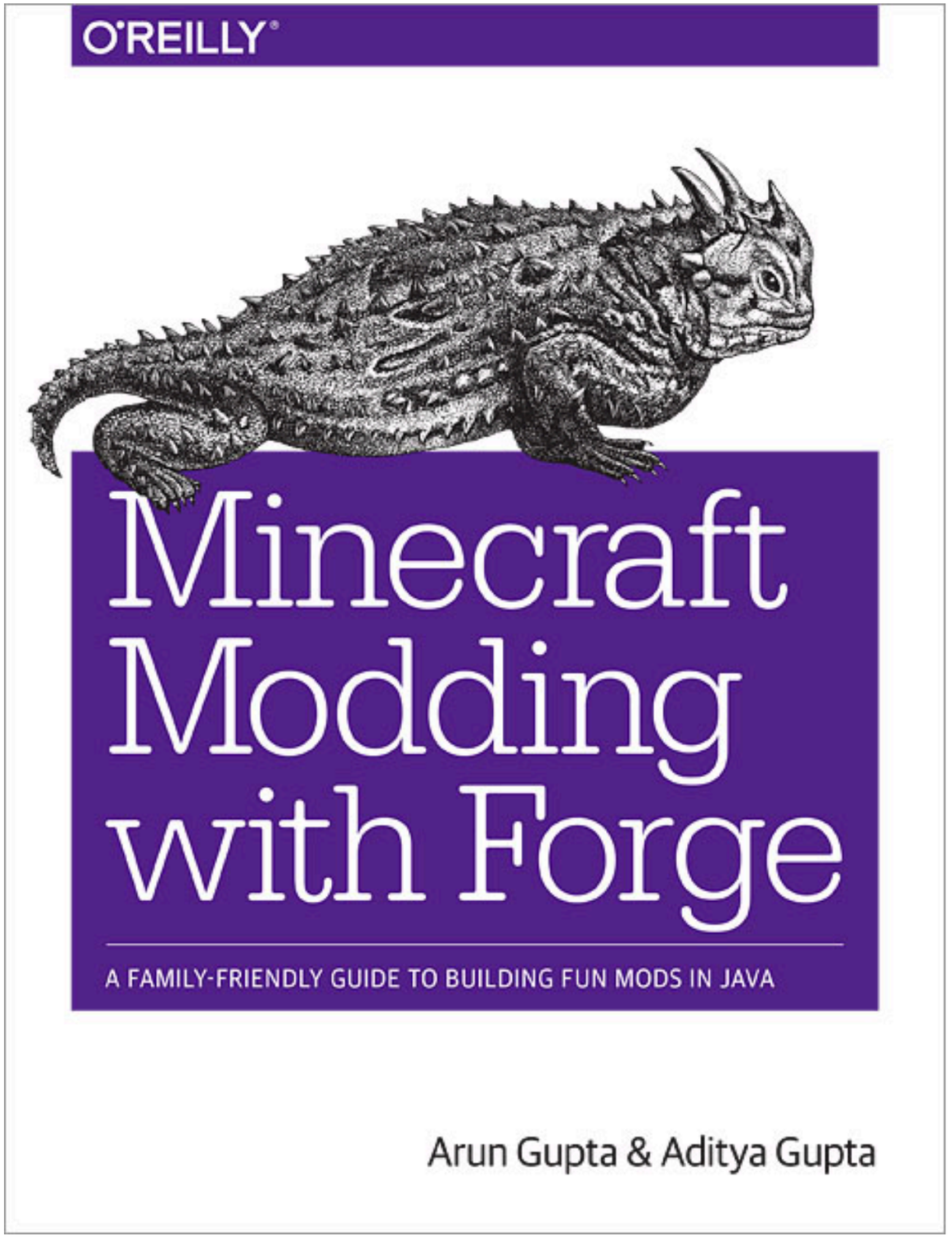
Refactoring your Java EE applications using Microservices and Containers

