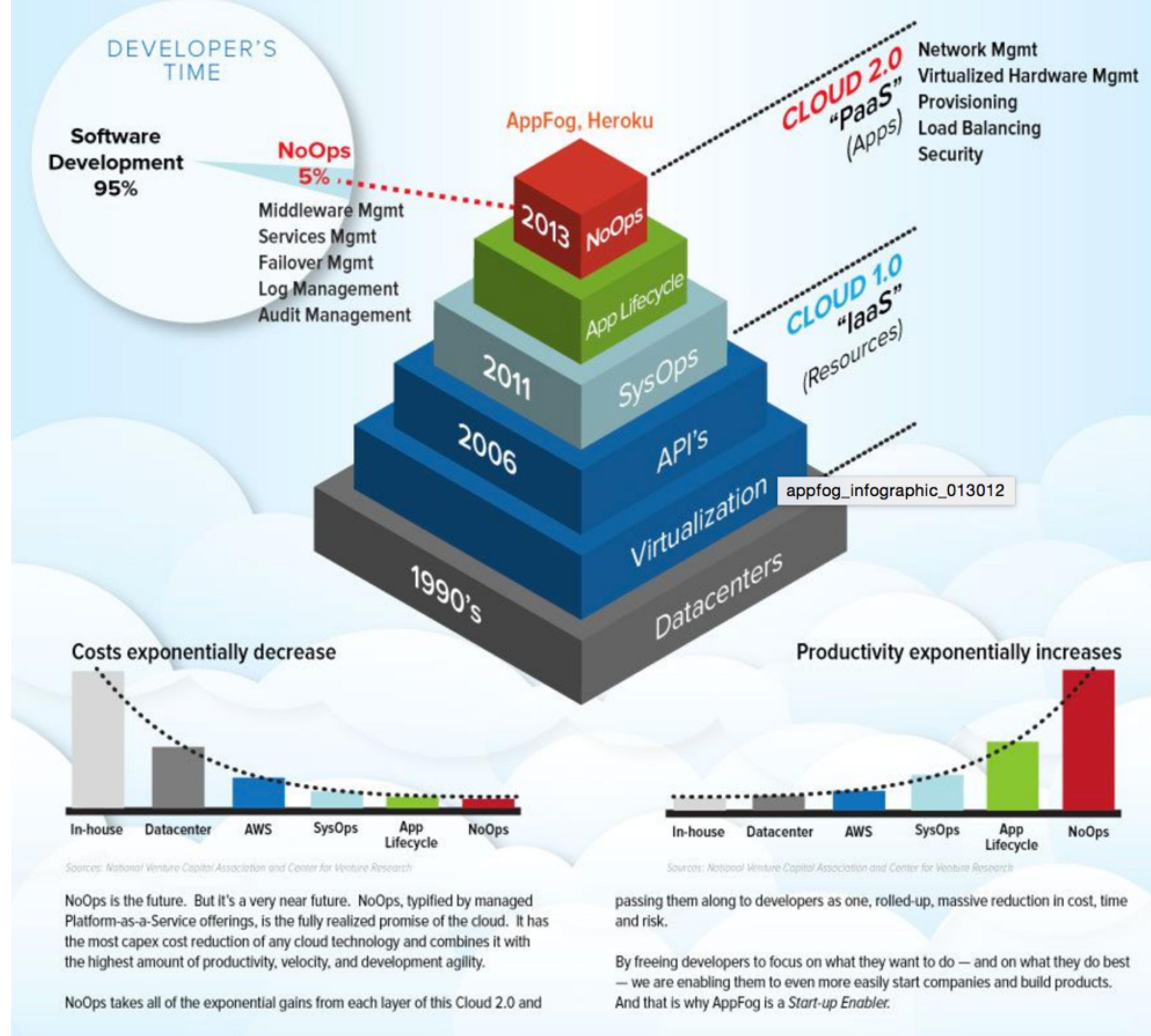
LIKES
11



6:37 AM - 7 Nov 2015

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

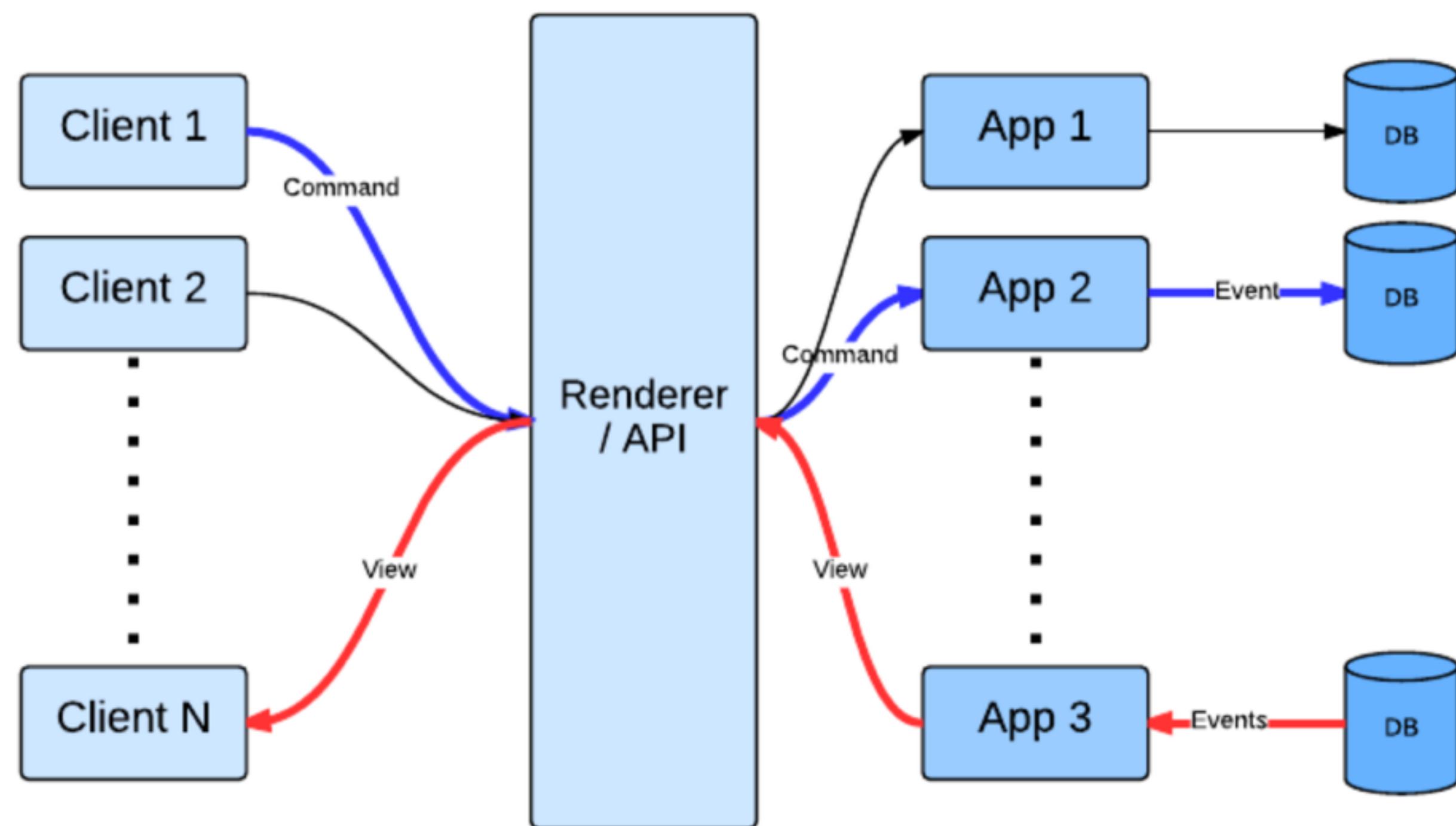
- Private tables-, schema-, or database-per-service
- No sharing tables
 - Only one service writes to a table(s), read-only by others
 - No transactions across databases
 - Logical transactions in application

