

Microservices Design Patterns for Java Applications

Arun Gupta

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



O'REILLY®

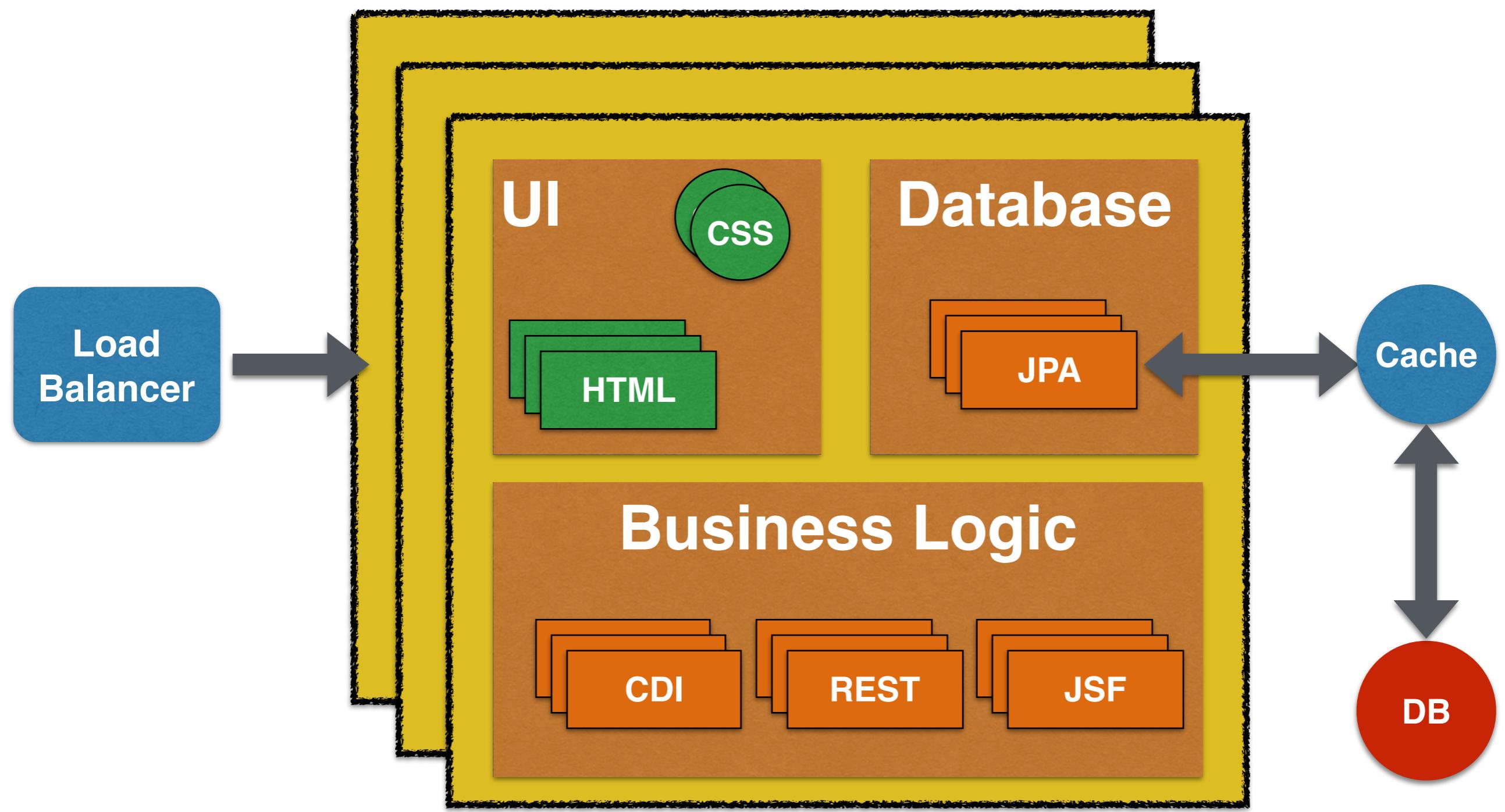
A detailed black and white illustration of a spiny lizard, specifically a horned lizard, resting on a purple rectangular background. The lizard has a textured, scaly body with several prominent spines along its back and tail.