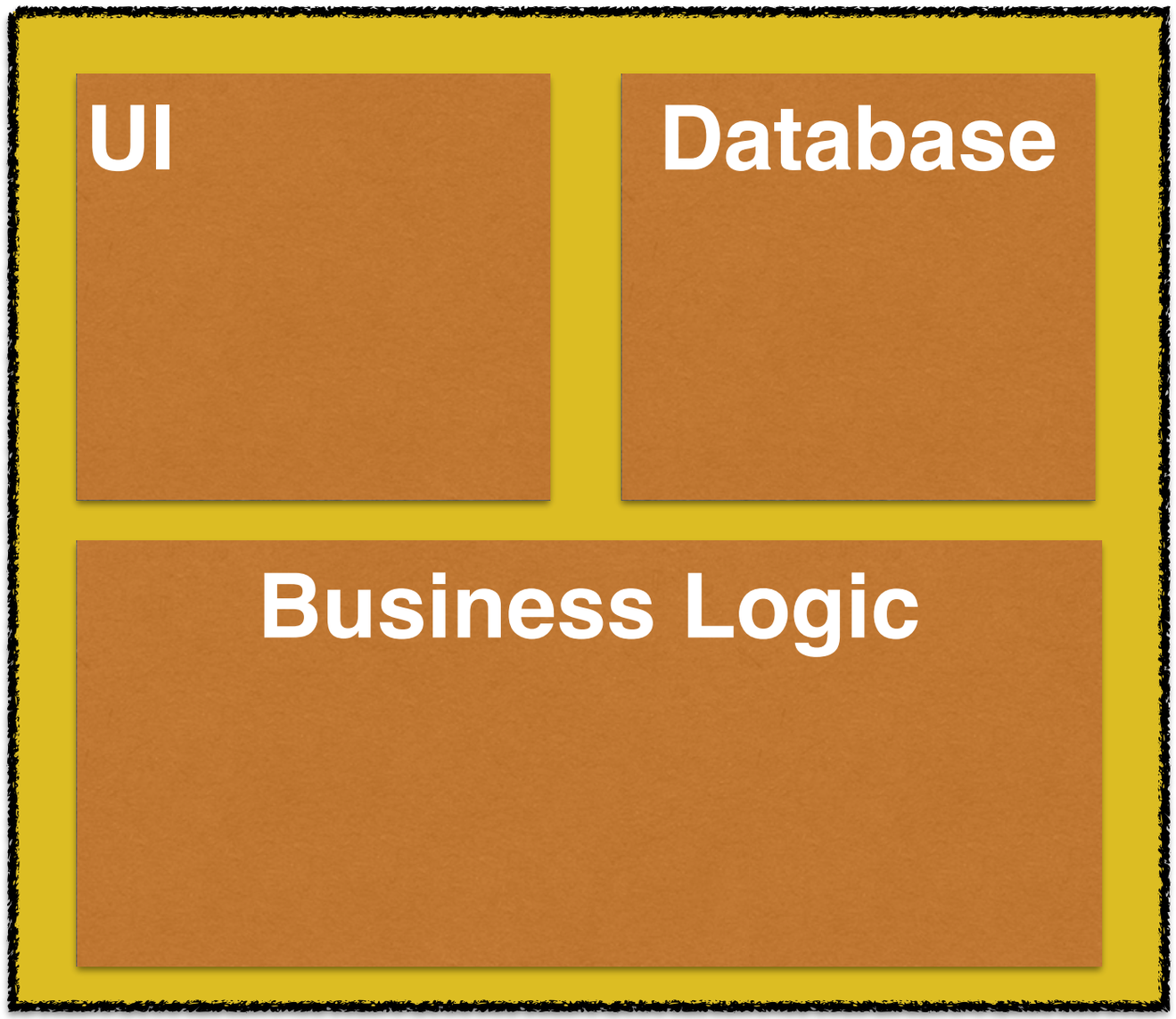
Arun Gupta
Vice President, Developer Advocacy, Couchbase
@arungupta, blog.arungupta.me



