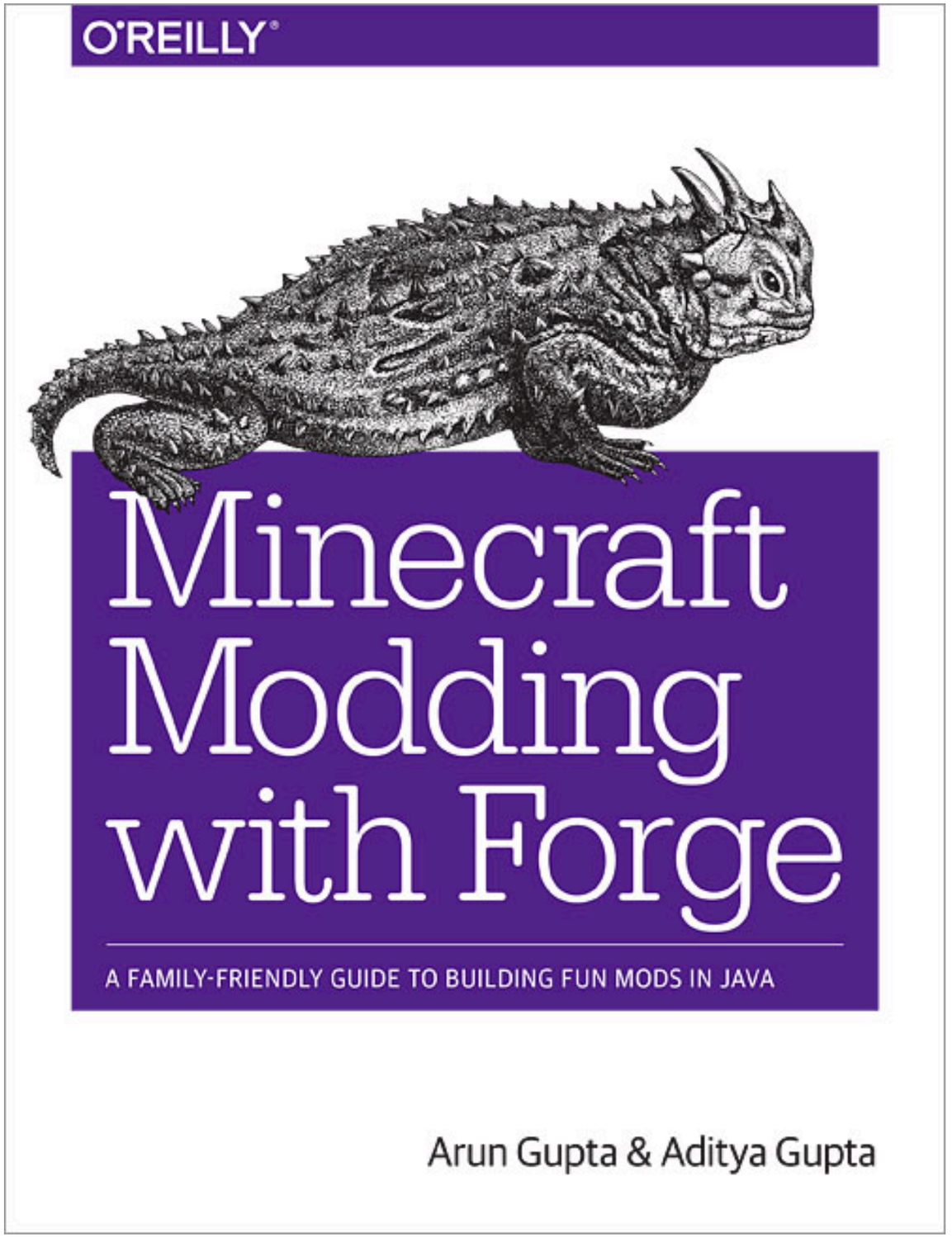
Refactoring your Java EE applications using Microservices and Containers

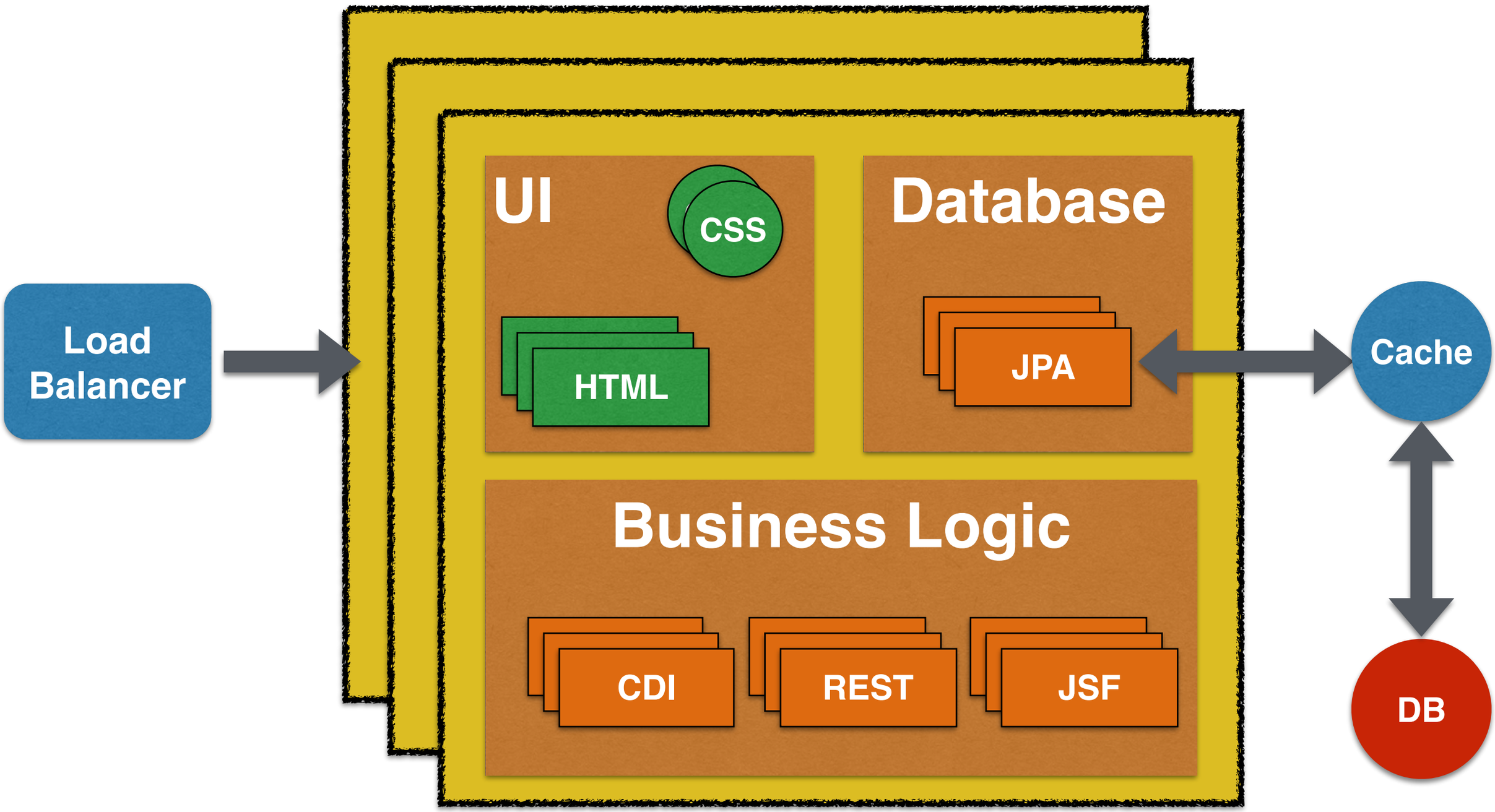
Arun Gupta
Vice President, Developer Advocacy
@arungupta, blog.arungupta.me
arun@couchbase.com