Event Sourcing

- State of the application is defined by a sequence of events
- Events are stored in a document format
- Events are immutable
 - Delete is implemented as an event

Event Source @ WixStores

- Flexible
- Immutable
- Stateless



Event Sourcing

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": "aaaaaa"}
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"}

10 rows in set (0.00 sec)



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



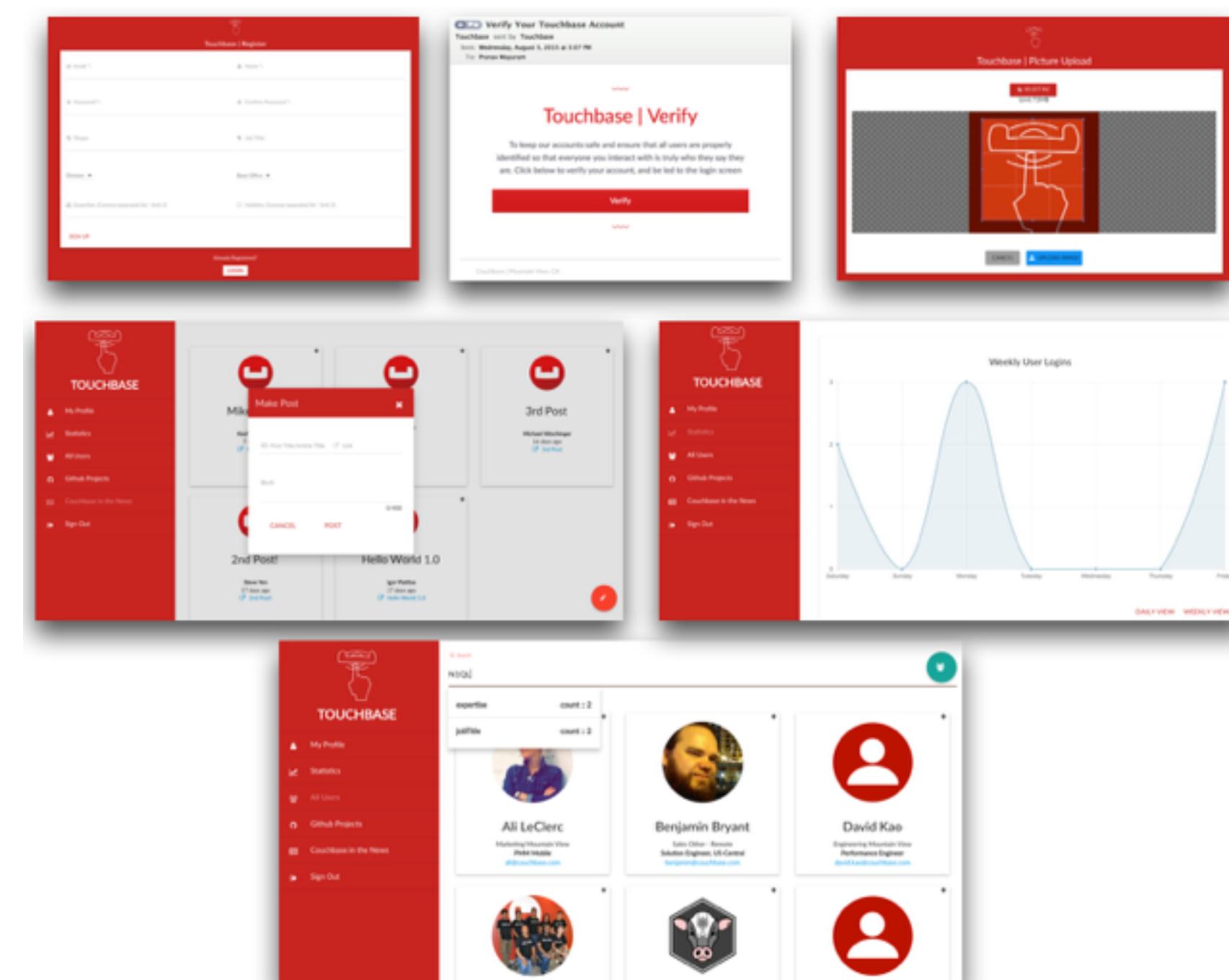
Multiple writes

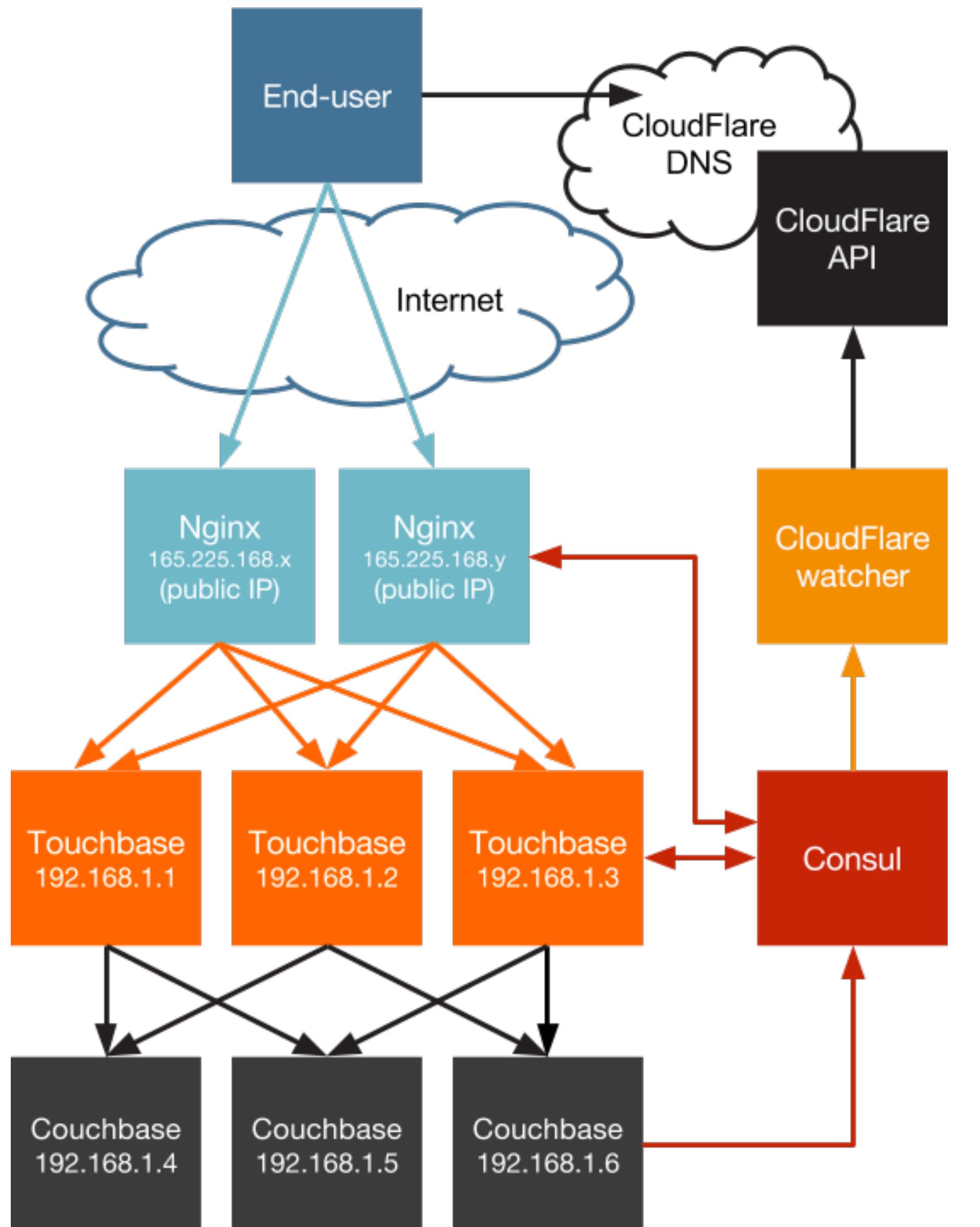
Less read, cached



Microservices stack in seconds

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





NGINX



```

1  # Consul as a service discovery tier
2
3  consul:
4      image: program/consul:latest
5      command: -server -bootstrap -ui-dir /ui
6      restart: always
7      mem_limit: 128m
8      ports:
9          - 53
10         - 8300
11         - 8301
12         - 8302
13         - 8400
14         - 8500
15     dns:
16         - 127.0.0.1
17
18 # Manually bootstrap the first instance, then...
19 # Scale this tier and each additional container/instance will
20 # self-configure as a member of the cluster
21 couchbase:
22     image: misterbisson/triton-couchbase:enterprise-4.0.0-1
23     restart: always
24     links:
25         - consul
26     mem_limit: 4g
27     ports:
28         - 8091
29         - 8092
30         - 8093
31         - 11207
32         - 11210
33         - 11211
34         - 18091
35         - 18092
36     environment:
37         - CONSUL_HOST=http://consul:8500
38         - COUCHBASE_SERVICE_NAME=couchbase
39         - COUCHBASE_USER=Administrator
40         - COUCHBASE_PASS=password
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 cloudflare:
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

```





- Lightweight open source framework for Java EE microservices
- Uses standard Java EE APIs
- Produces standalone JARs that run anywhere
- Completely modular and easily extensible

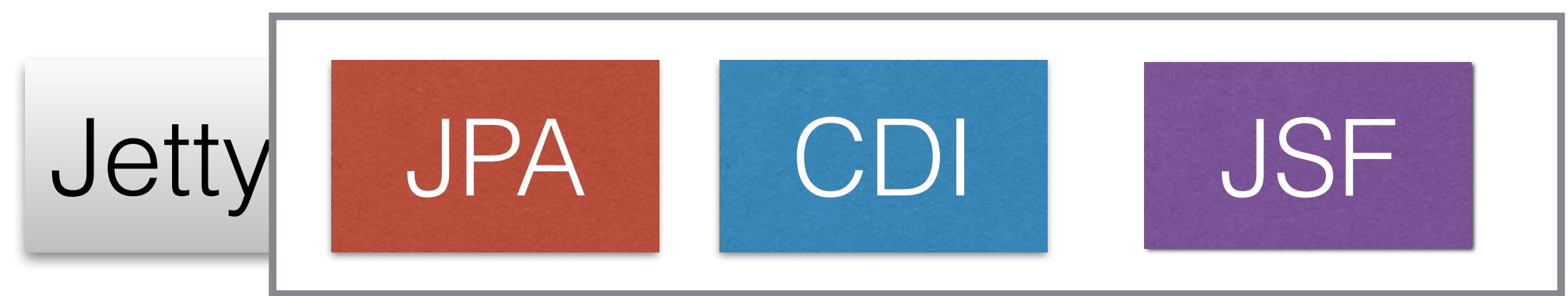
Great Fit for Microservices

- Dependency-driven - simple to use
- Pick and choose Java EE components that you want
 - Even their implementations
- JAR includes all the dependencies
- Ideal for running in PaaS and Docker