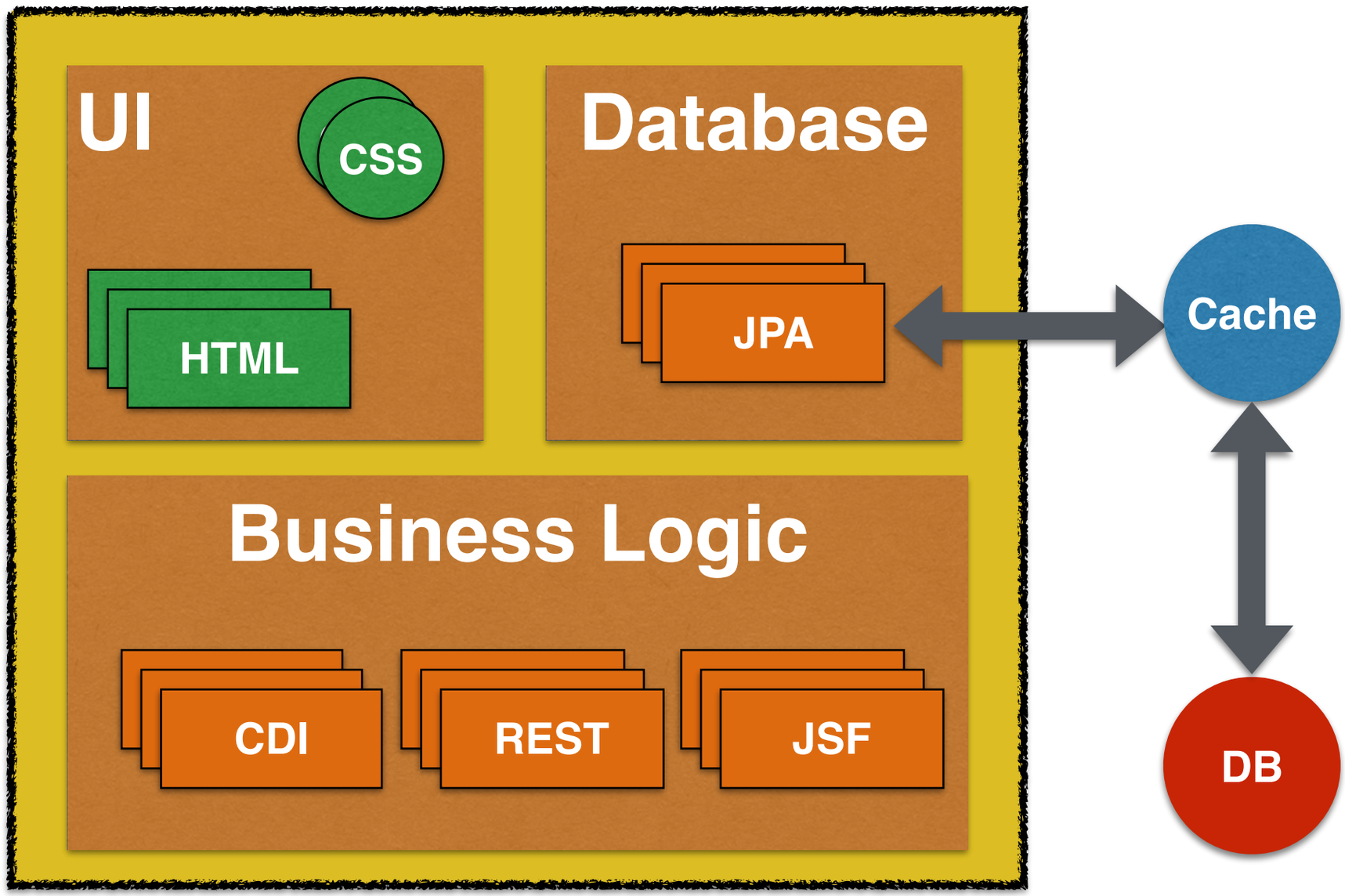
DEVOXX™ 4KIDS USA

Couchbase

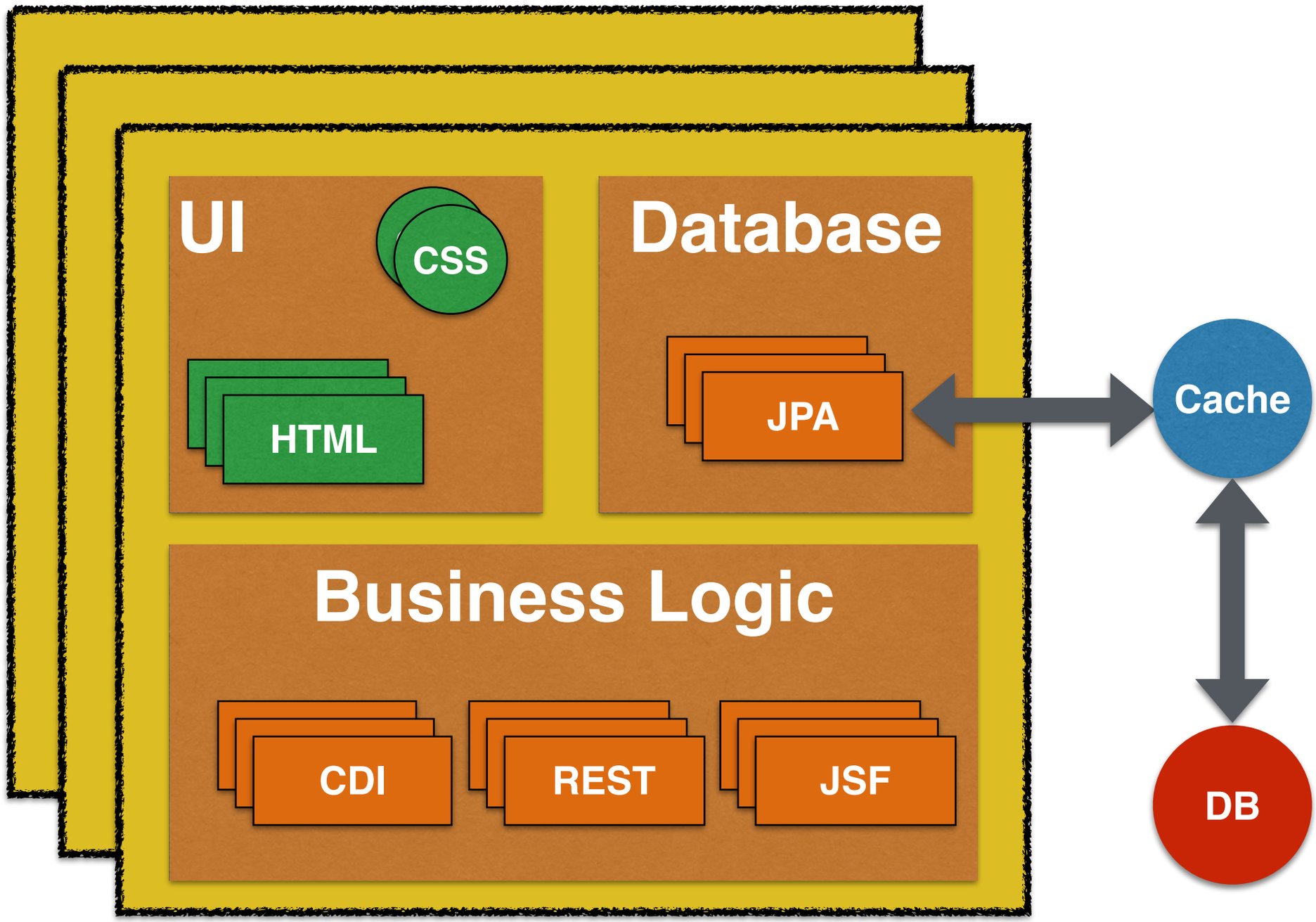
Monolith Application