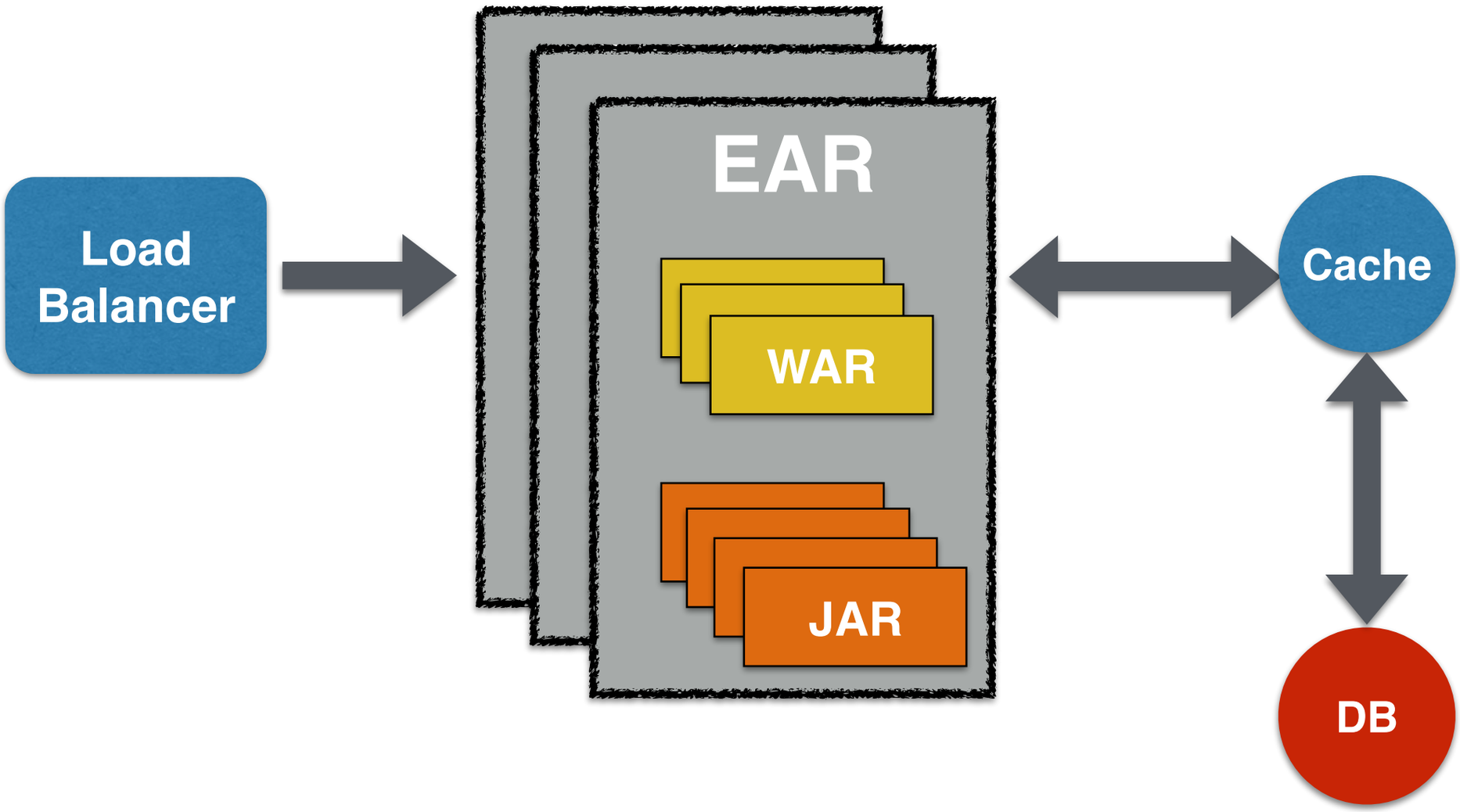
DEVOXX™ 4KIDS USA

Couchbase

Monolith Application



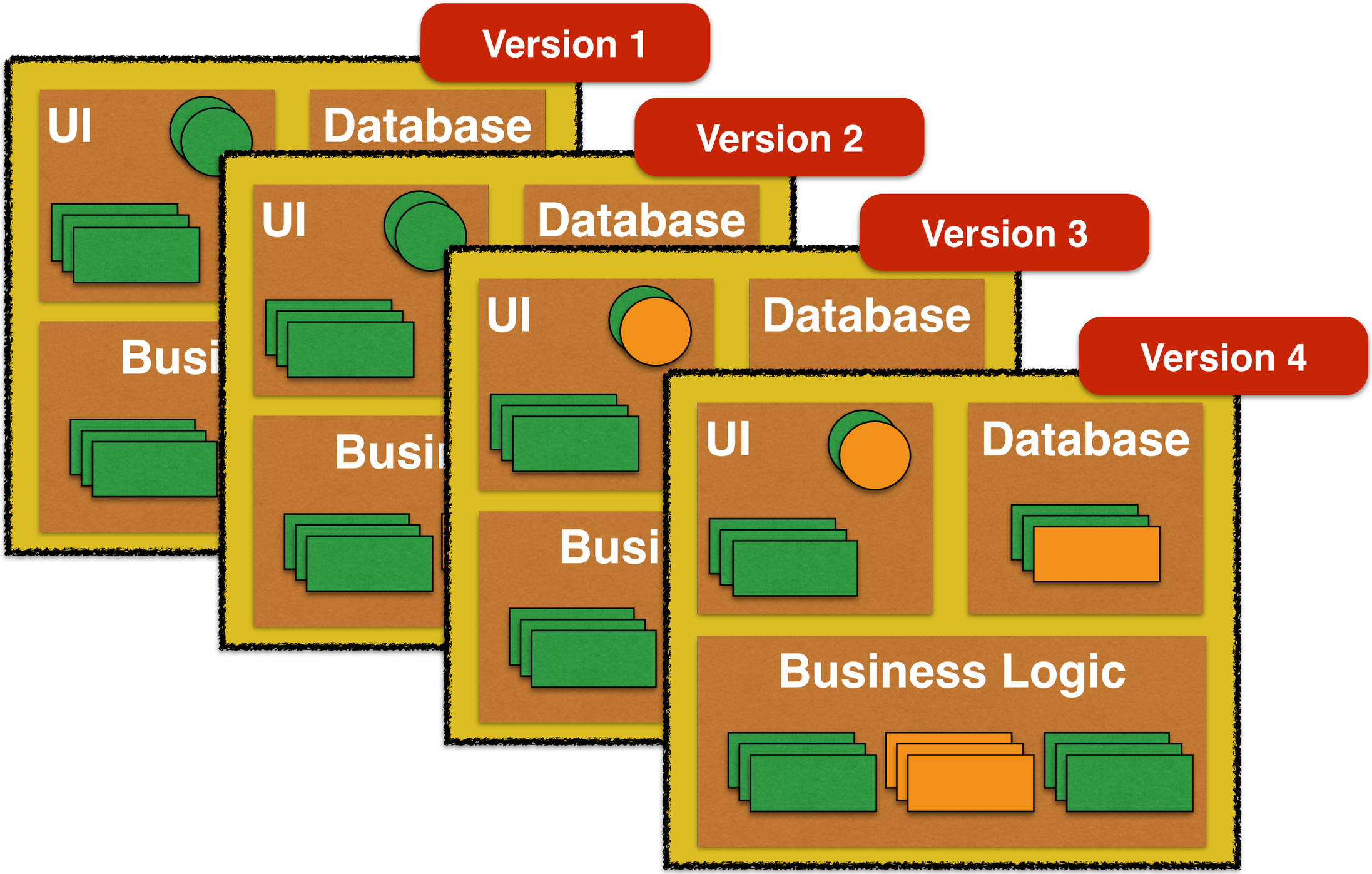
Monolith Application



Advantages of Monolith Application

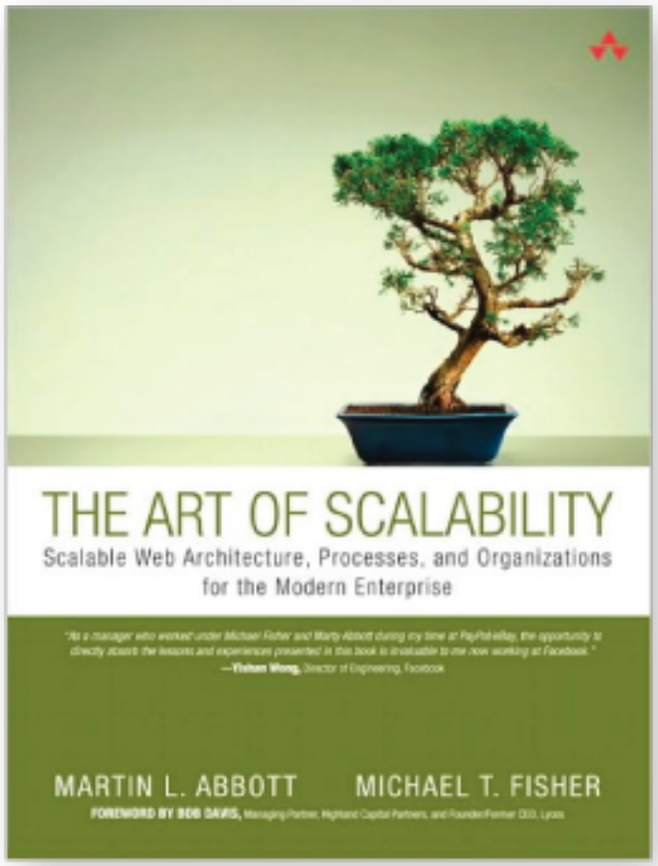
- Typically packaged in a single `.ear`
- Easy to test (all required services are up)
- Simple to develop

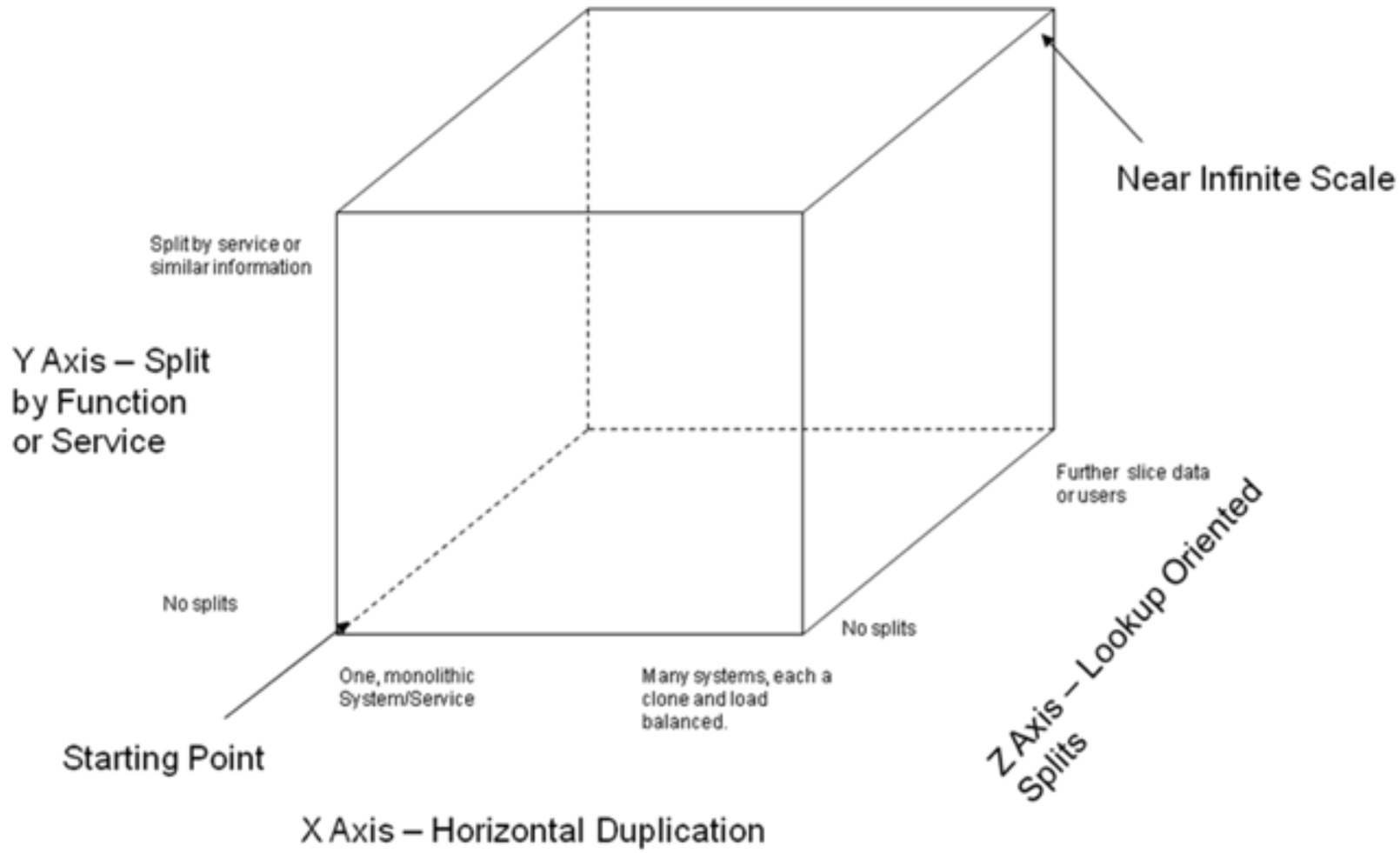
Monolith Application



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

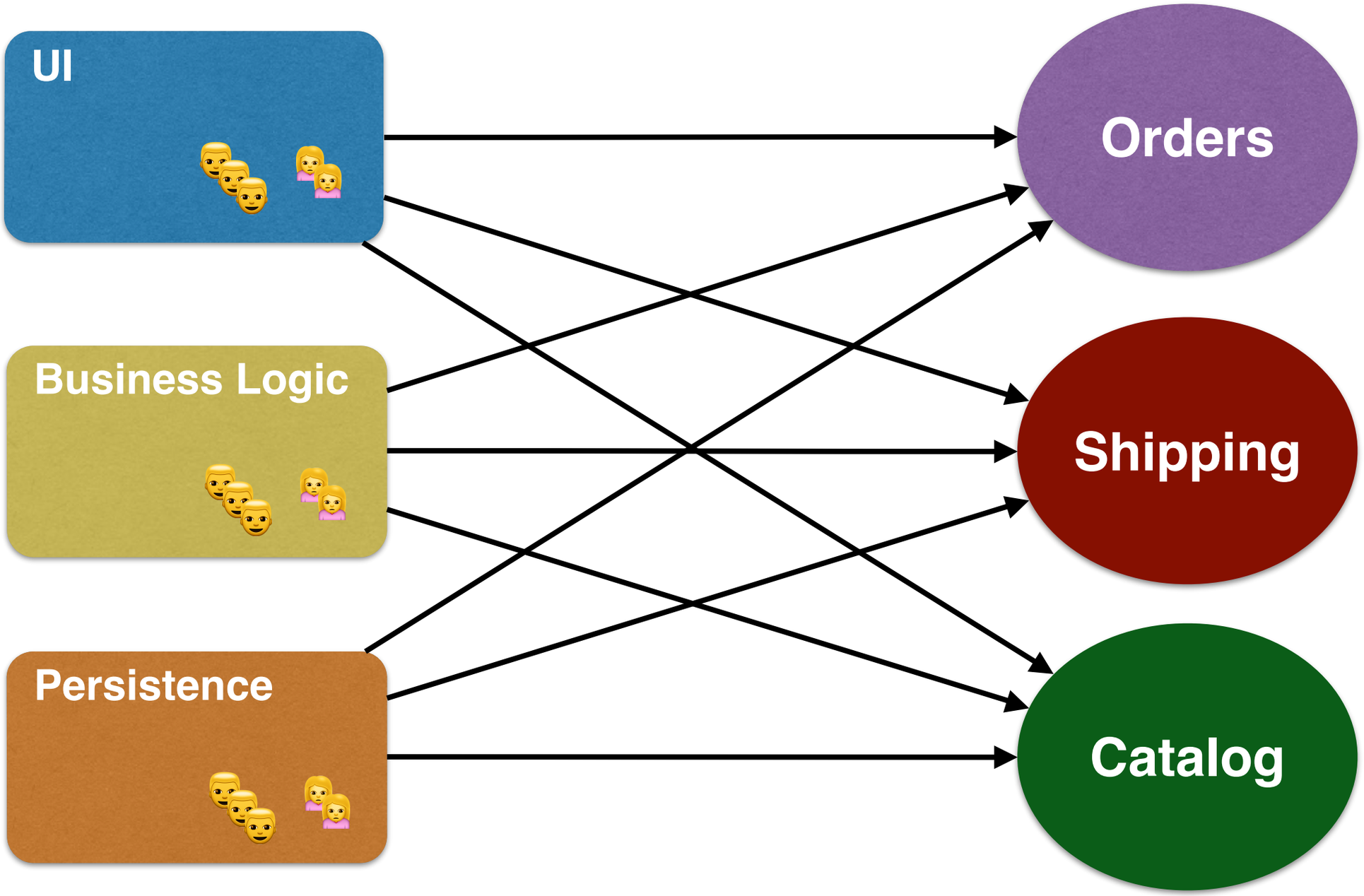




http://akfpartners.com/techblog/2008/05/08/splitting-applications-or-services-for-scale/

*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
BUT WHEN I DO I
USE MICRO SERVICES
imgflip.com