Minecraft Modding with Forge

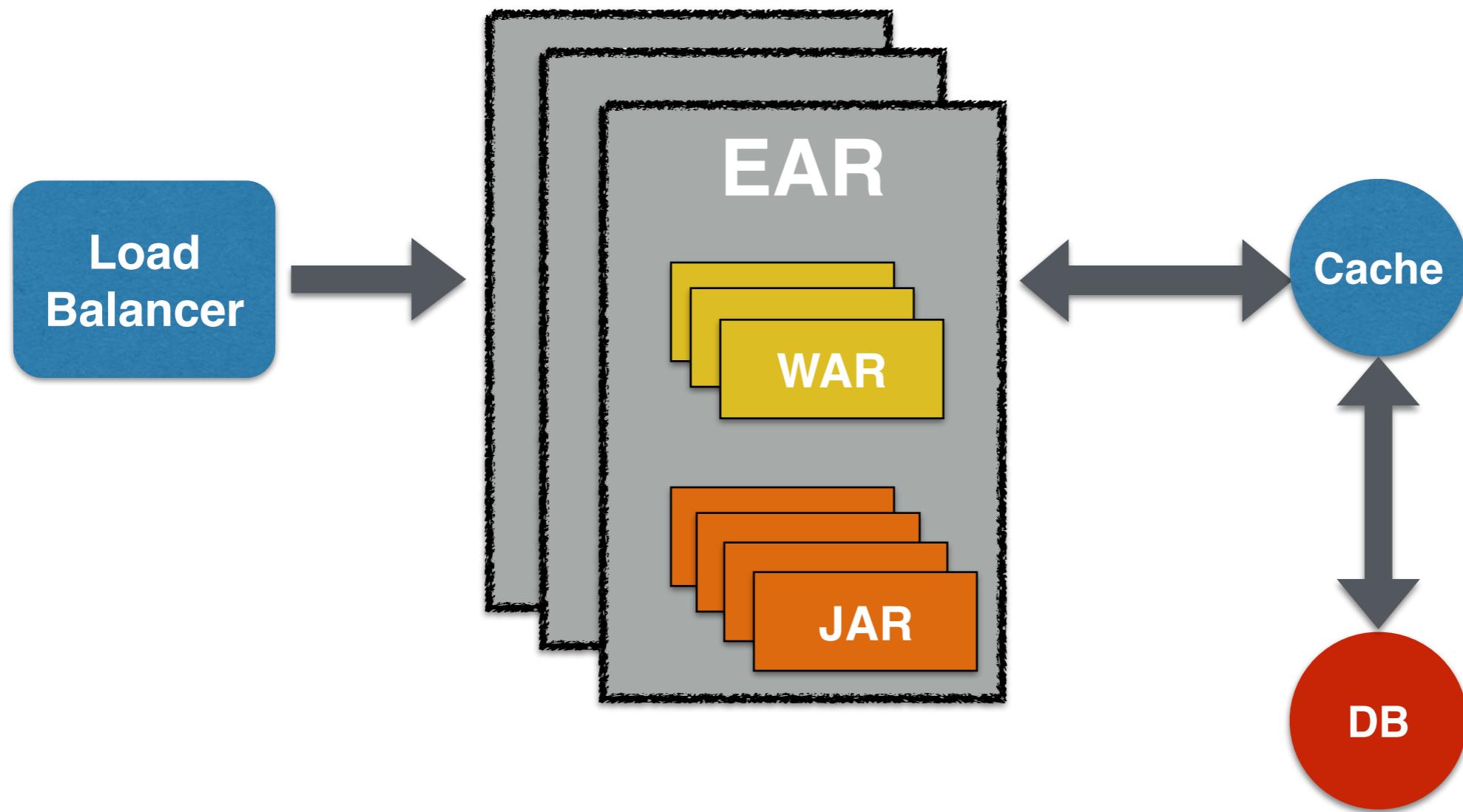
A FAMILY-FRIENDLY GUIDE TO BUILDING FUN MODS IN JAVA

Arun Gupta & Aditya Gupta

Monolith Application



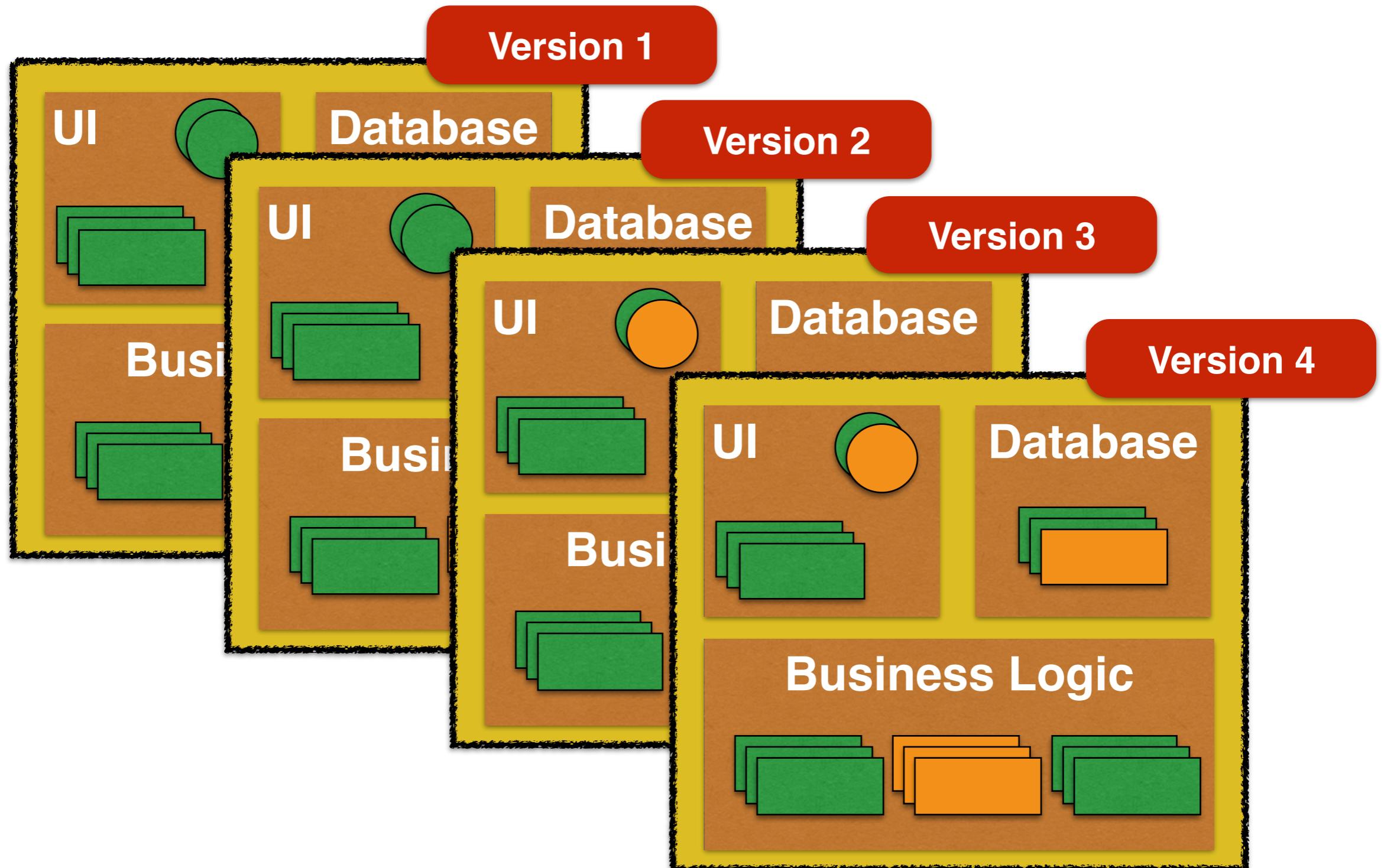
Monolith Application



Advantages of Monolith Application

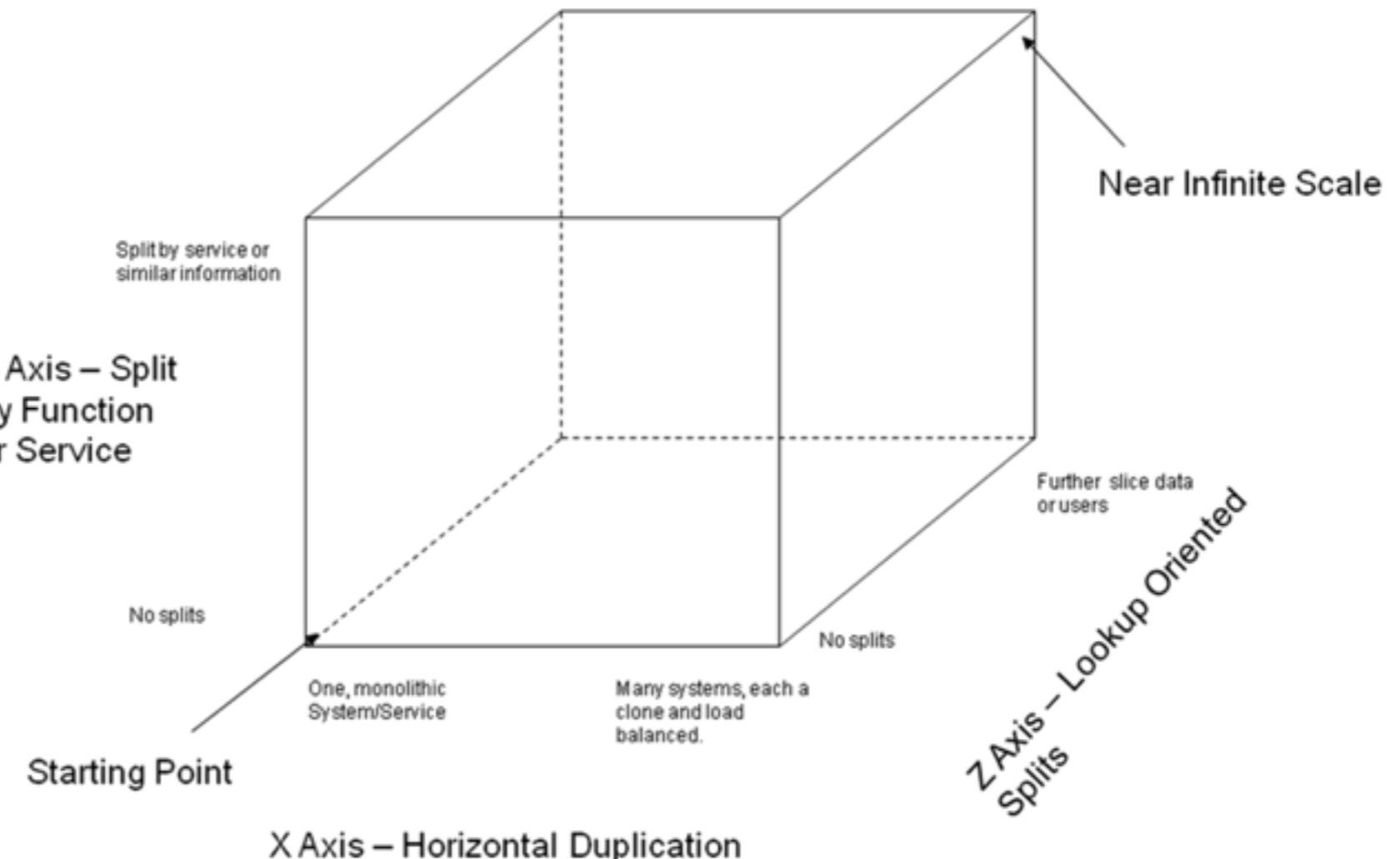
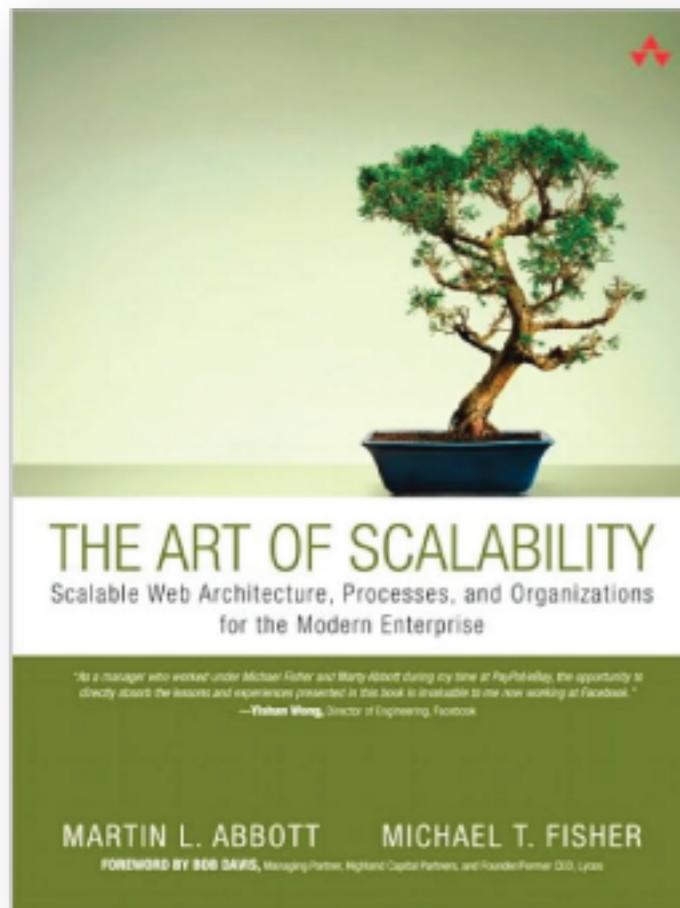
- Typically packaged in a single .ear
- Easy to test (all required services are up)
- Simple to develop