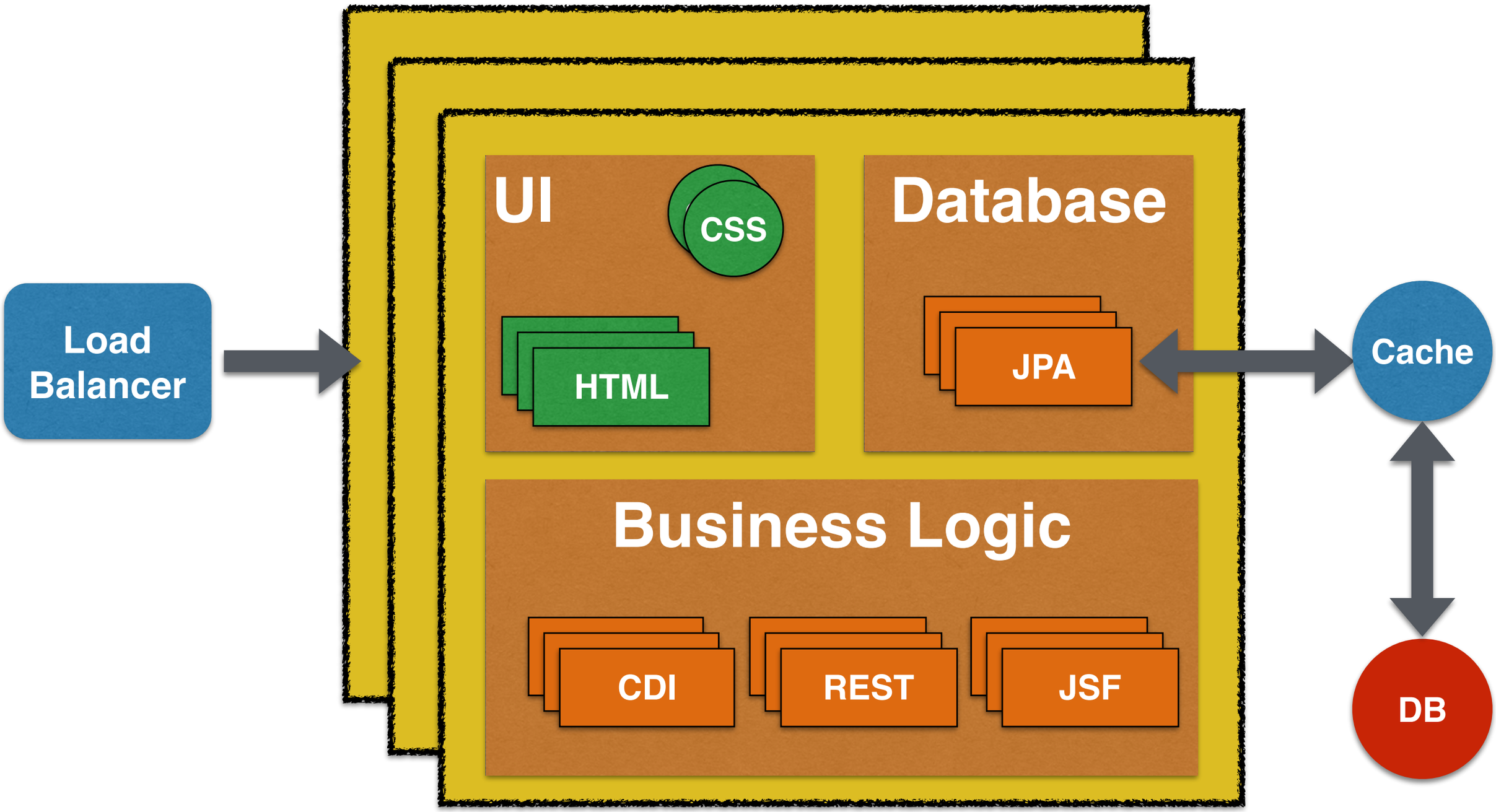
Monolith Application



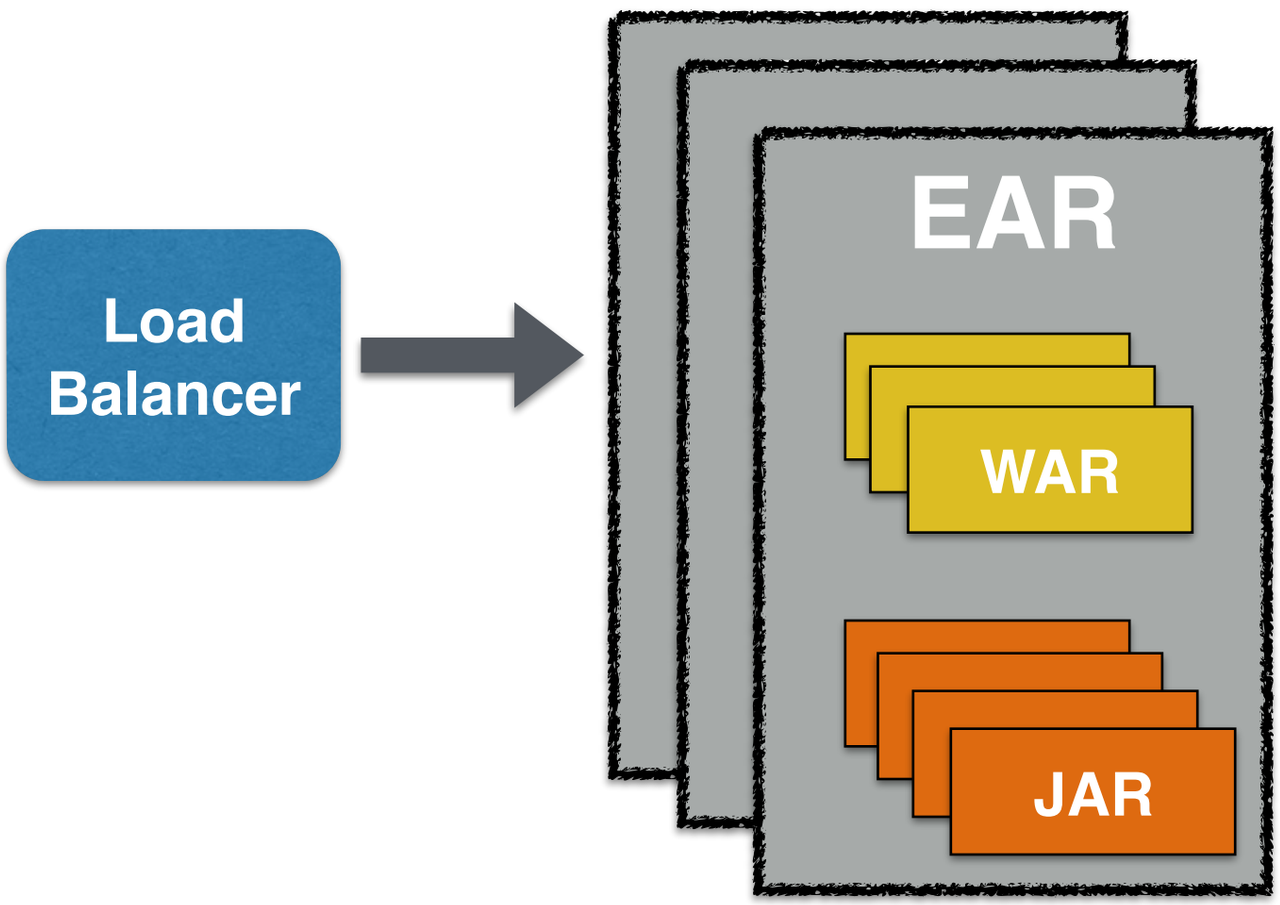
Monolith Application