UI
Business Logic
Persistence
Orders
Shipping
Catalog

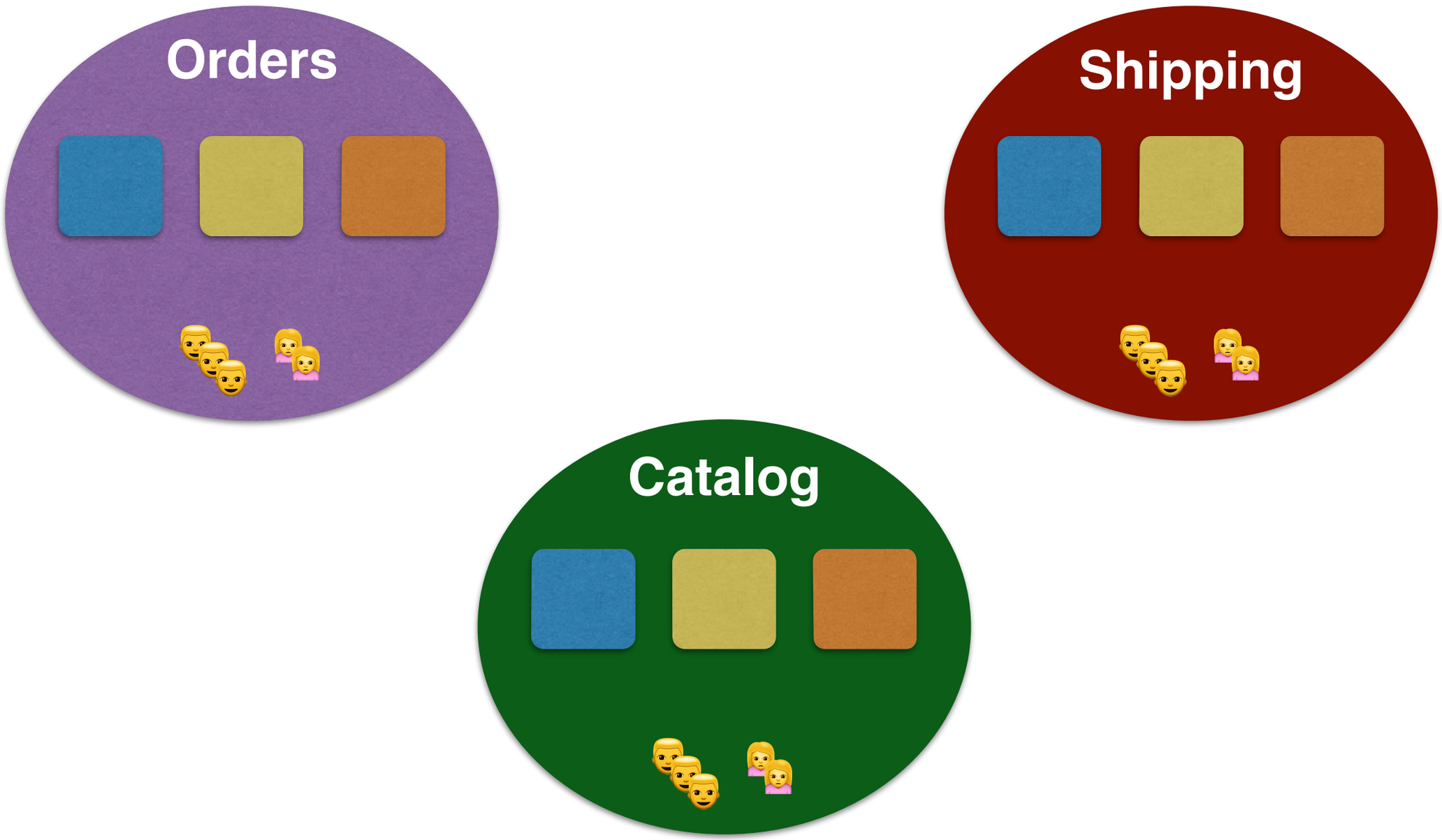
*"Any **organization that designs a system** (defined more broadly here than just information systems) will inevitably produce a design whose structure is a **copy of the organization's communication structure**."*

–Melvin Conway

http://www.melconway.com/Home/Committees_Paper.html

Characteristics

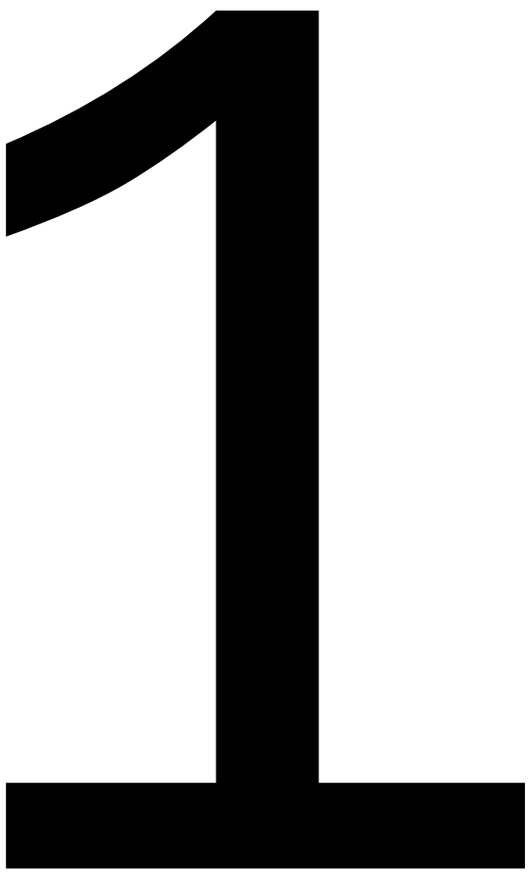
Teams around business capability



Characteristics

Single Responsibility Principle

DO



THING

Characteristics

Explicitly Published interface

Characteristics

Independently replaceable and upgradeable

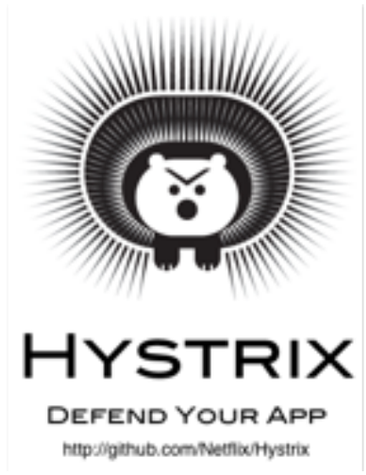
Characteristics

With great power, comes great responsibility

"you build it, you run it!"

Characteristics

Designed for failure



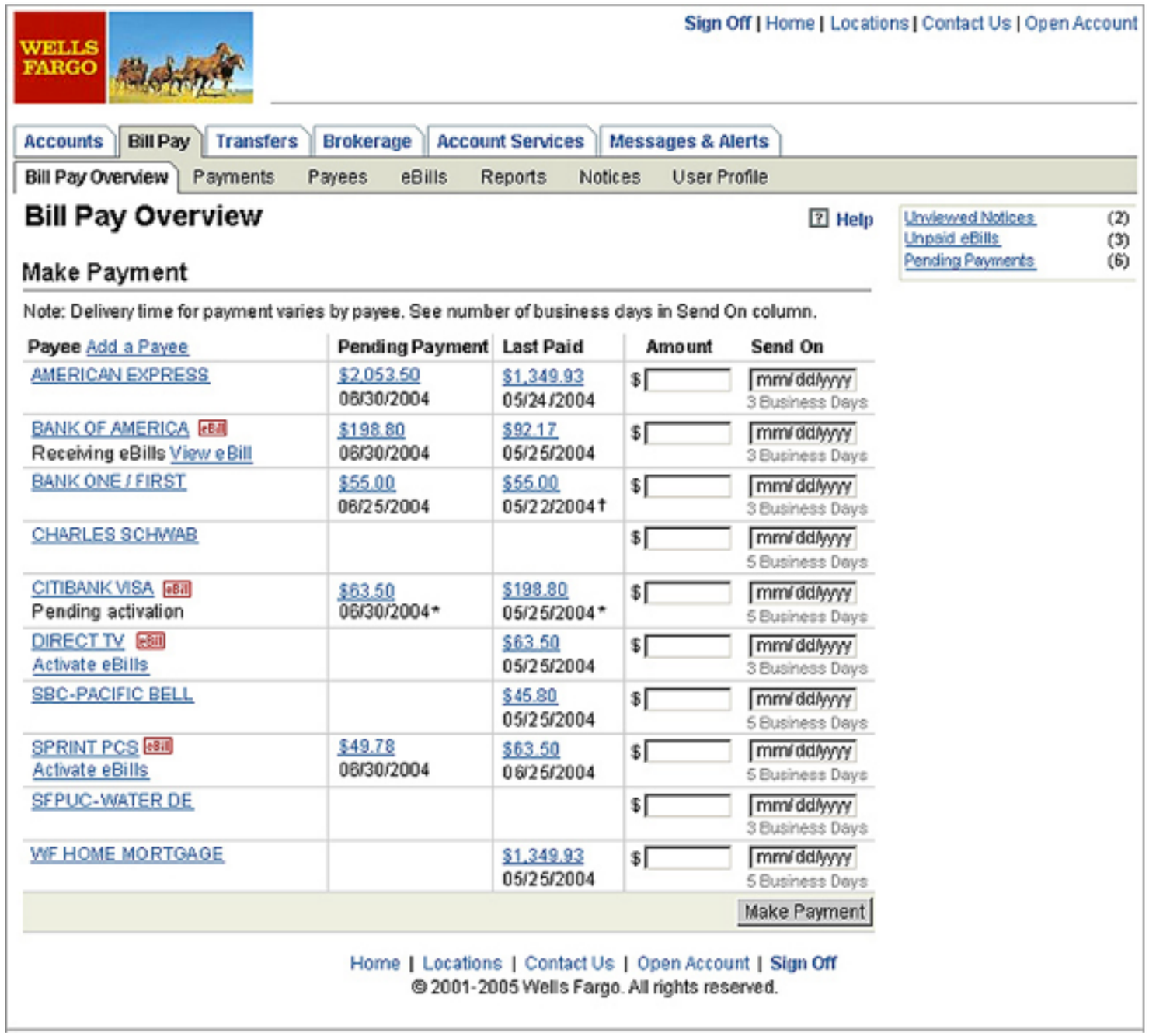
Fault tolerance is a requirement, not a feature