Monolith Application



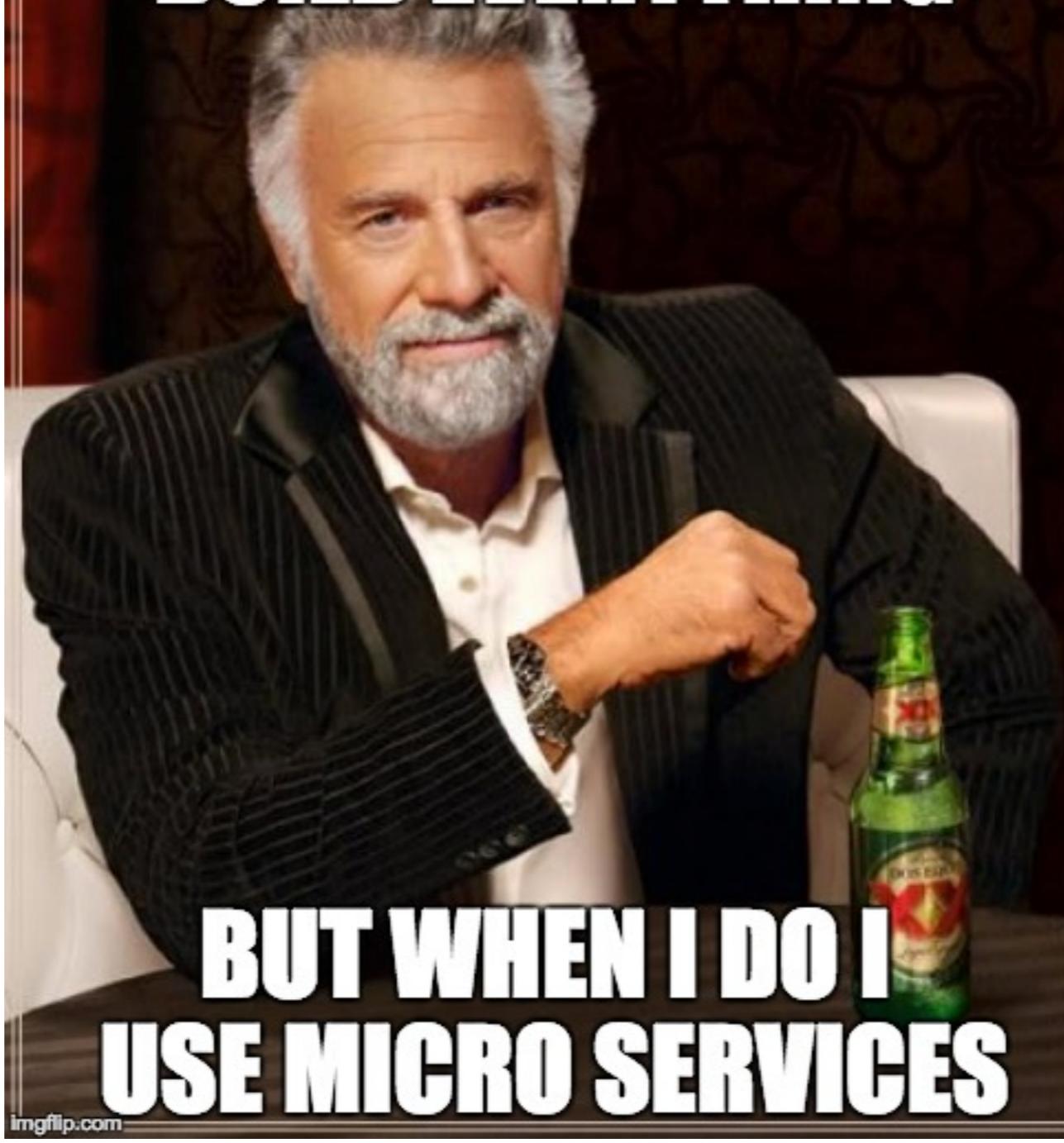
Disadvantages of Monolith Application

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

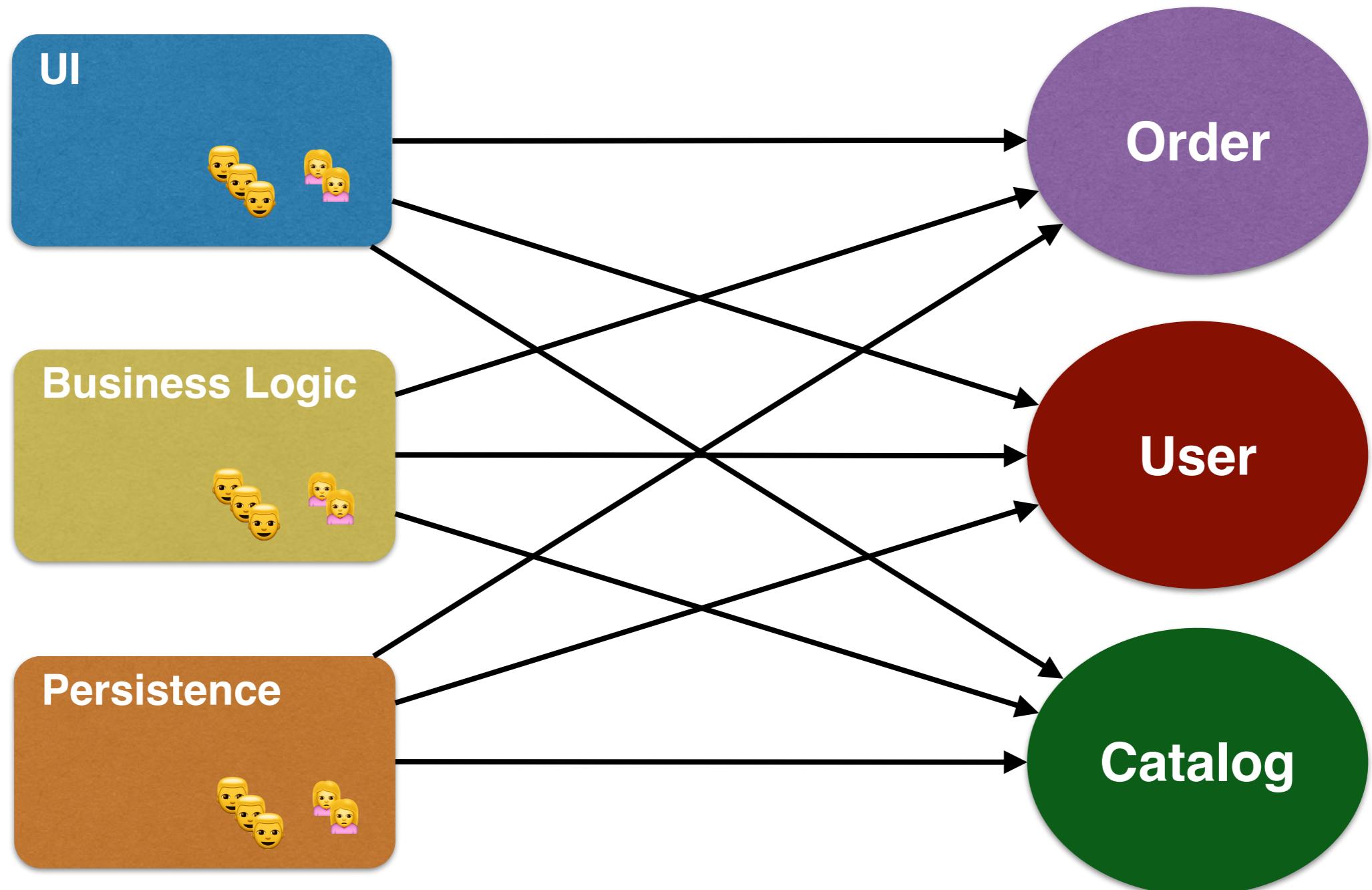


An ***architectural approach***, that emphasizes the ***decomposition of applications*** into ***single-purpose, loosely coupled*** services managed by ***cross-functional teams***, for delivering and maintaining ***complex software systems*** with the velocity and quality required by today's ***digital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

I DONT ALWAYS
BUILD EVERYTHING



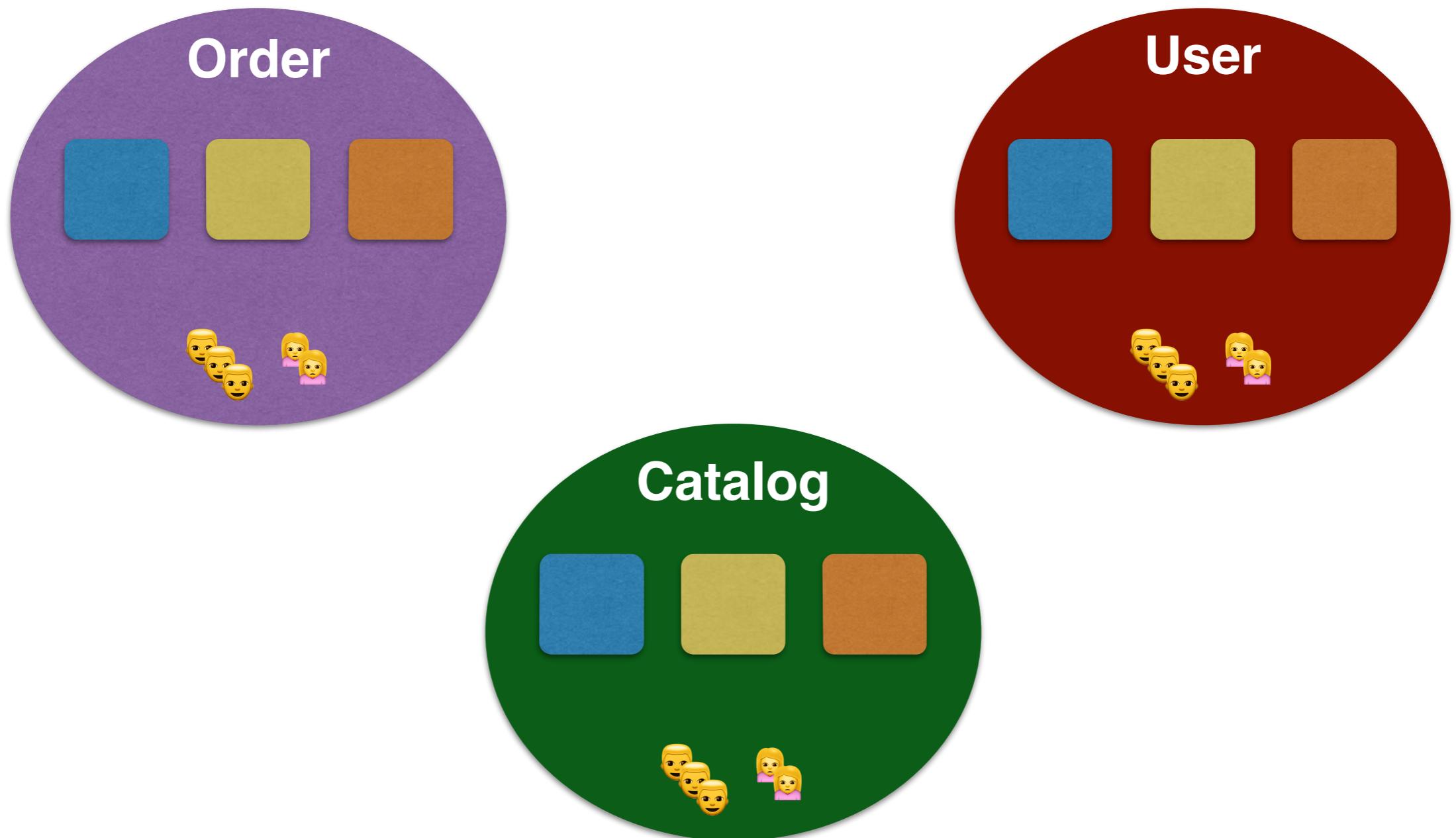
imgflip.com



*“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



Couchbase

100% automated

WELLS FARGO



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

Accounts Bill Pay Transfers Brokerage Account Services Messages & Alerts

Bill Pay Overview Payments Payees eBills Reports Notices User Profile

Bill Pay Overview

Help Unviewed Notices (2)
Unpaid eBills (3)
Pending Payments (6)