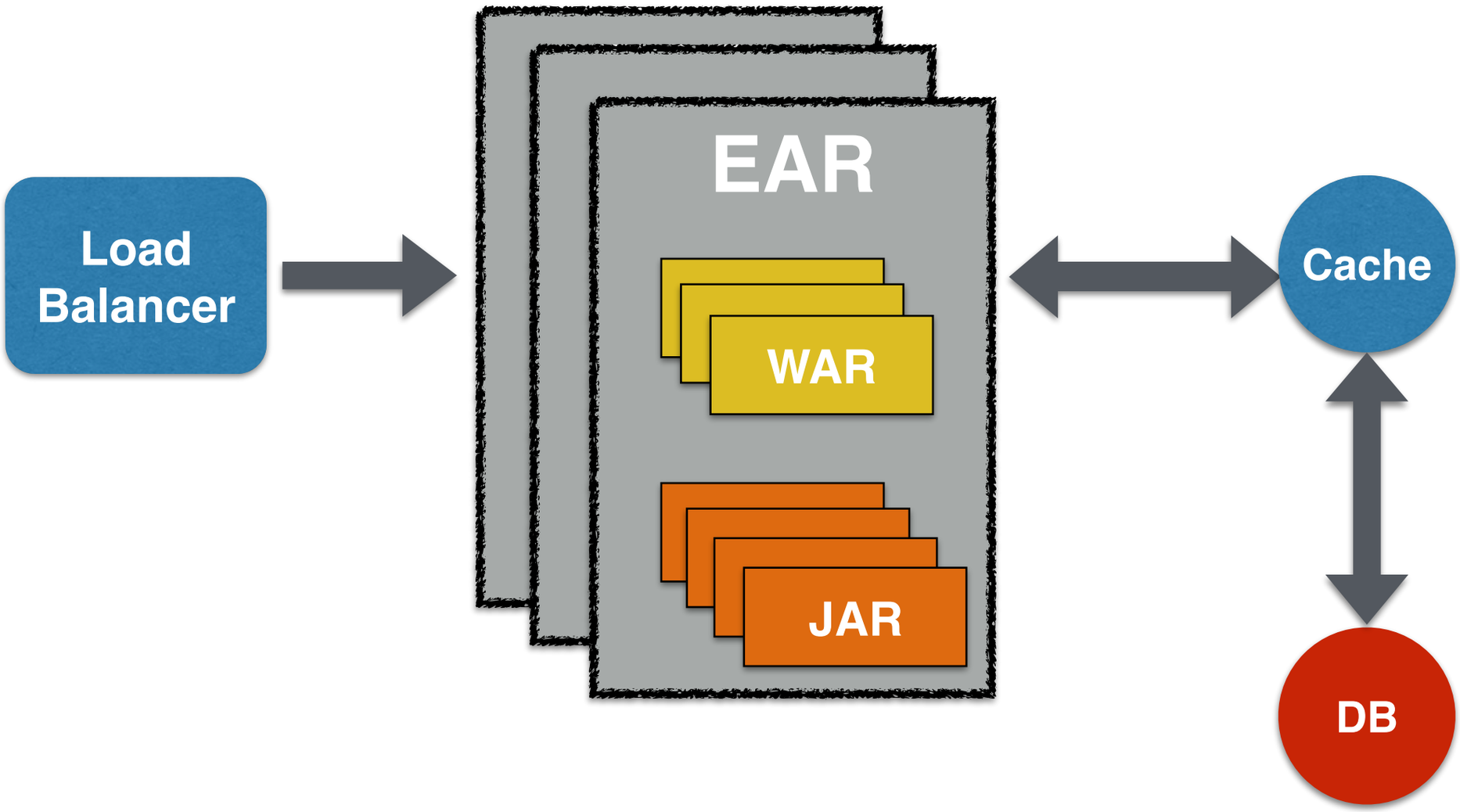
Monolith Application



Monolith Application



Monolith Application



Advantages of Monolith Application