http://techblog.netflix.com/2012/02/fault-tolerance-in-high-volume.html

Characteristics

Characteristics

100% automated

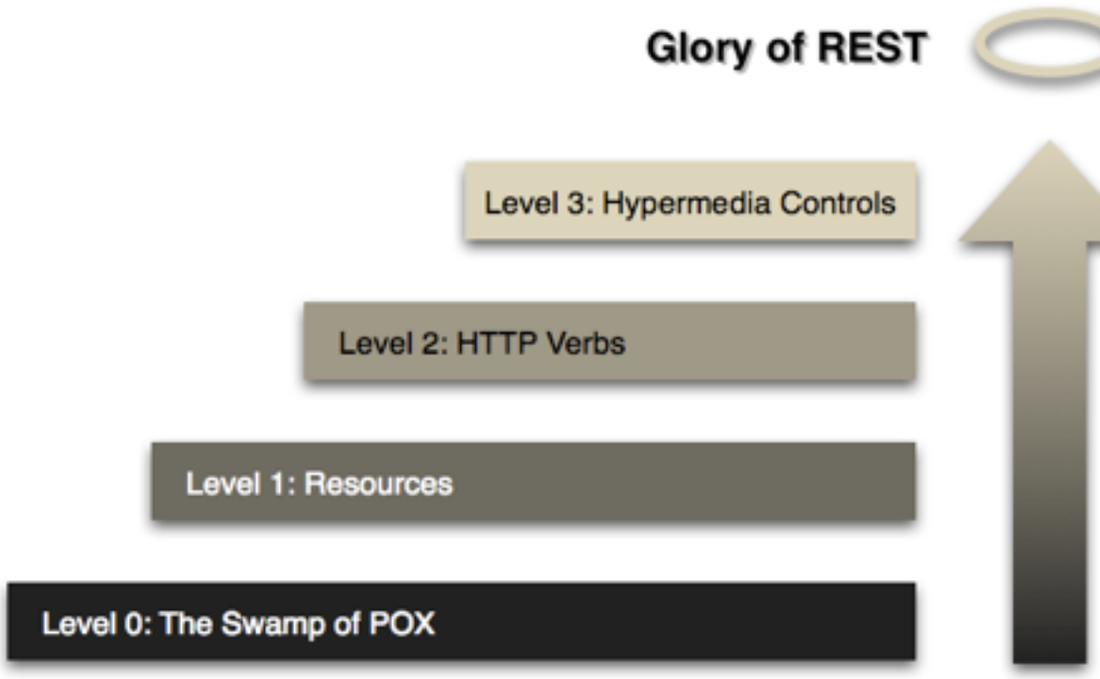


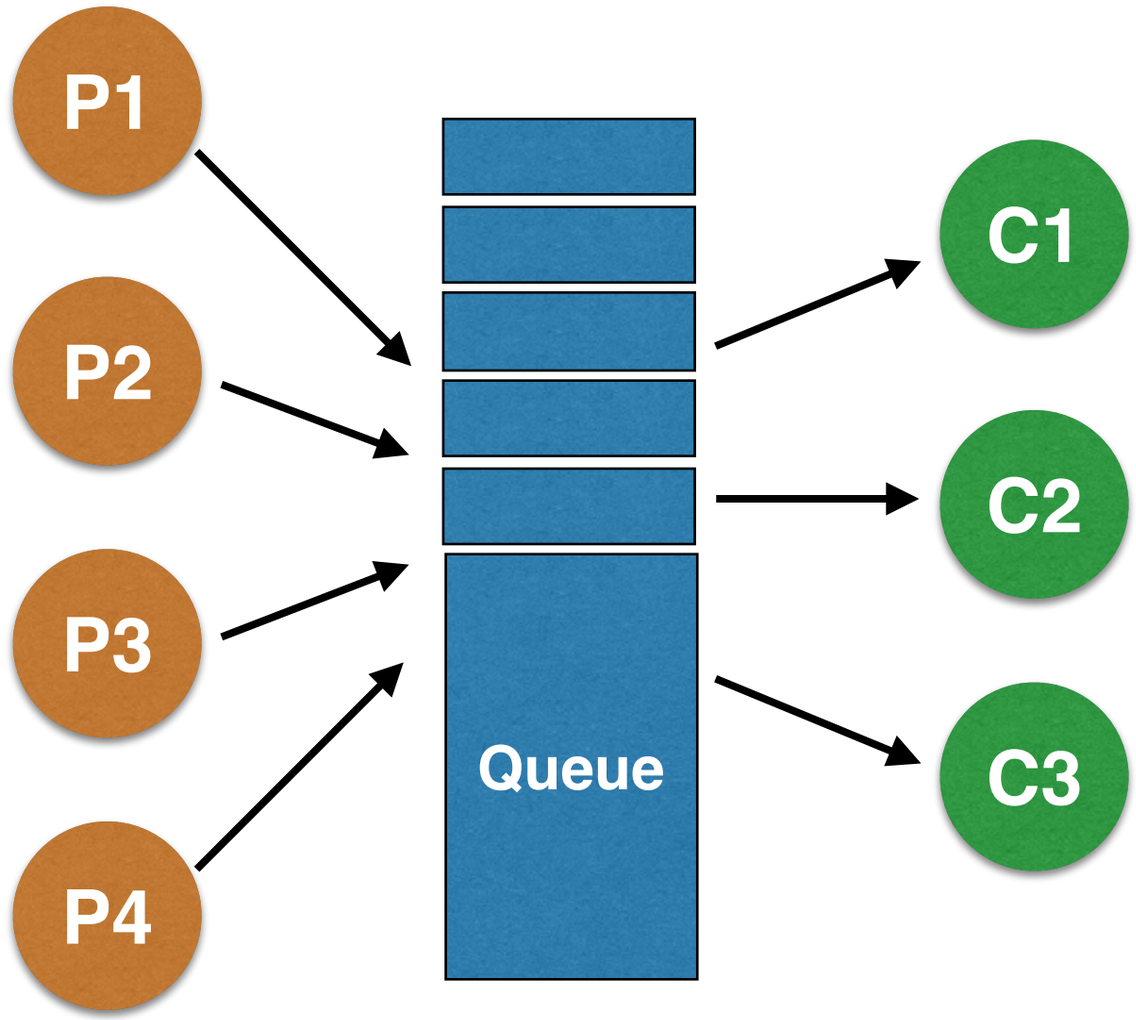
Characteristics

Sync or Async Messaging

Characteristics

Sync vs Async





Thrift

Protobuf

Avro

Characteristics

“Smart endpoints Dumb pipes”

SOA

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

SOA 2.0?

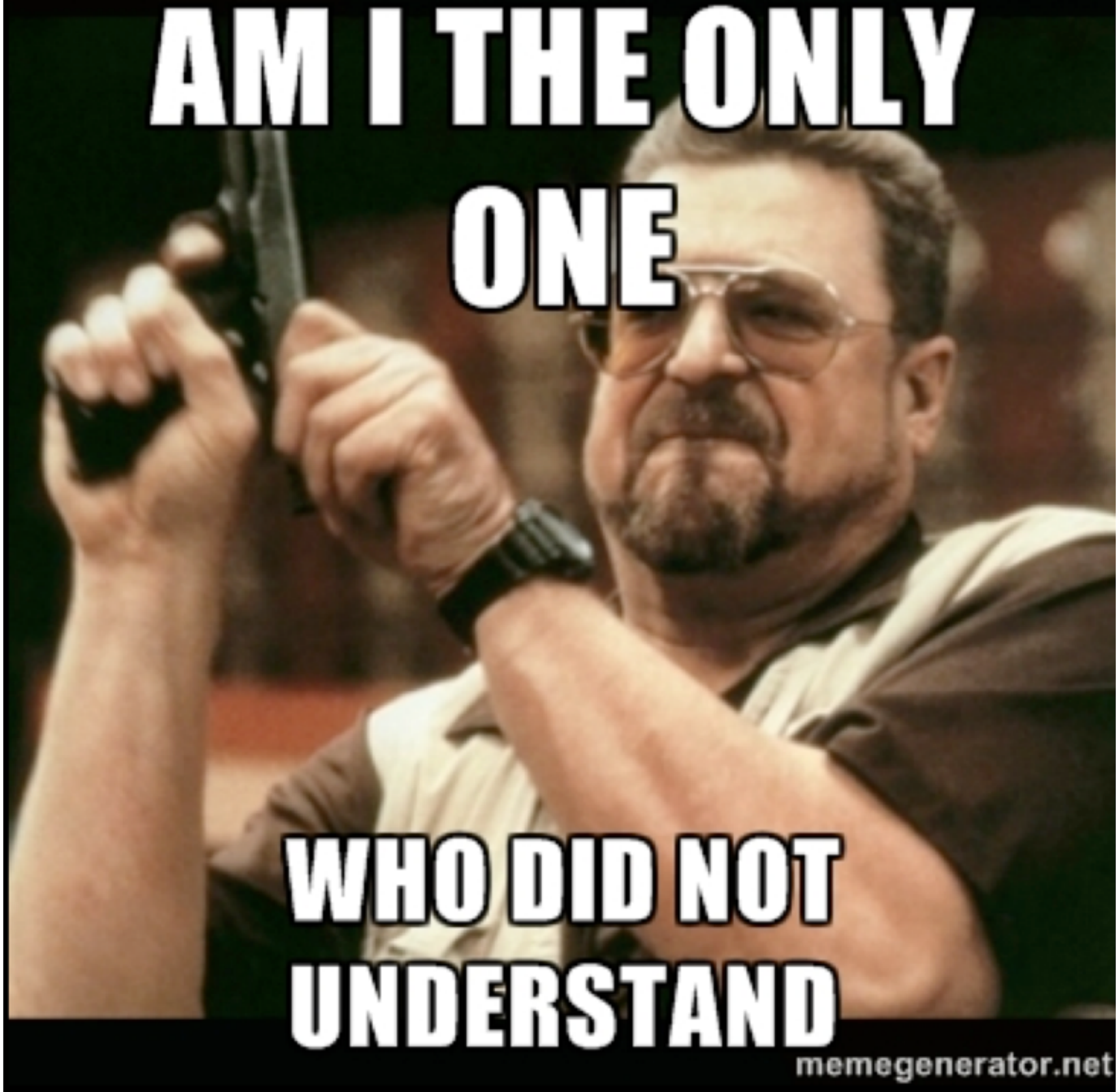
- Conway's Law
- Service Discovery
- Immutable VM