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	\$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

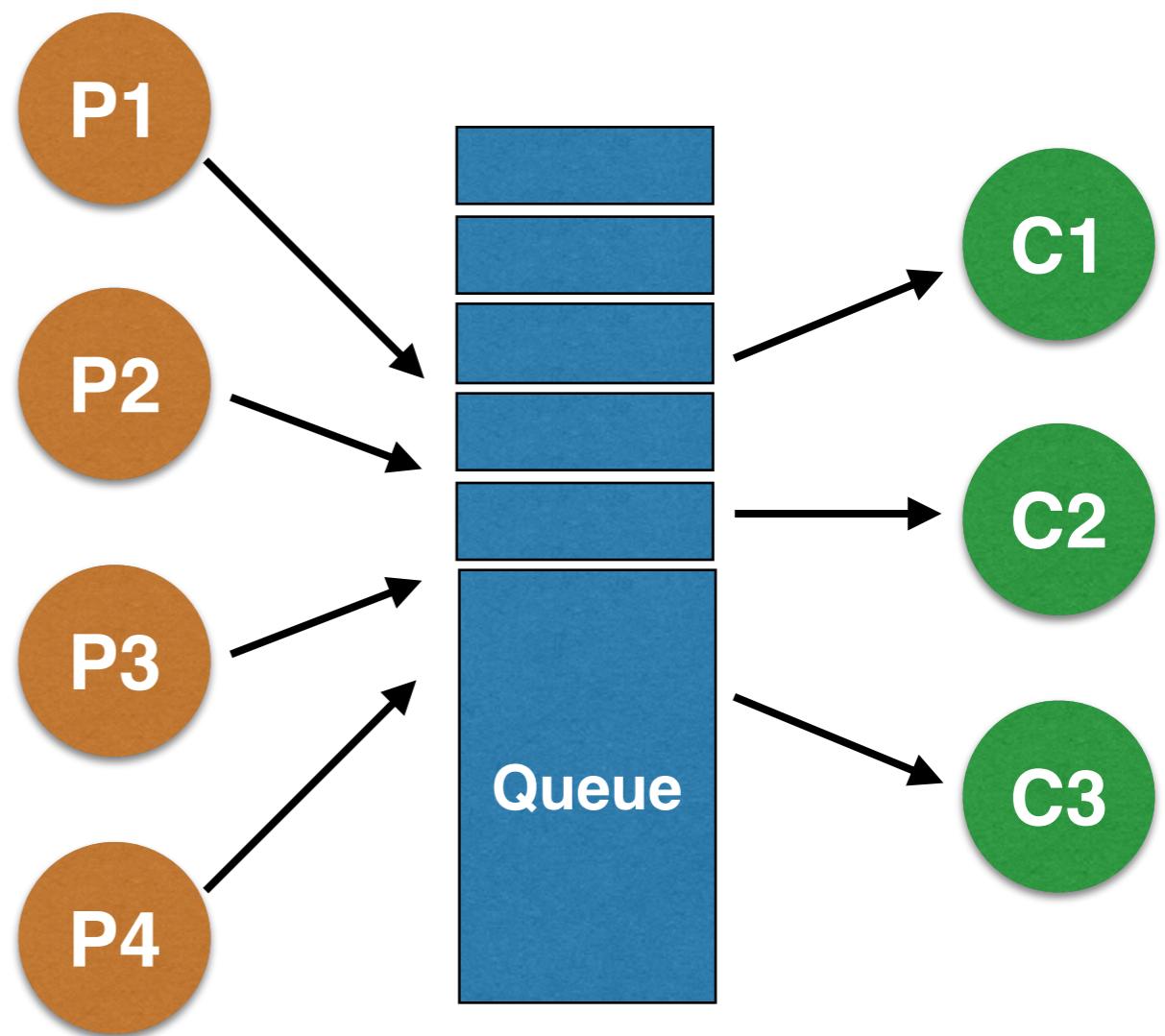
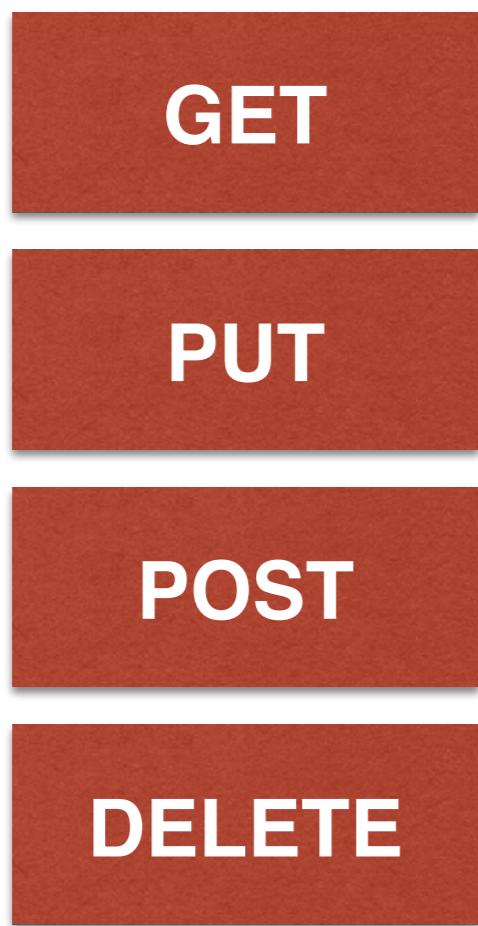
Make Payment