Advantages of Monolith Application

- Typically packaged in a single `.ear`

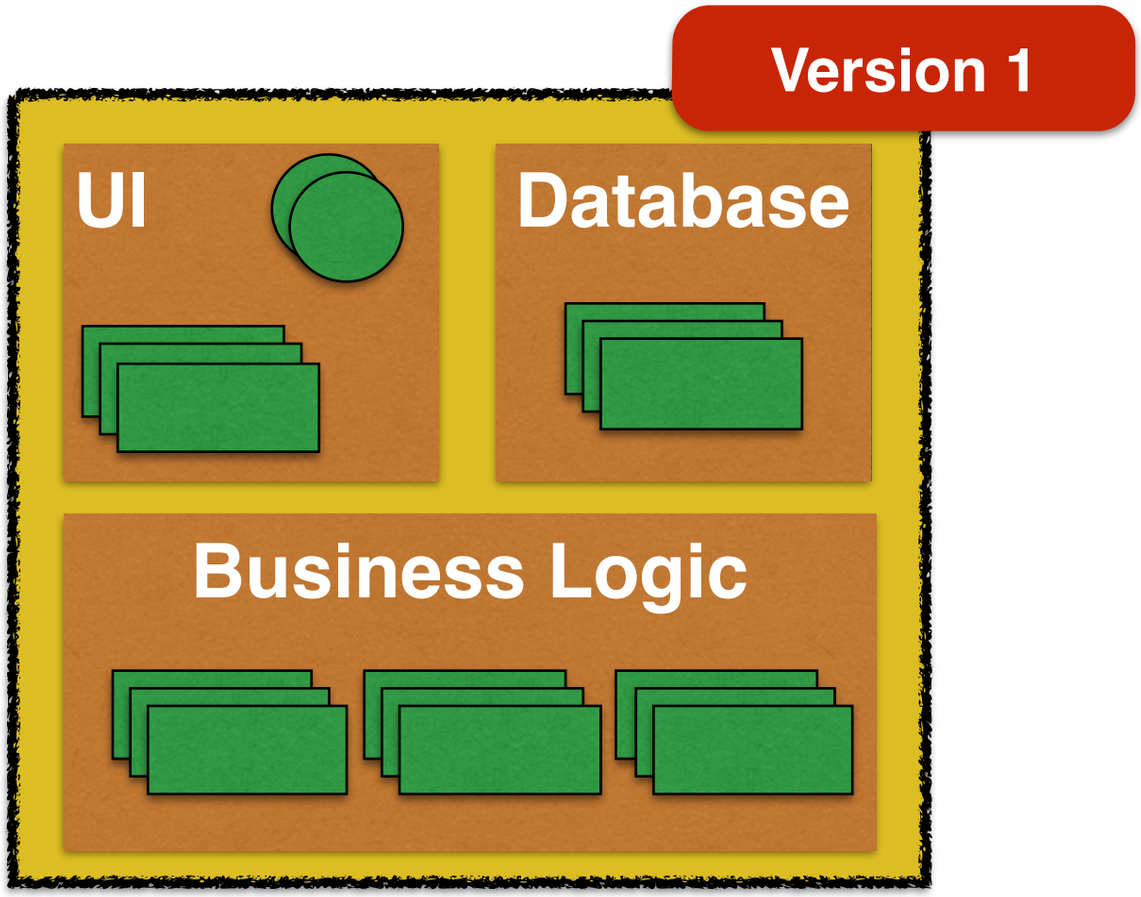