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

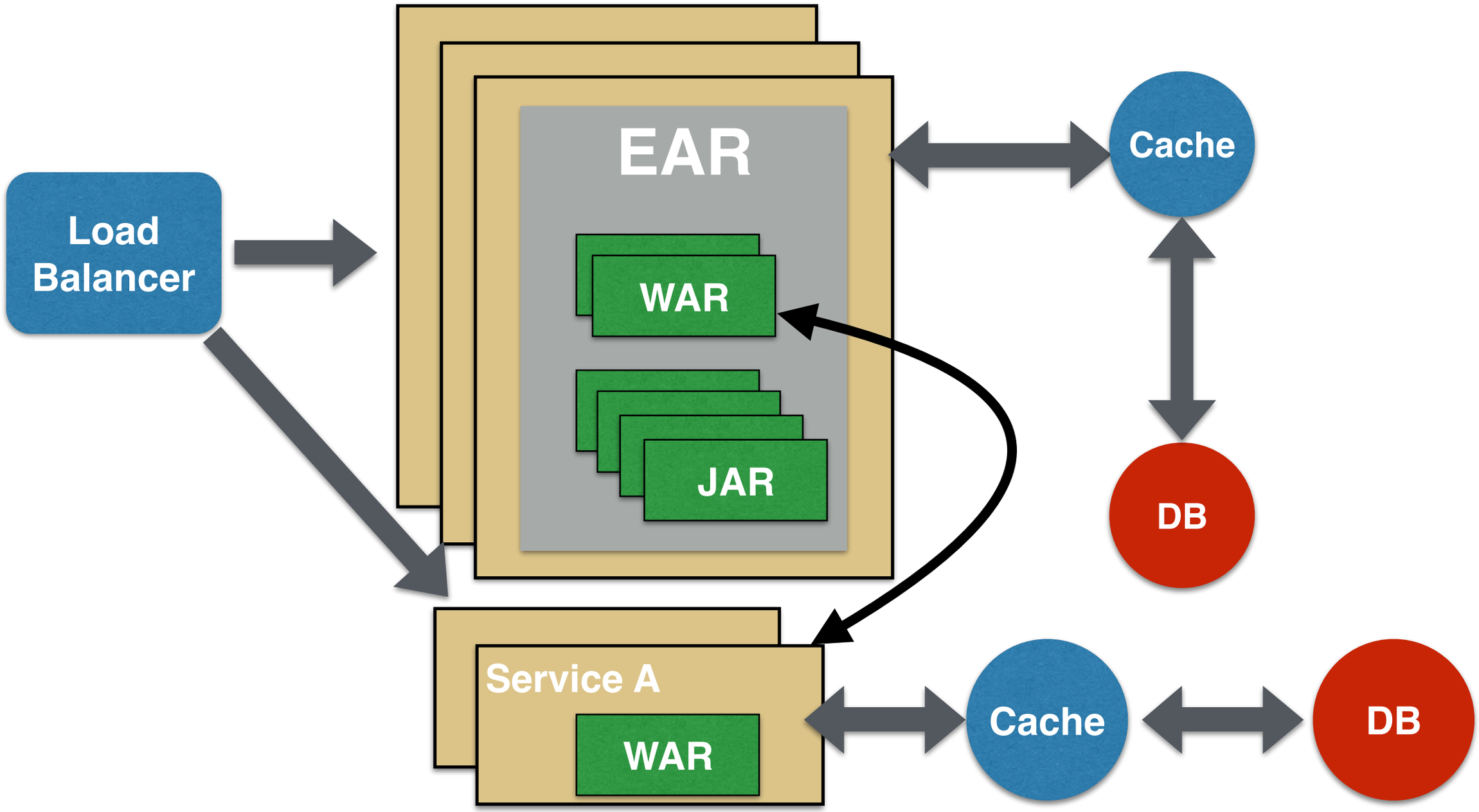
5:07 PM - 27 May 2015


AM I THE ONLY ONE
WHO DID NOT UNDERSTAND
memegenerator.net

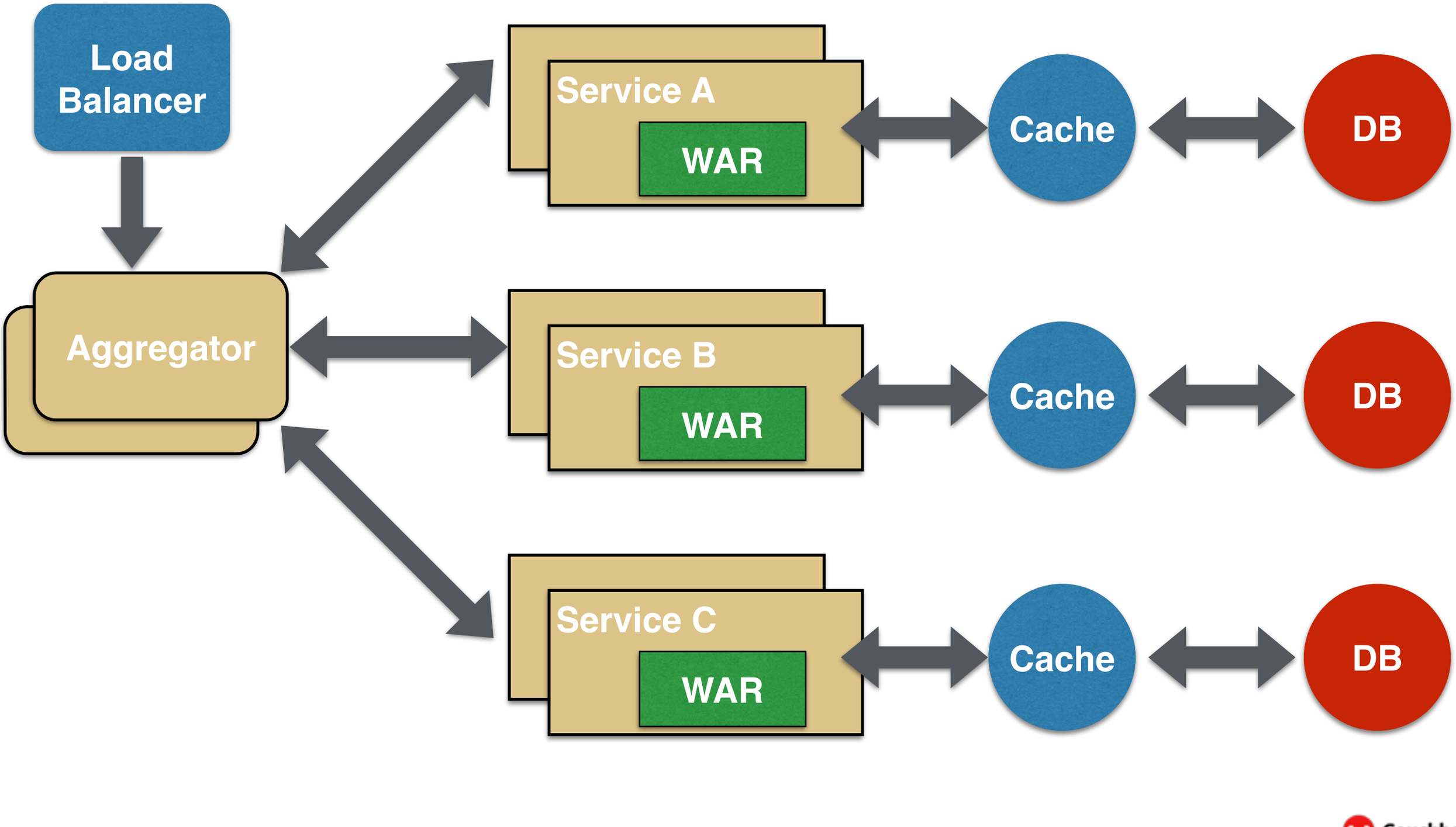
Strategies for decomposing

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

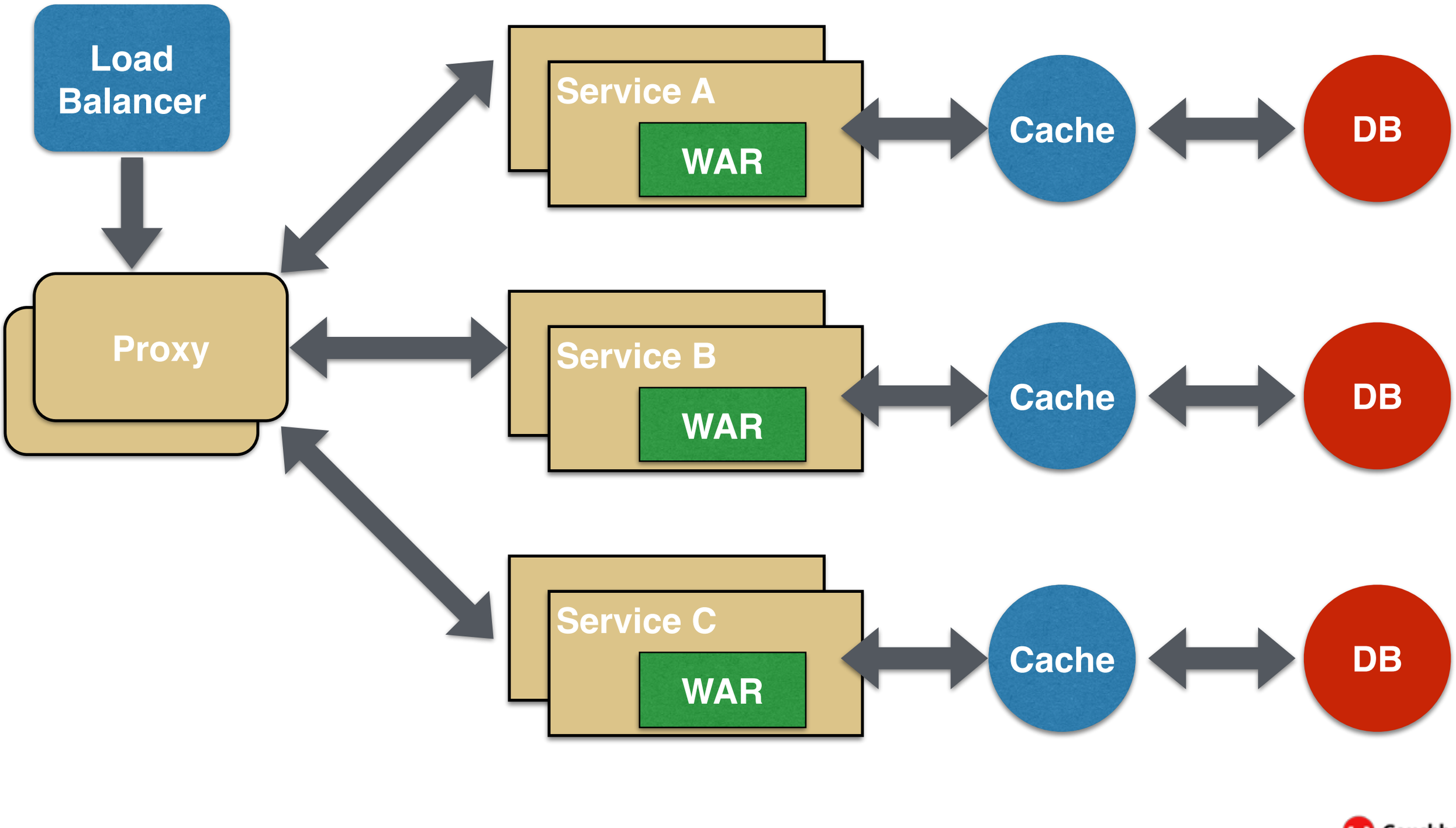
Towards microservices



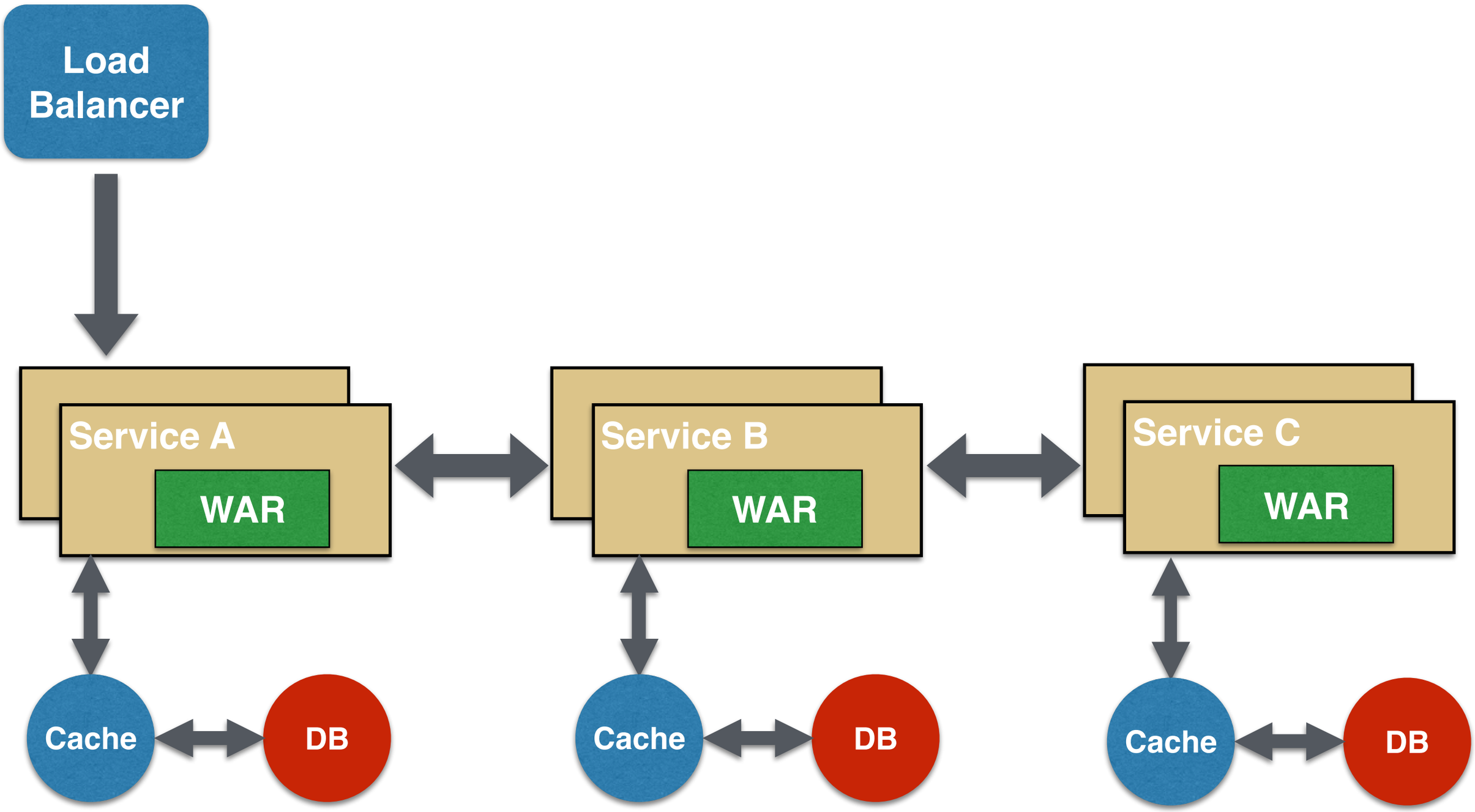
Aggregator Pattern #1



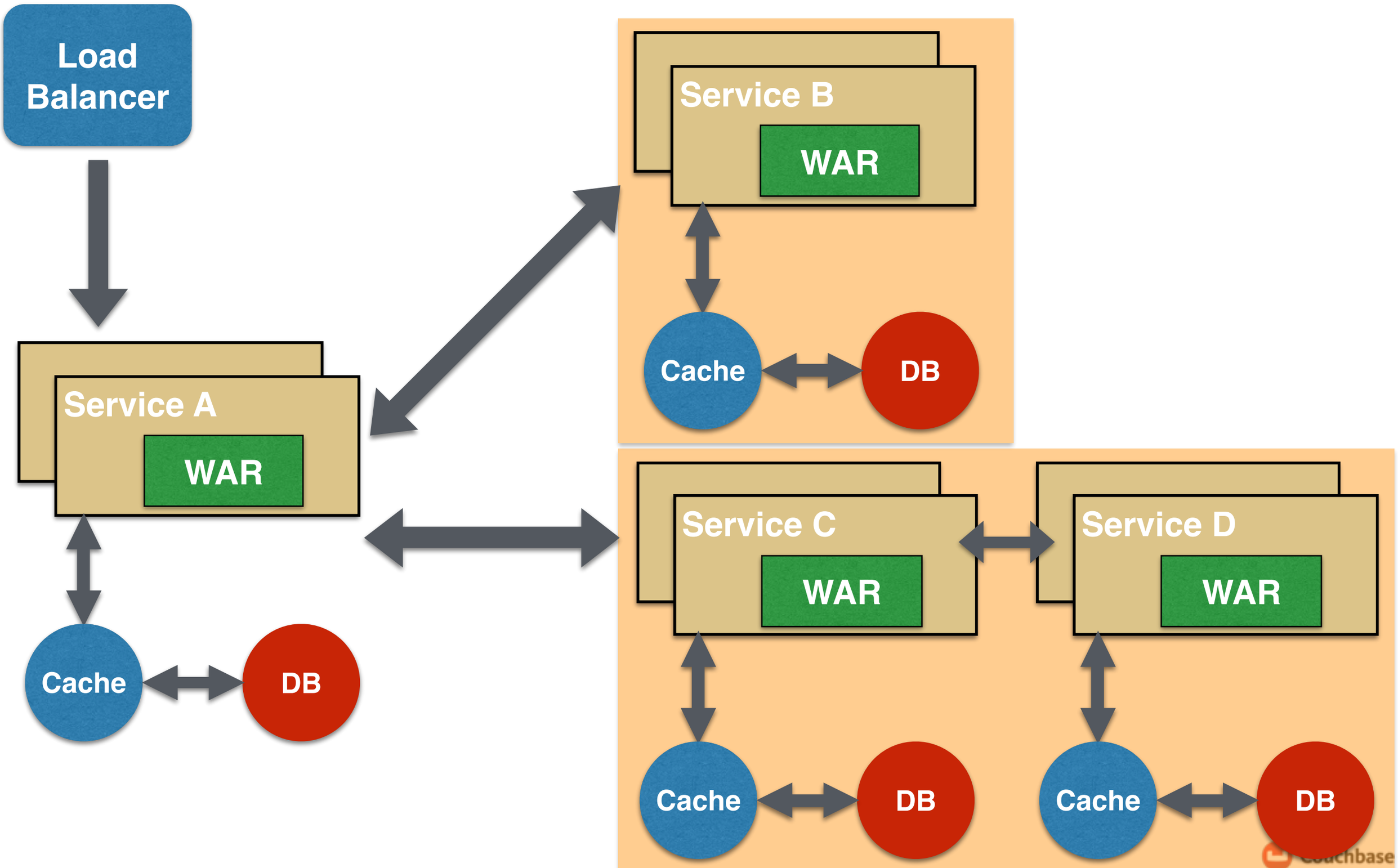
Proxy Pattern #2



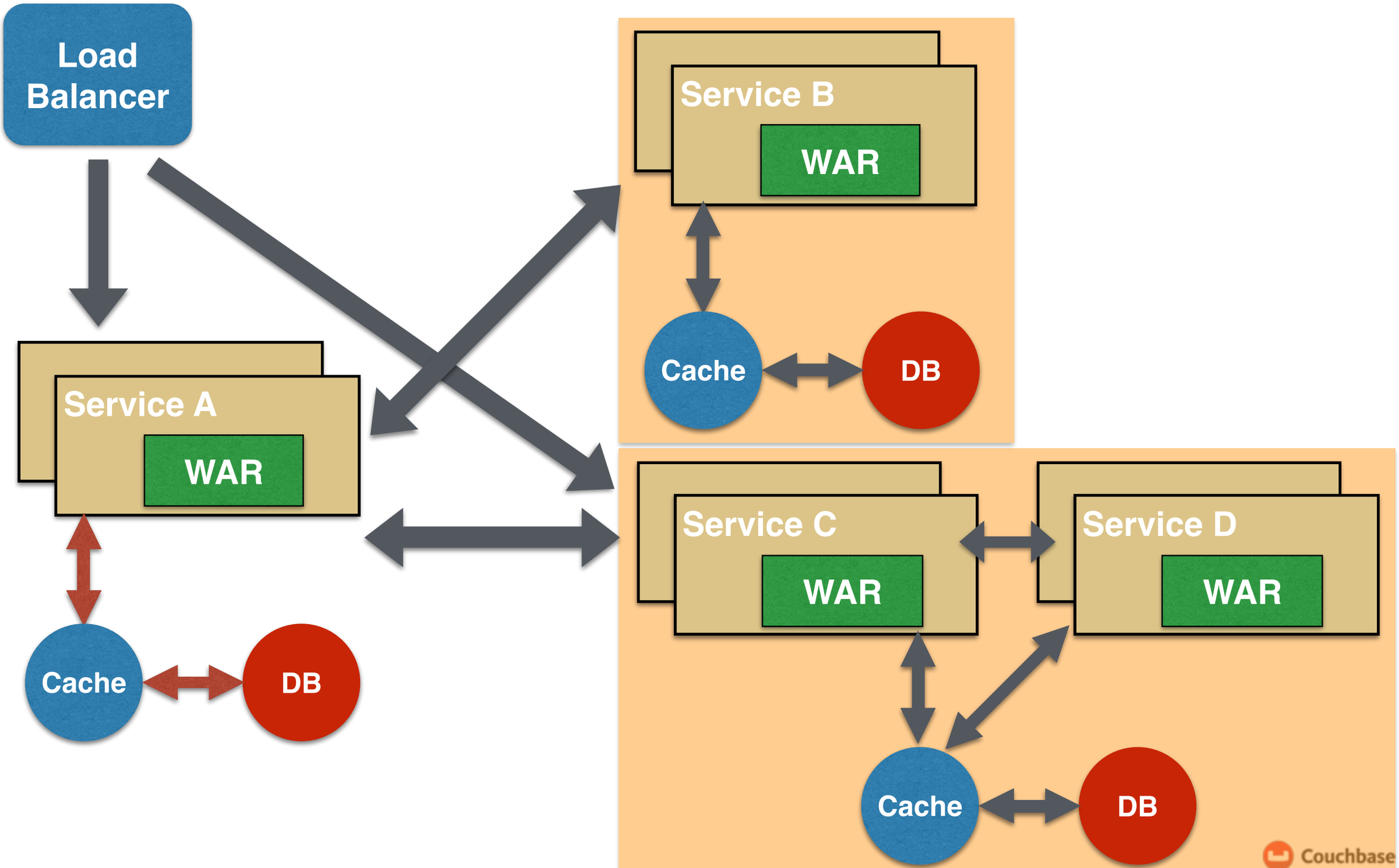
Chained Pattern #3



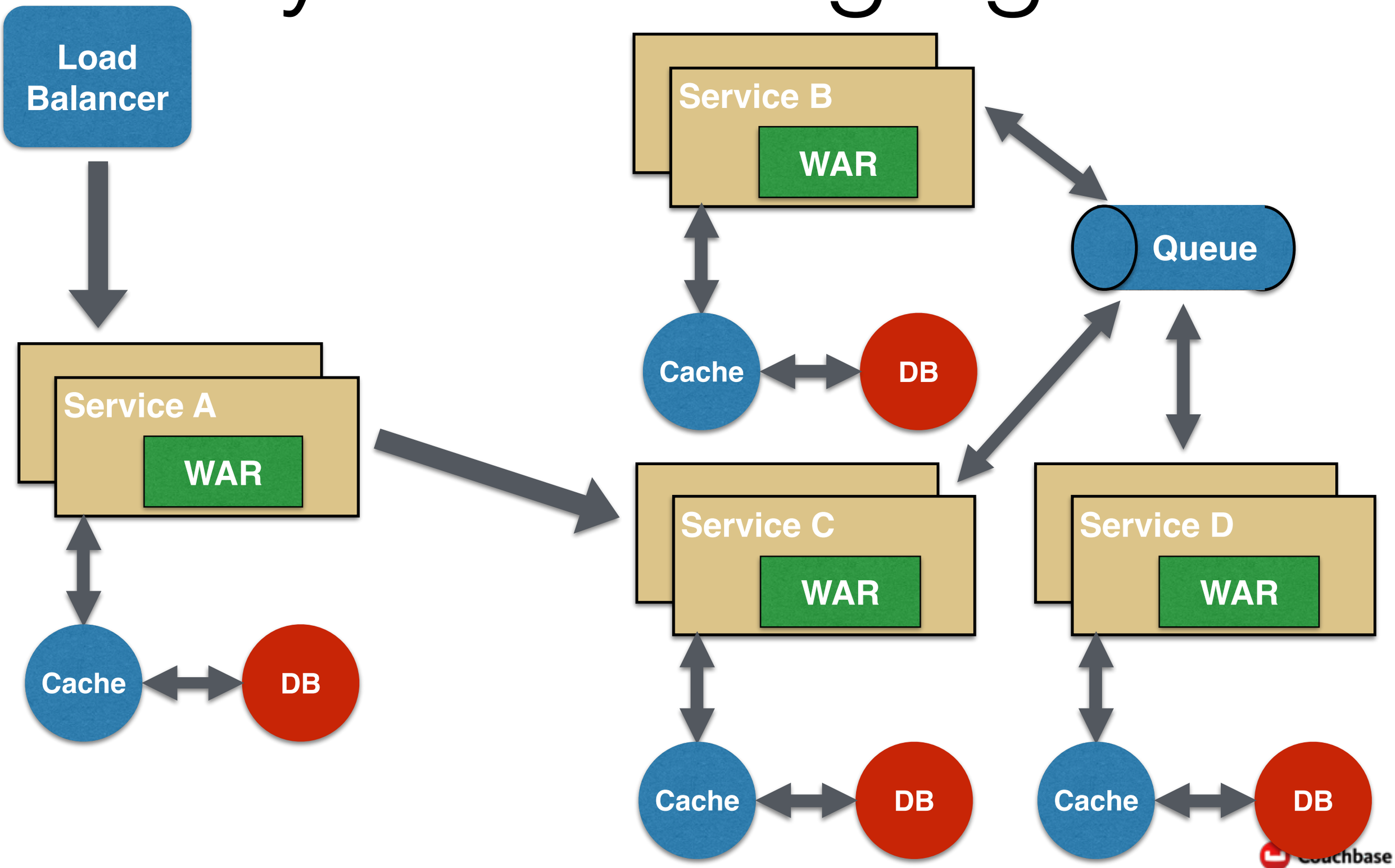