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

Sync or Async Messaging



REST vs Pub/Sub



“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 FAVORITES
72 **63**



5:07 PM - 27 May 2015

- Conway's Law
- Service Discovery
- Immutable VM

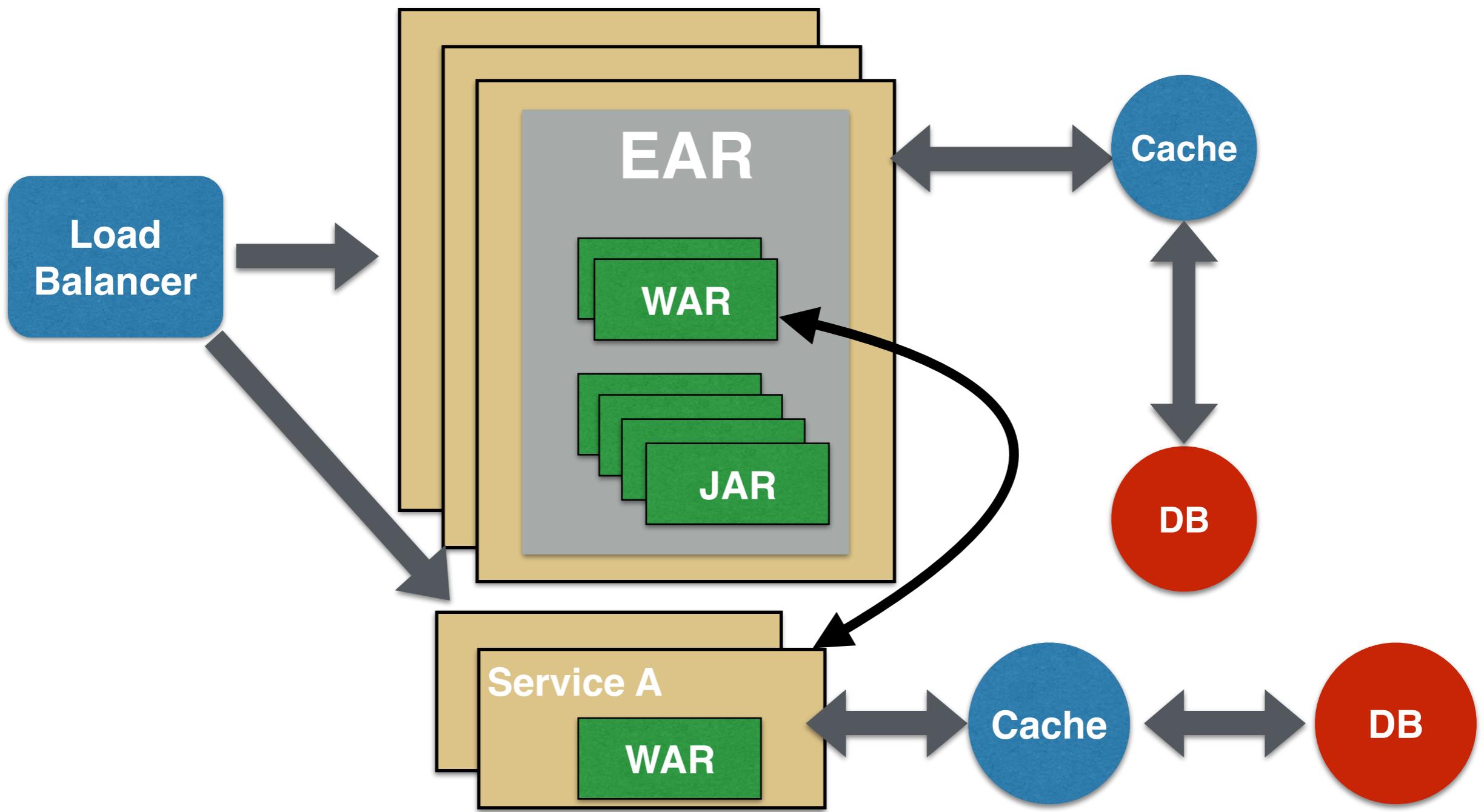
Strategies for decomposing



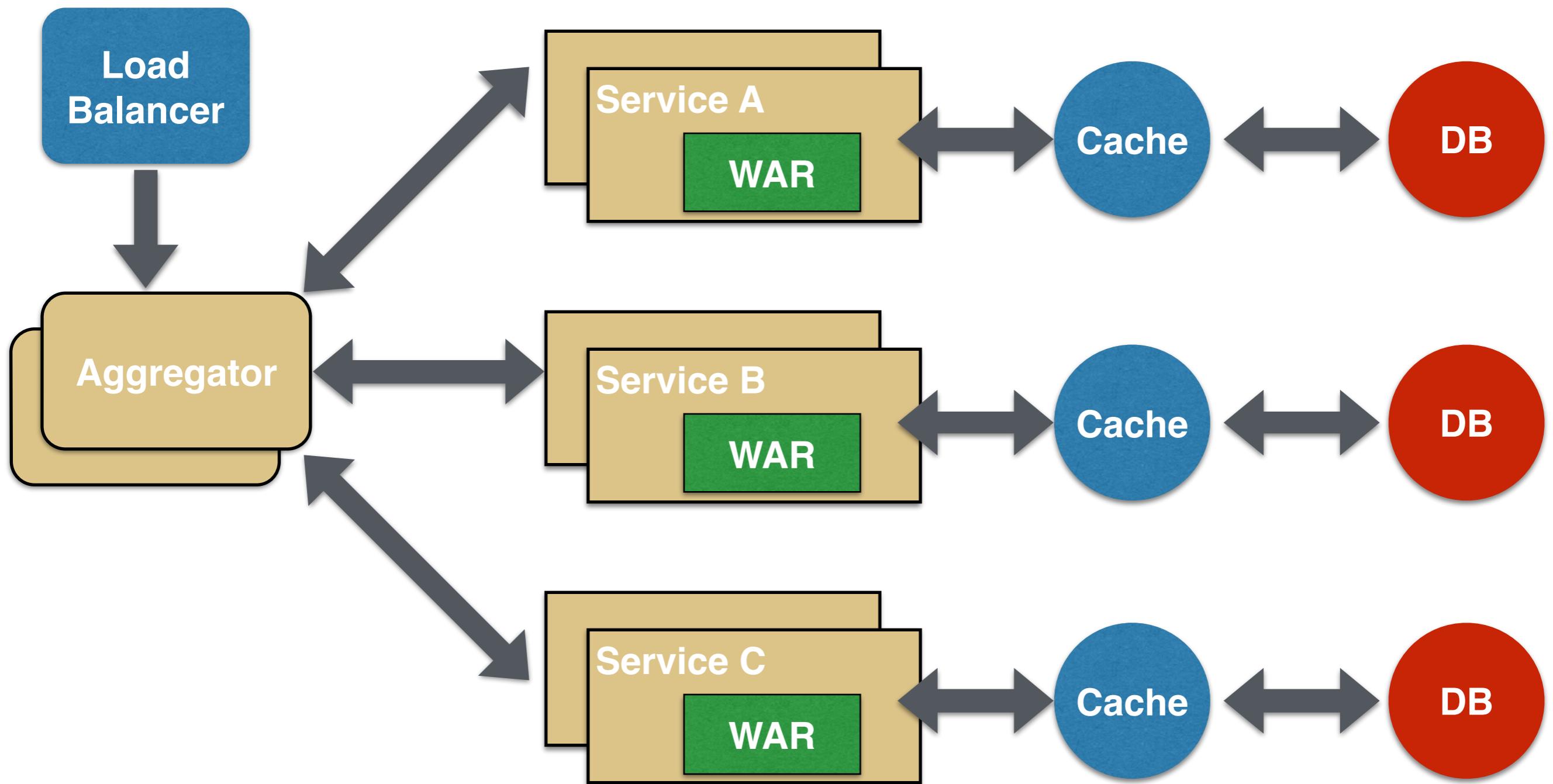
Strategies for decomposing

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