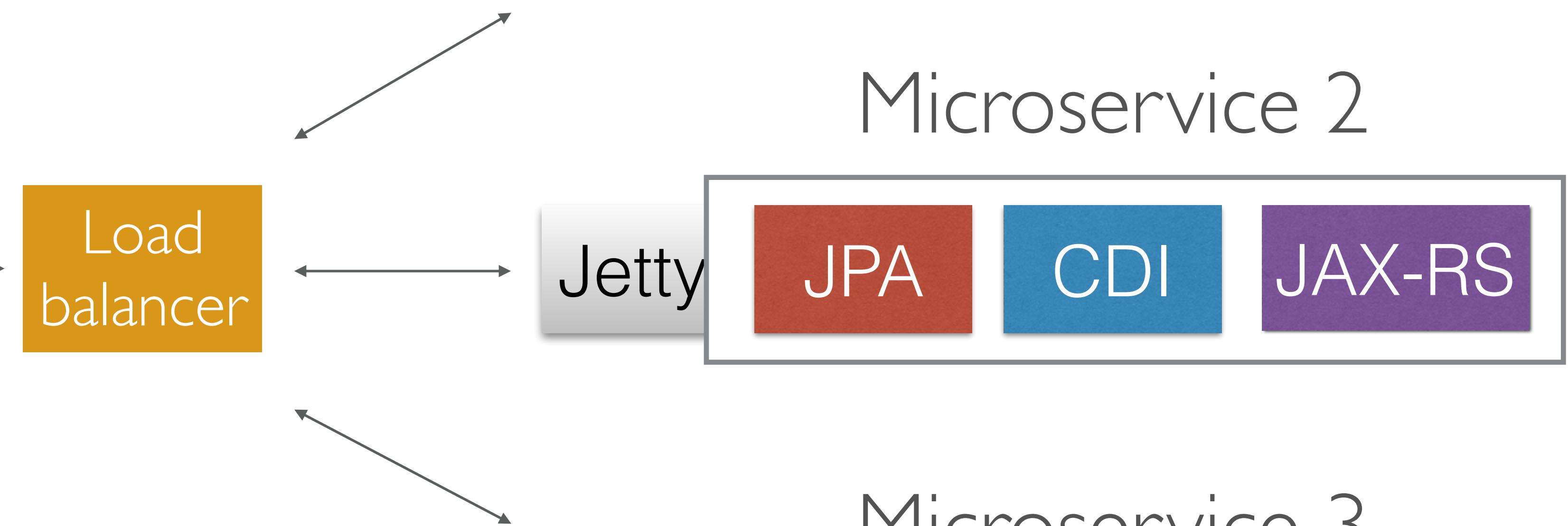
Components

- Containers
 - Jetty (default)
 - Coming: Tomcat, Undertow, Grizzly
- Java EE components
 - Current: Servlet, JSP, EL, CDI, JAX-RS, Bean Validation, JSON, JSF
 - Coming: JMS, EJB, JAX-WS