Branch Pattern #4

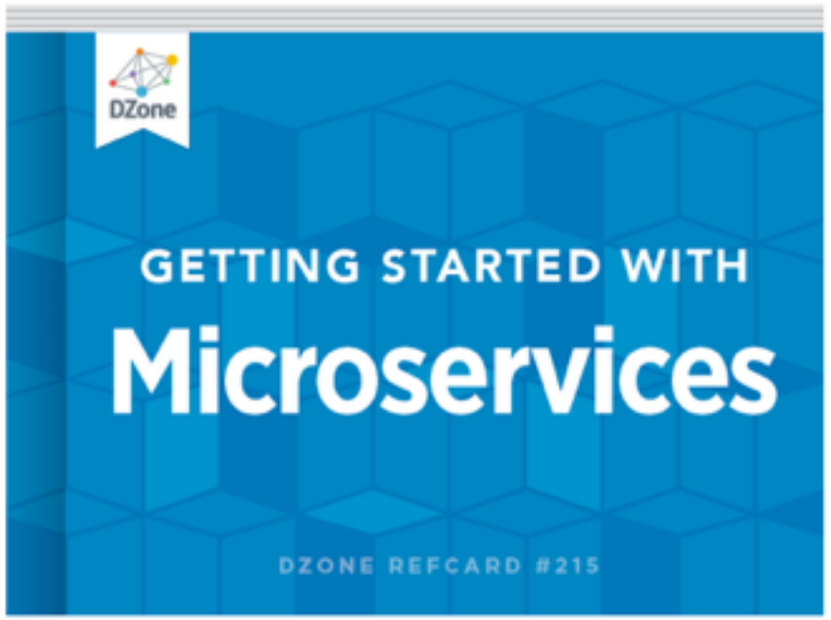


Shared Resources #5



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

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

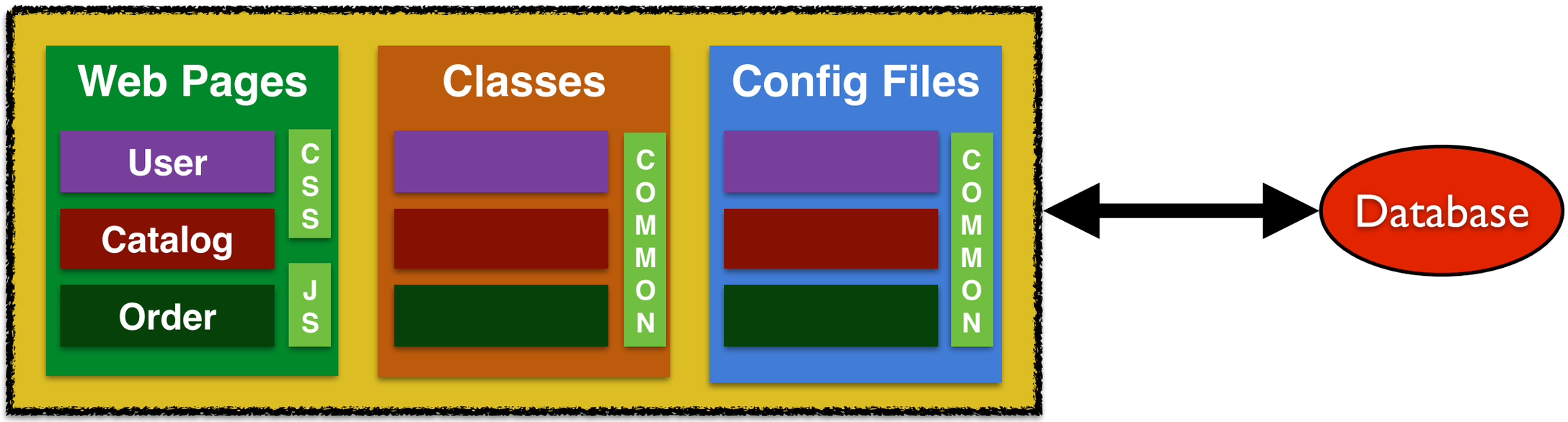
- Easier to develop, understand, maintain
- Starts faster than a monolith, speeds up deployments
- Local change can be easily deployed, great enabler of CD
- Each service can scale on X- and Z-axis
- Improves fault isolation
- Eliminates any long-term commitment to a technology stack
- Freedom of choice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