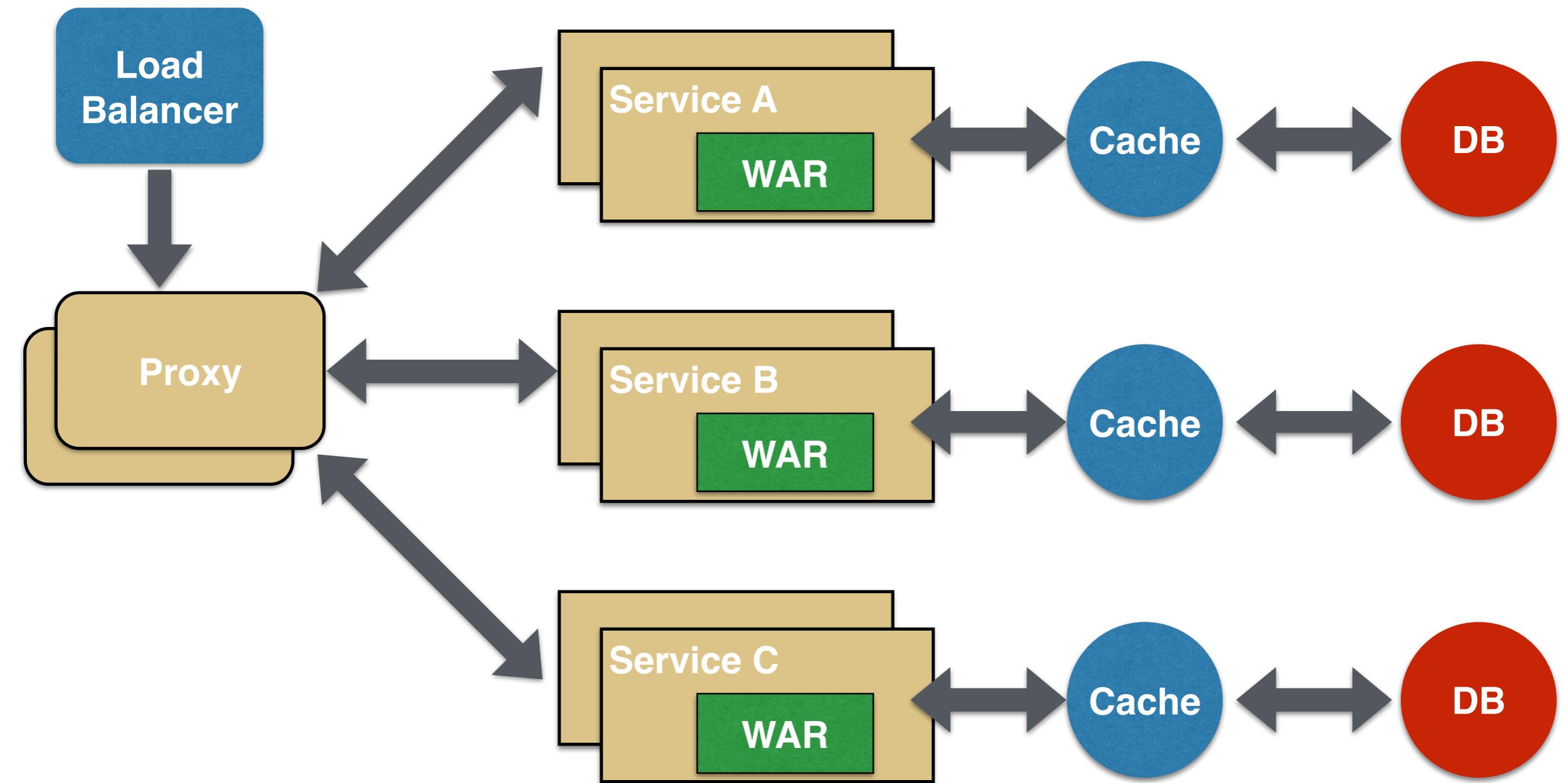
Towards microservices



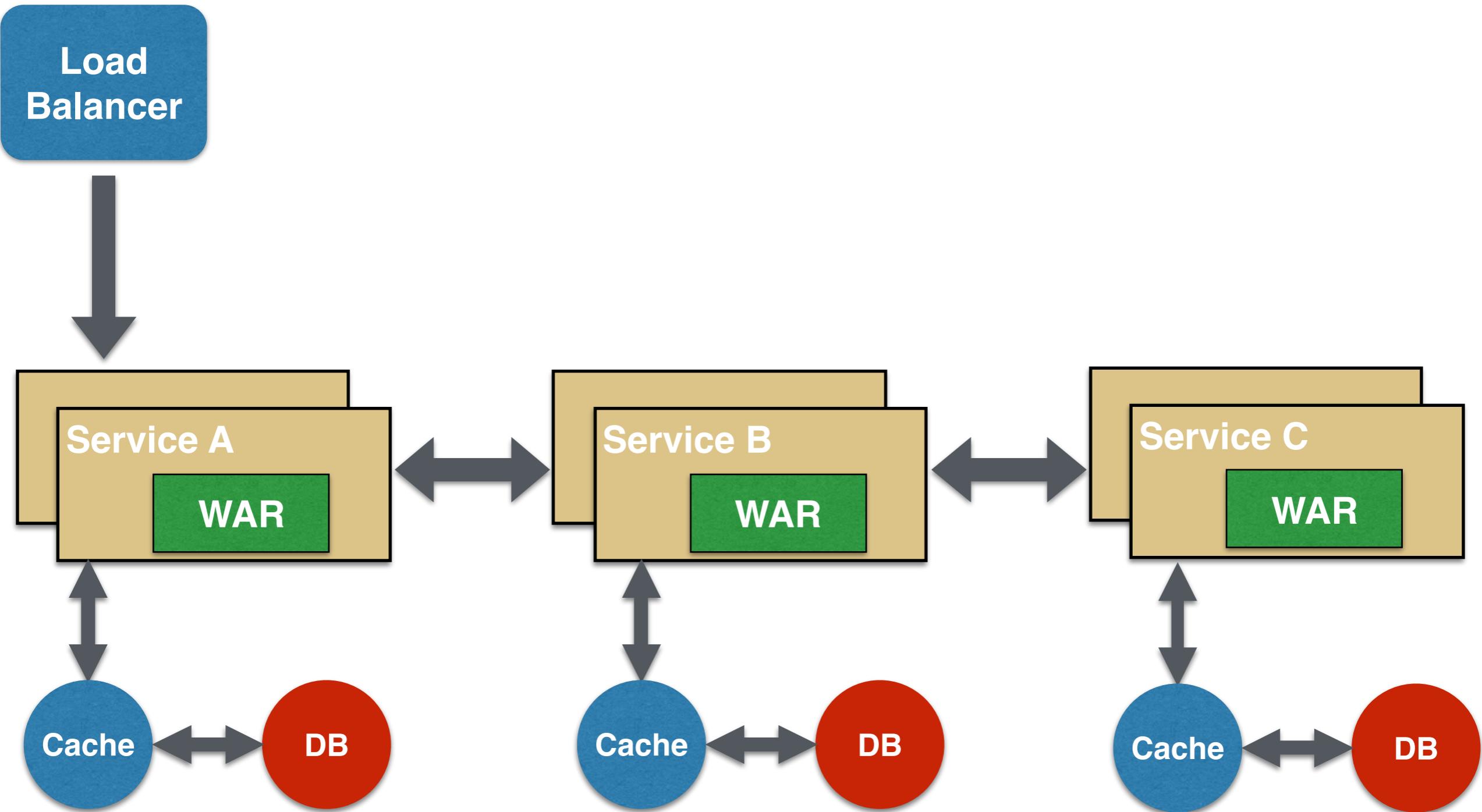
Aggregator Pattern #1



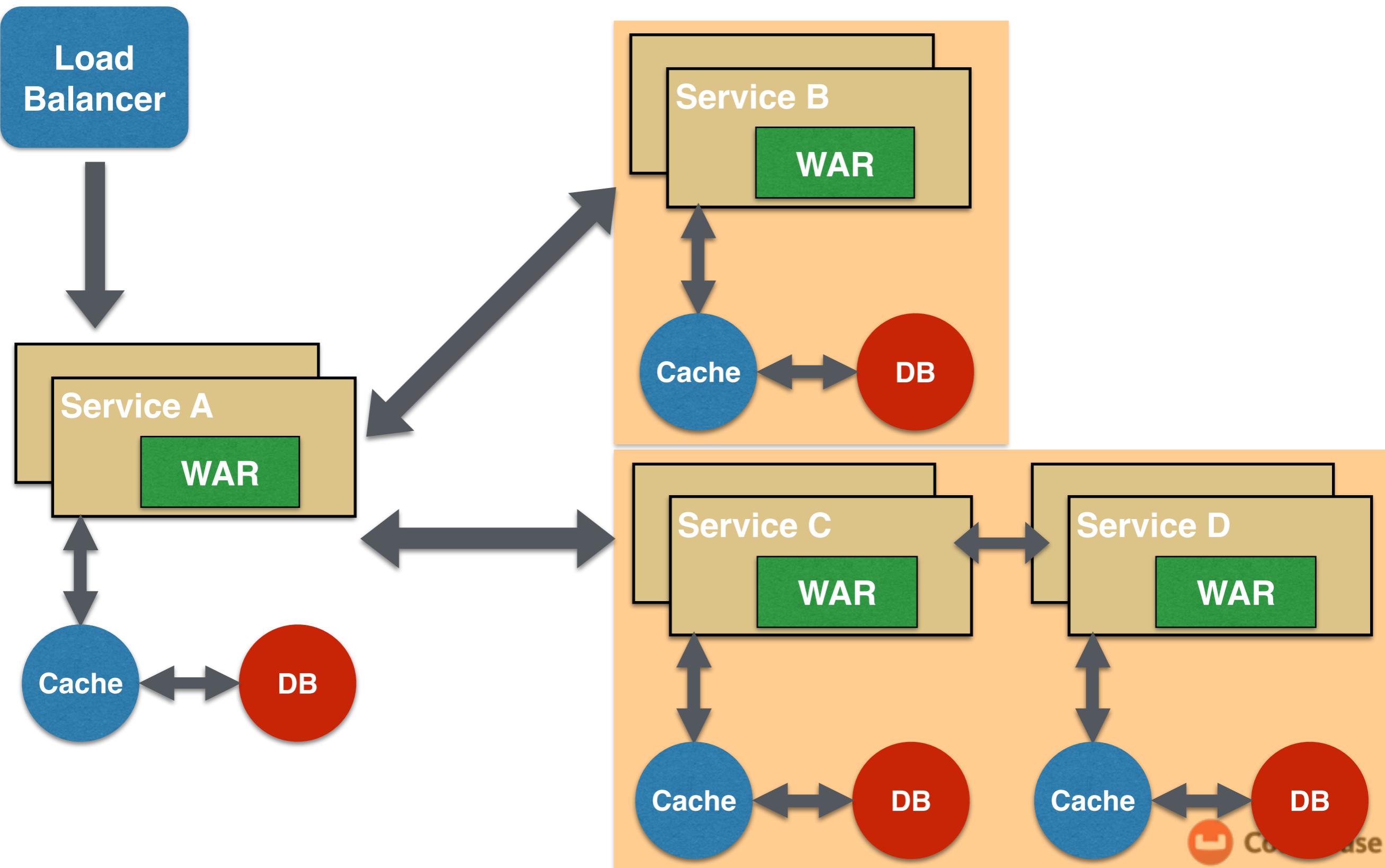
Proxy Pattern #2



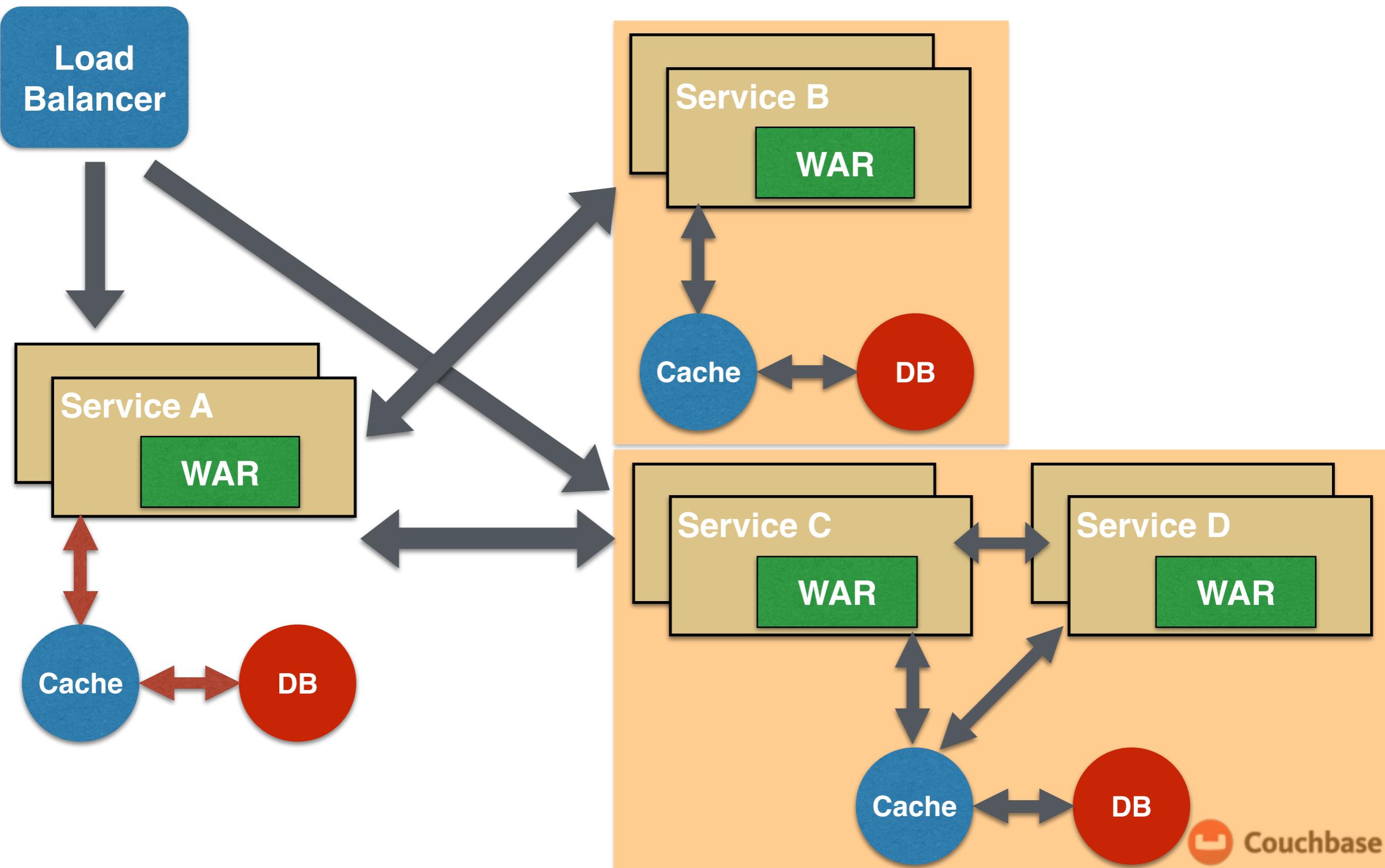
Chained Pattern #3



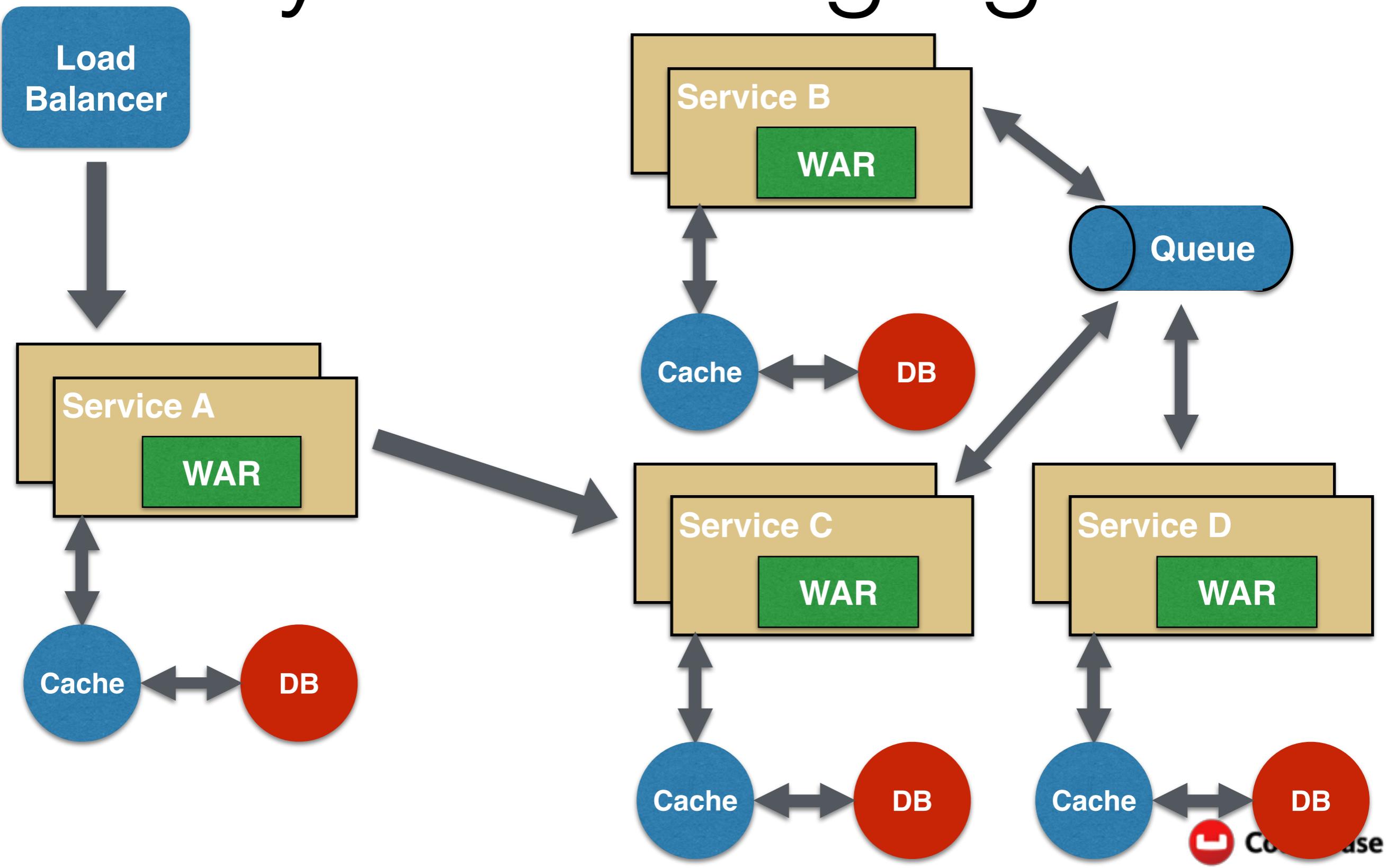
Branch Pattern #4



Shared Resources #5



Async Messaging #5





GETTING STARTED WITH Microservices

DZONE REF CARD #215