Advantages of Monolith Application

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

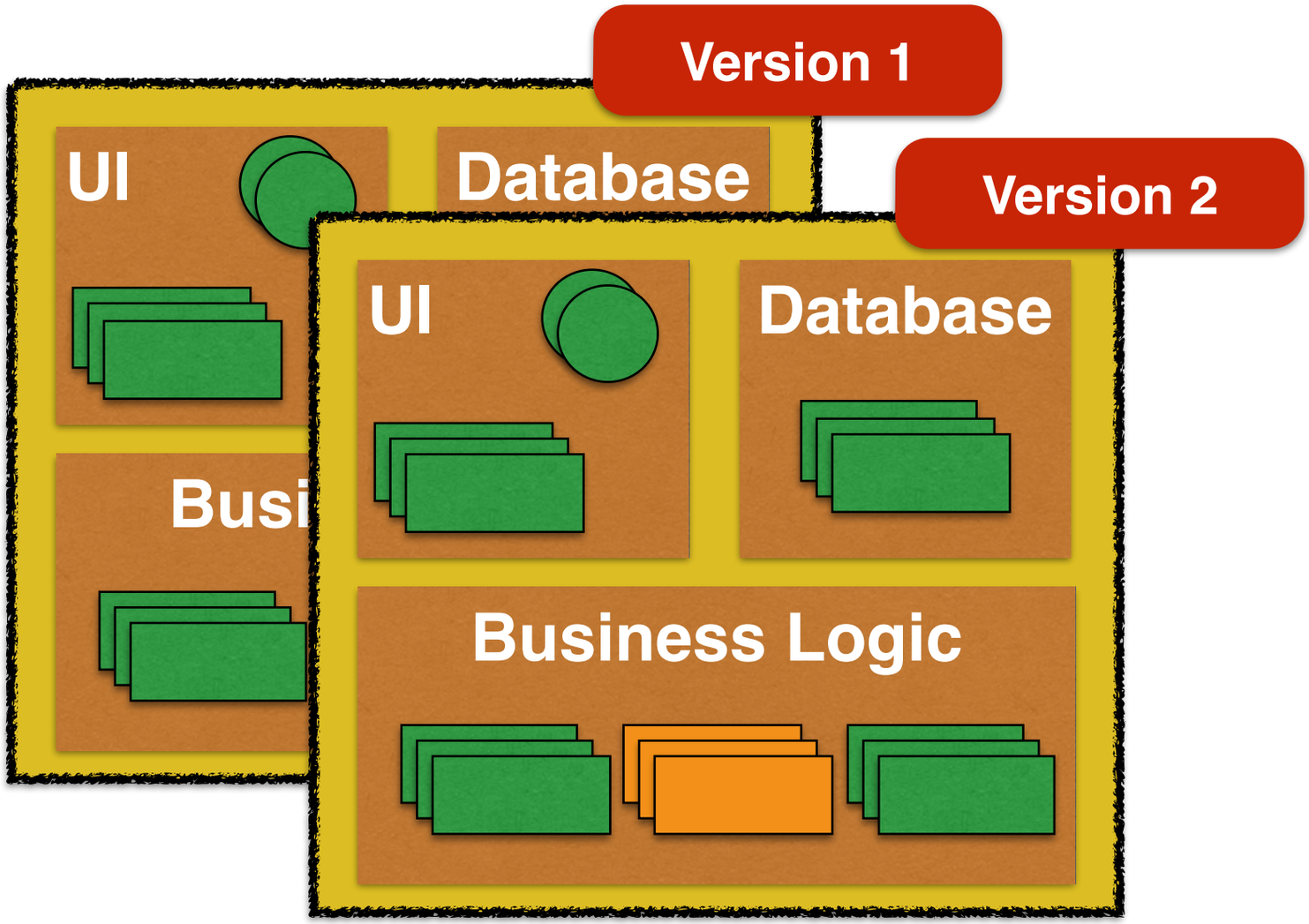
Advantages of Monolith Application