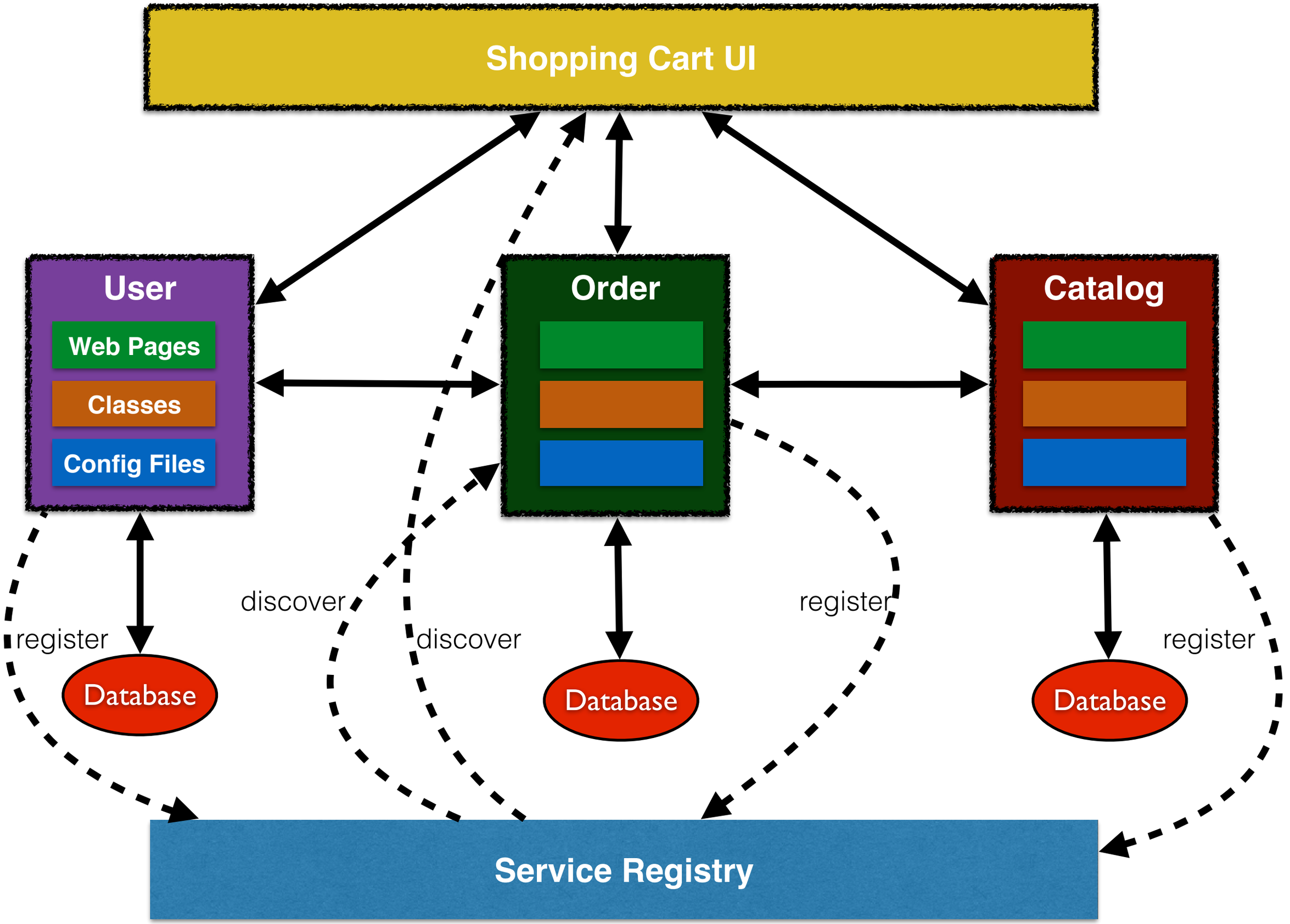
"If you can't build a [well-structured] monolith, what makes you think microservices are the answer?"

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


Web Pages
User
Catalog
Order
CSS
JS
Classes
COMMON
Config Files
COMMON
Database



https://github.com/arun-gupta/microservices/

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

http://blog.arungupta.me/monolithic-microservices-refactoring-javaee-applications/

Service Registry/Discovery

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

https://github.com/arun-gupta/microservices/blob/master/service-discovery.adoc

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

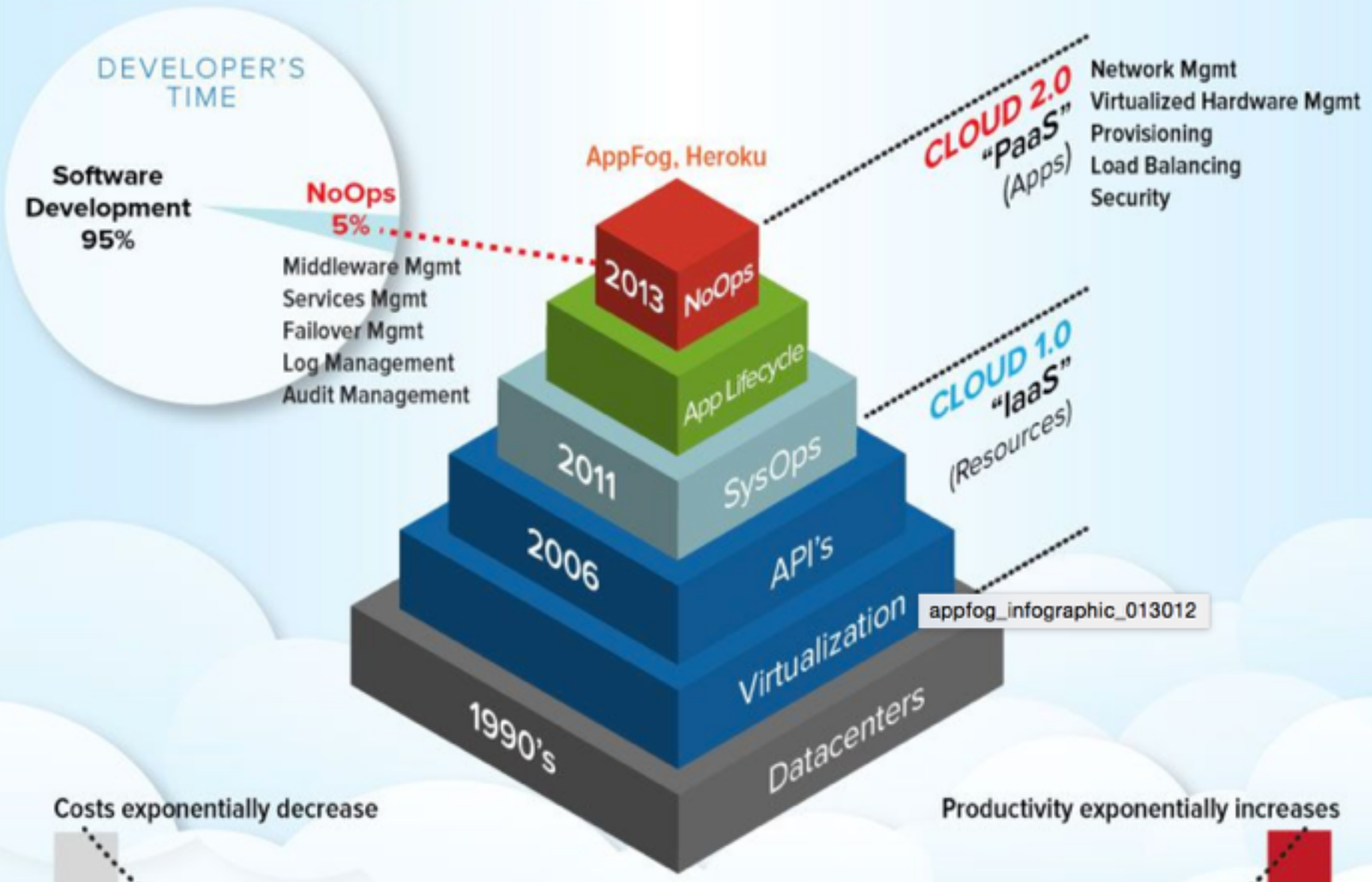
Containers not necessary for #microservices, #microservices does not mean you must use containers. Similarities, but can exist w/o other!

RETWEETS 20 LIKES 11

6:37 AM - 7 Nov 2015

2013: A bright NoOps future

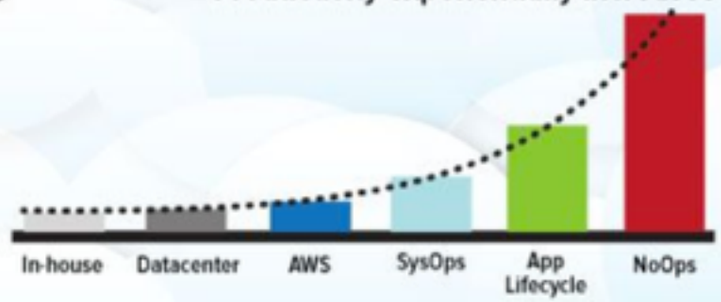
So where does this all lead? The end-game is NoOps. Where building and running an app is purely a developer process — and where developers are not having to spend time doing Ops work.



Costs exponentially decrease

Sources: National Venture Capital Association and Center for Venture Research

Productivity exponentially increases



Sources: National Venture Capital Association and Center for Venture Research

NoOps is the future. But it's a very near future. NoOps, typified by managed Platform-as-a-Service offerings, is the fully realized promise of the cloud. It has the most capex cost reduction of any cloud technology and combines it with the highest amount of productivity, velocity, and development agility.

NoOps takes all of the exponential gains from each layer of this Cloud 2.0 and passing them along to developers as one, rolled-up, massive reduction in cost, time and risk.

By freeing developers to focus on what they want to do — and on what they do best — we are enabling them to even more easily start companies and build products. And that is why AppFog is a *Start-up Enabler.*

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

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

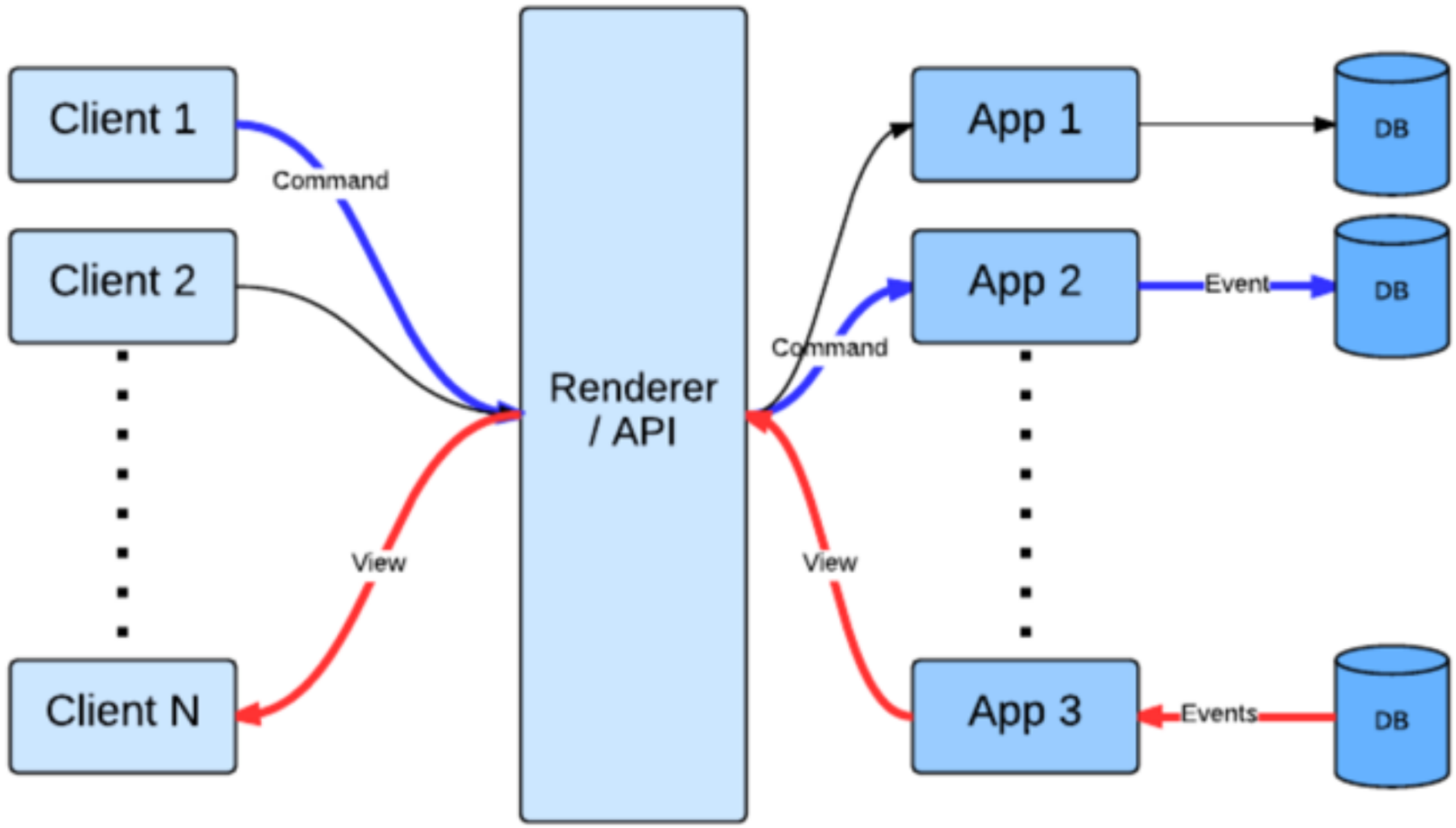
- Events are stored in a document format