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 let you make modular updates and increase the speed of application deployments.

Free PDF

DOWNLOAD

SAVE

5,937

dzone.com/refcardz/getting-started-with-microservices

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?”

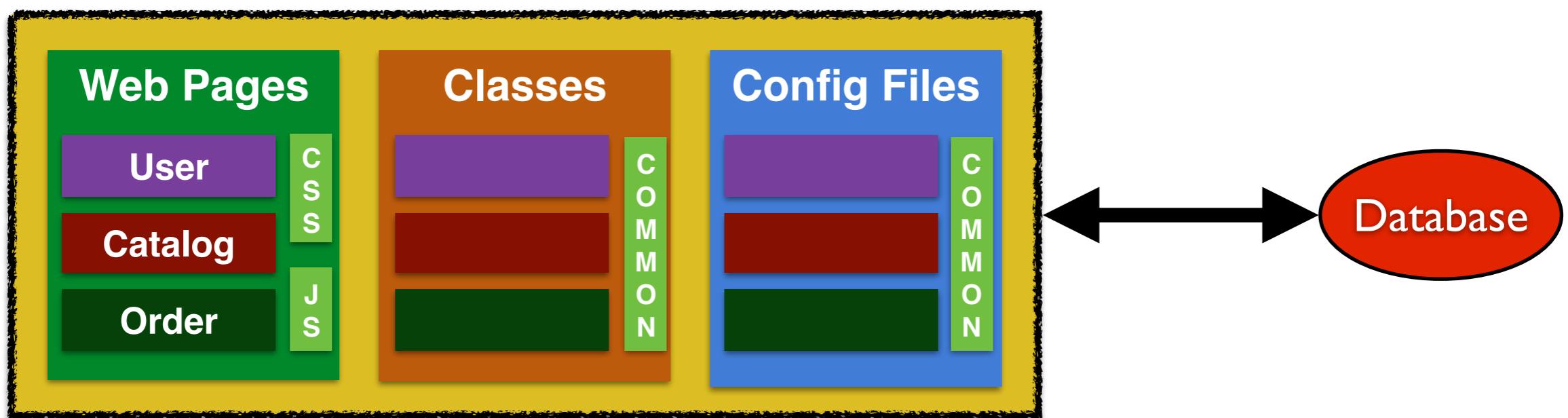
http://www.codingthearchitecture.com/2014/07/06/distributed_big_balls_of_mud.html

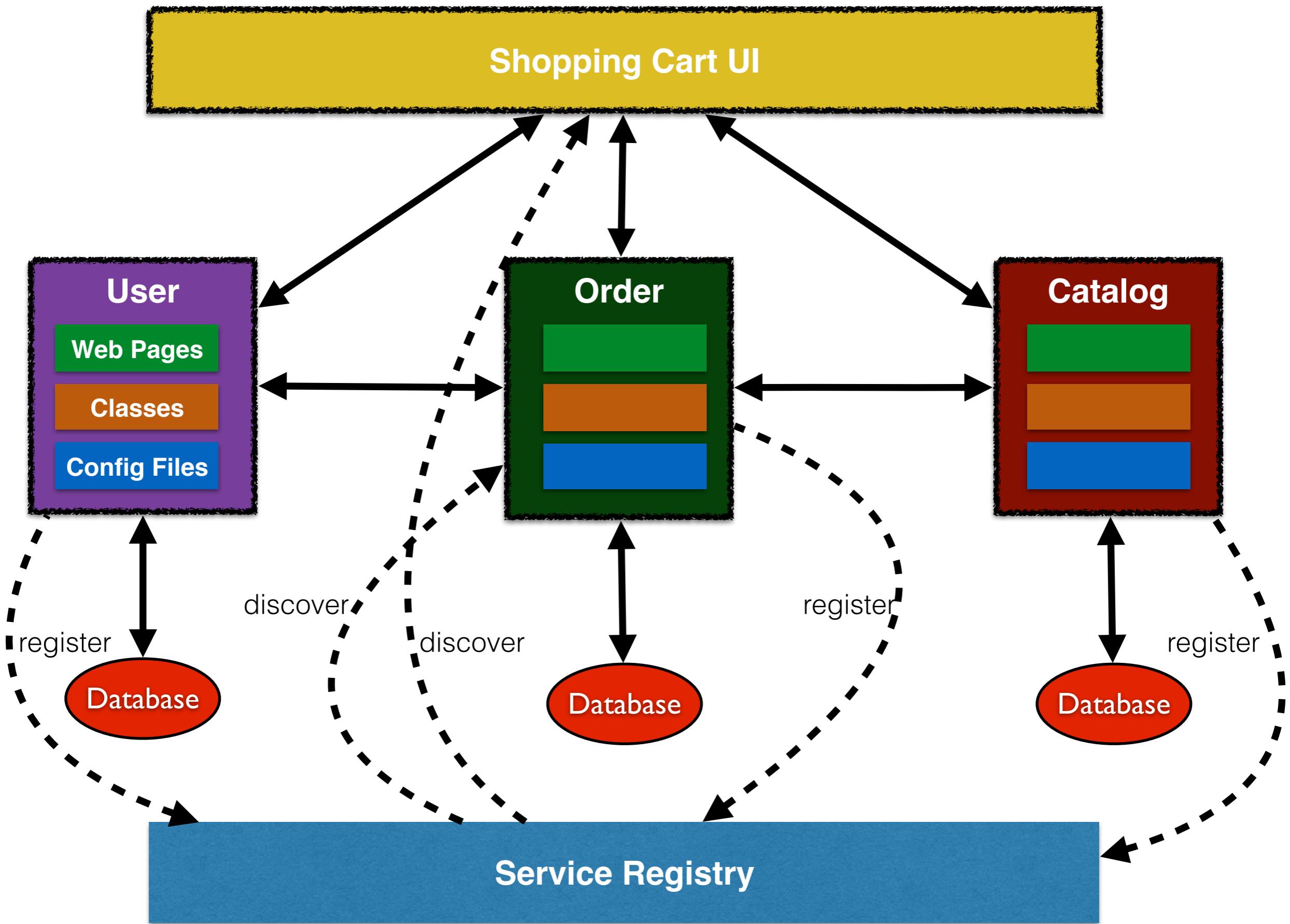
“If your monolith is a big ball of mud, your microservice will be a bag of dirt”