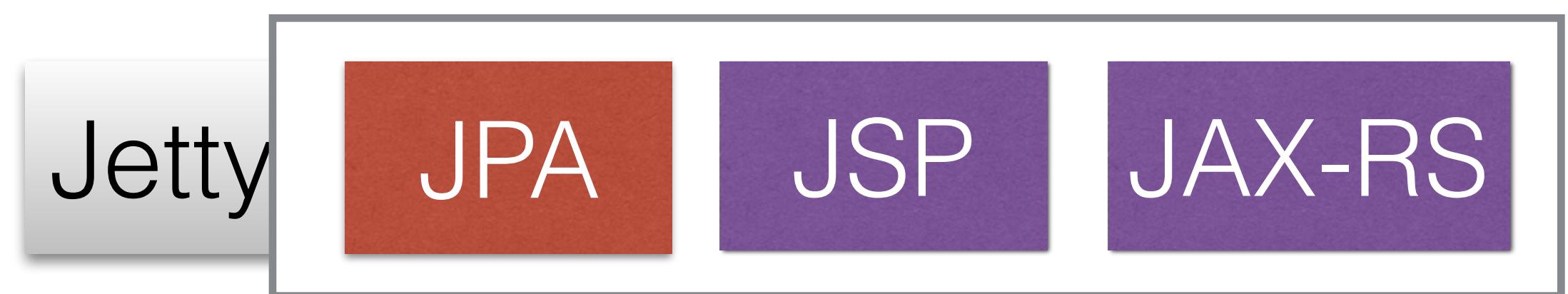
Microservice I



Microservice 2



Microservice 3



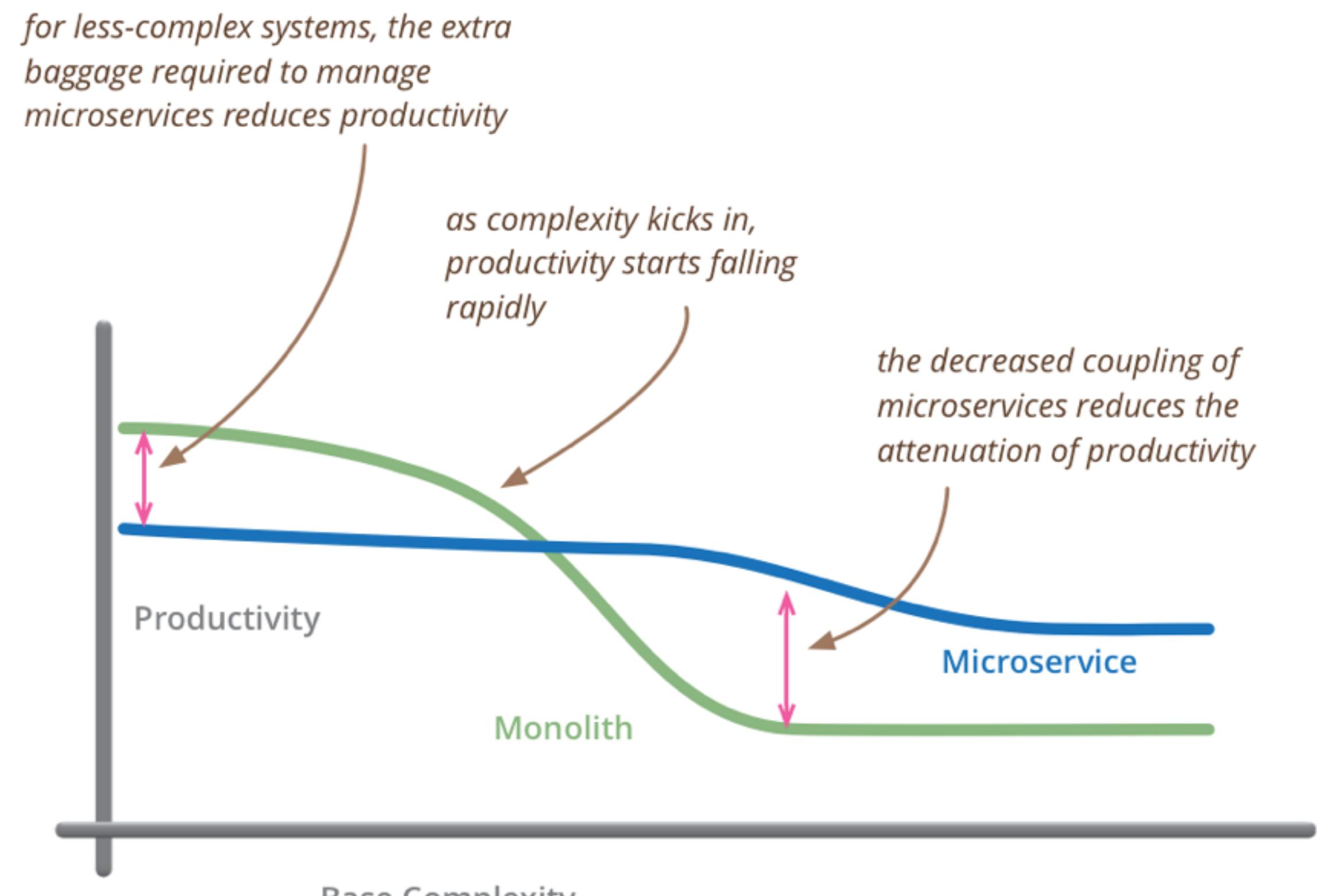
How to scale?

- Only requires Java SE
- Multiple instances and load balance
- Scale using PaaS and all Docker-like environments
- Each microservice can scale horizontally or vertically

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



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

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

References

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