- 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)

http://www.slideshare.net/aviranwix/road-to-continuous-delivery-wixcom

Couchbase

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

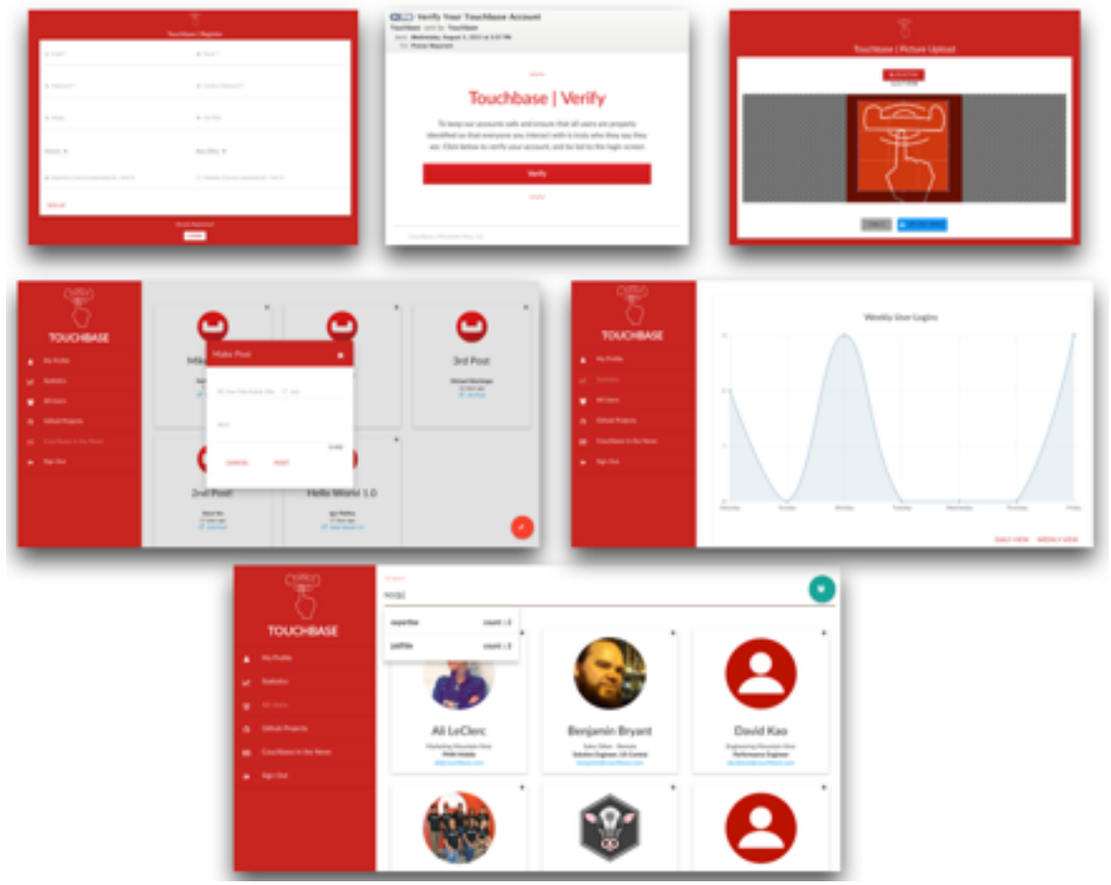
http://www.slideshare.net/aviranwix/scaling-wix-with-microservices-and-multi-cloud-2015

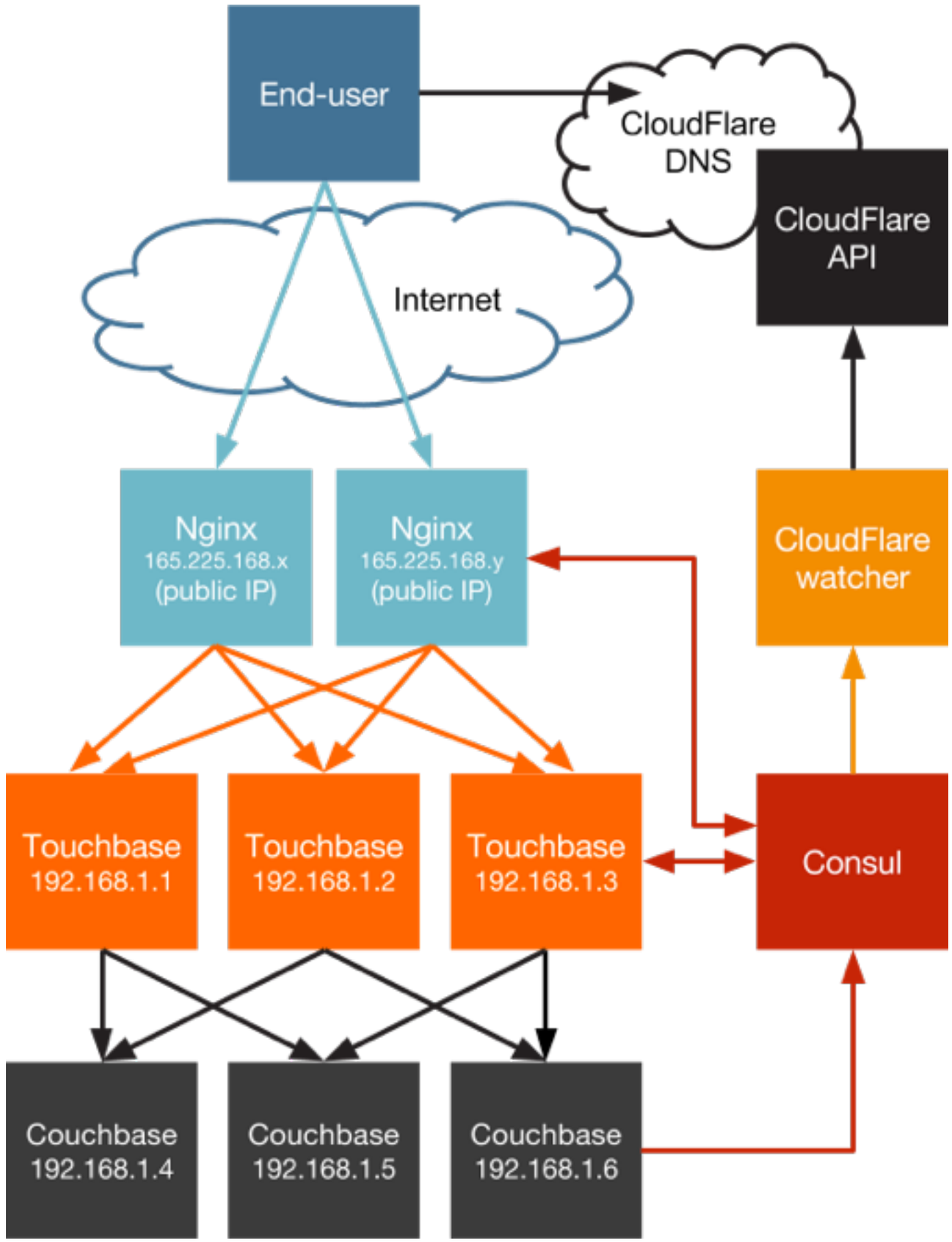
Couchbase

Microservices stack in seconds

- Touchbase: Social network platform built using CEAN

- https://github.com/couchbaselabs/touchbase/





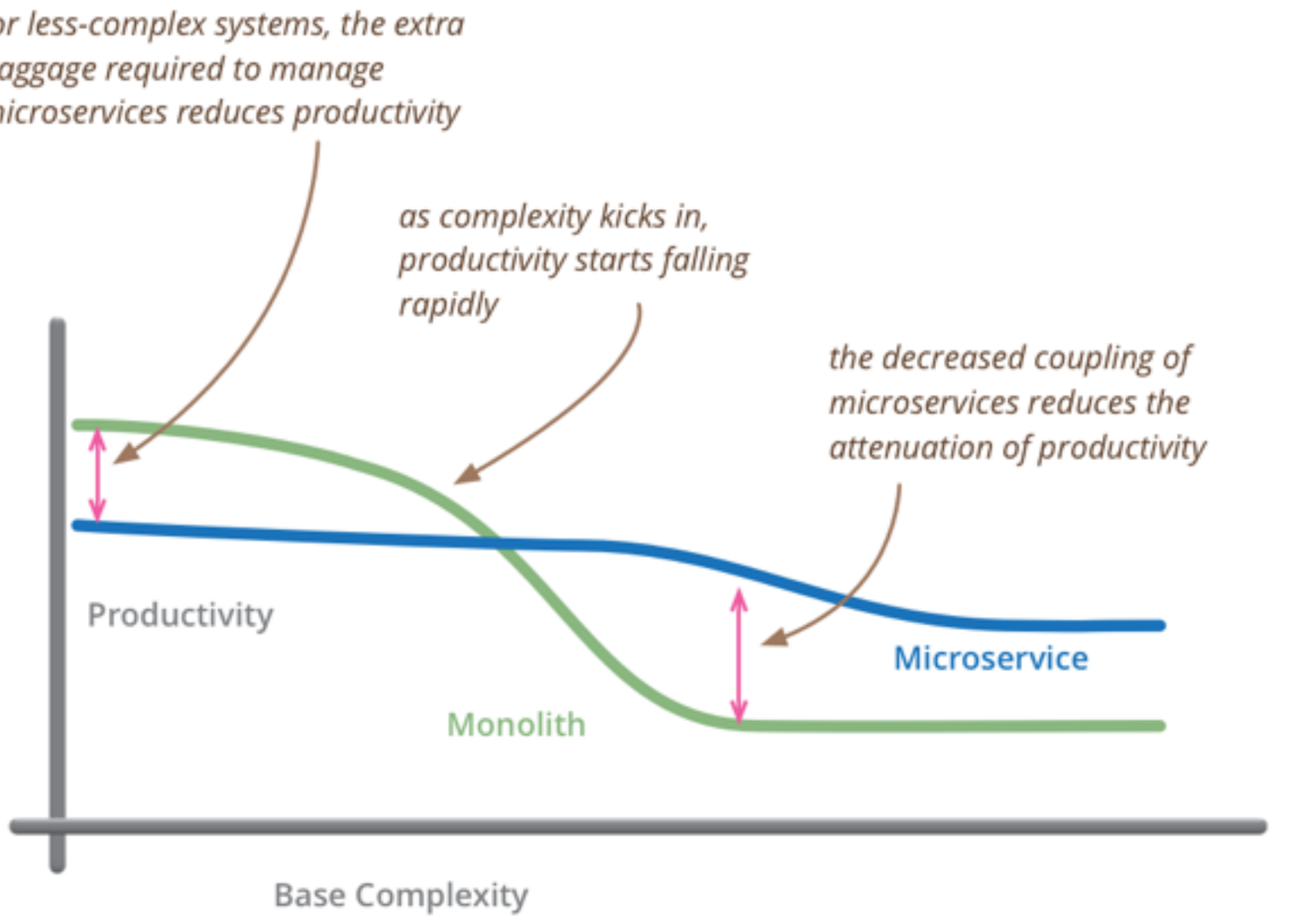
https://www.joyent.com/blog/how-to-dockerize-a-complete-application

```
# Consul as a service discovery tier
consul:
    image: progrium/consul:latest
    command: -server -bootstrap -ui-dir /ui
    restart: always
    mem_limit: 128m
    ports:
        - 53
        - 8300
        - 8301
        - 8302
        - 8400
        - 8500
    dns:
       - 127.0.0.1

# Manually bootstrap the first instance, then...
# Scale this tier and each additional container/instance wi
# self-configure as a member of the cluster
couchbase:
    image: misterbisson/triton-couchbase:enterprise-4.0.0-r
    restart: always
    links:
        - consul
    mem_limit: 4g
    ports:
        - 8091
        - 8092
        - 8093
        - 11207
        - 11210
        - 11211
        - 18091
        - 18092
    environment:
        - CONSUL_HOST=http://consul:8500
        - COUCHBASE_SERVICE_NAME=couchbase
        - COUCHBASE_USER=Administrator
        - COUCHBASE_PASS=password


# the main application
touchbase:
    image: 0x74696d/triton-touchbase
    links:
    - consul
    mem_limit: 1g
    ports:
    - 3000
    restart: always
    command: >
      /opt/containerbuddy/containerbuddy
      -config file:///opt/containerbuddy/touchbase.json
      /usr/local/bin/run-touchbase.sh

# Nginx as a load-balancing tier and reverse proxy
nginx:
    image: 0x74696d/triton-touchbase-demo-nginx
    mem_limit: 512m
    ports:
    - 80
    links:
    - consul:consul
    restart: always
    environment:
    - CONTAINERBUDDY=file:///opt/containerbuddy/nginx.json
    command: >
      /opt/containerbuddy/containerbuddy
      nginx -g "daemon off;"

# Support dynamic DNS for the load balancing tier
# https://www.joyent.com/blog/automatic-dns-updates-with-containerbuddy
cloudflare:
    image: 0x74696d/triton-cloudflare
    mem_limit: 128m
    links:
    - consul:consul
    restart: always
    env_file: .env
    command: >
      /opt/containerbuddy/containerbuddy
      -config file:///opt/containerbuddy/cloudflare.json
```

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



"don't even consider microservices unless you have a system that's too complex to manage as a monolith"

http://martinfowler.com/bliki/MicroservicePremium.html

Couchbase

References

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