Arun Gupta

Design Principles for Monoliths

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





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

Design Considerations

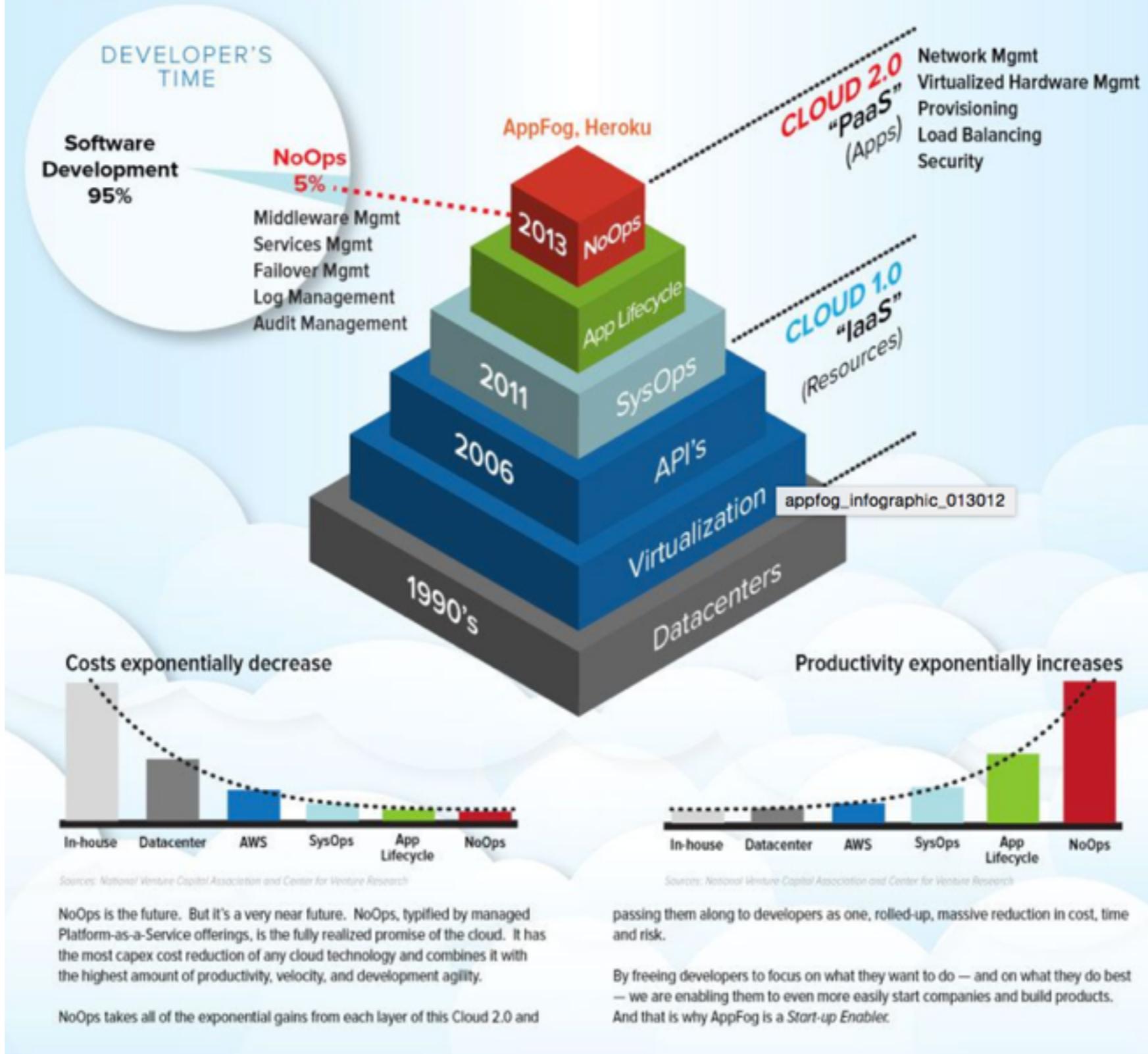
- UI and Full stack
 - Client-side composition (JavaScript?)
 - Server-side HTML generation (JSF?)
 - One service, one UI
- REST Services
- Compensating transactions instead of 2PC

NoOps

- Service replication (k8s, fabric8, etcd, ZK, ...)
- Dependency resolution (Nexus, ...)
- Failover (Circuit Breaker)
- Resiliency (Circuit Breaker)
- Service monitoring, alerts and events (New Relic, Log stash, ...)

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.



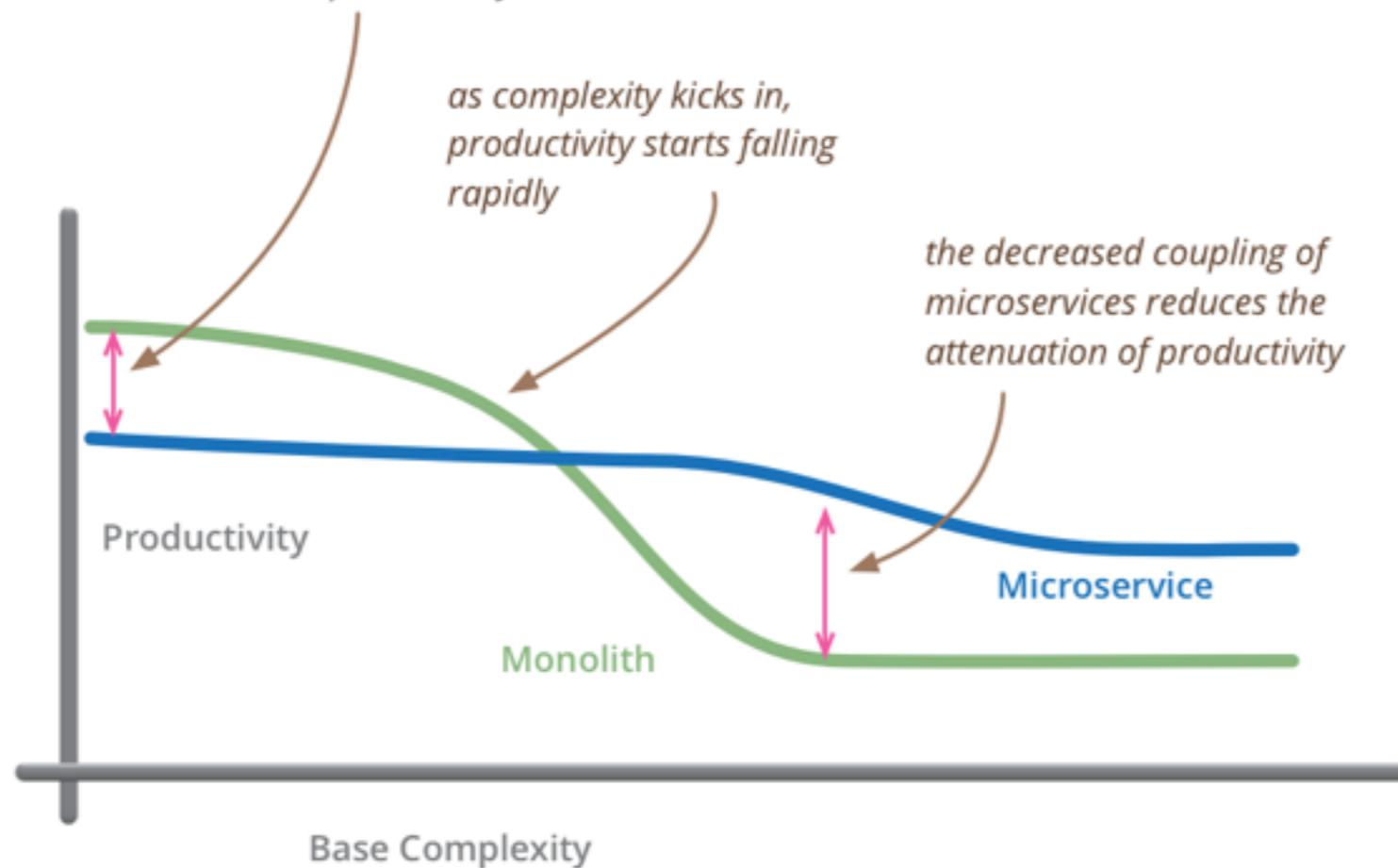
<https://gigaom.com/2012/01/31/why-2013-is-the-year-of-noops-for-programmers-infographic/>

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



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