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

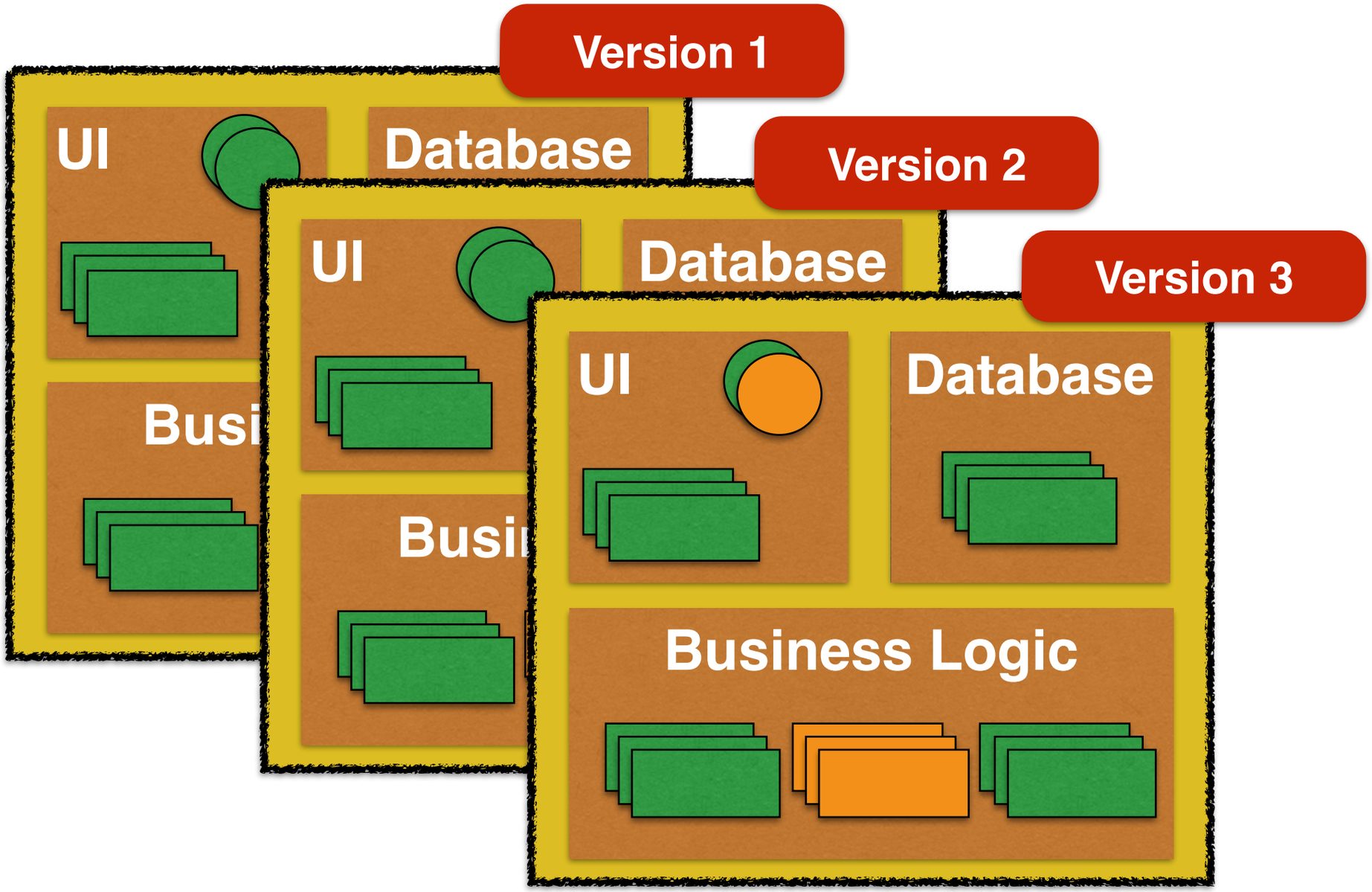
Monolith Application



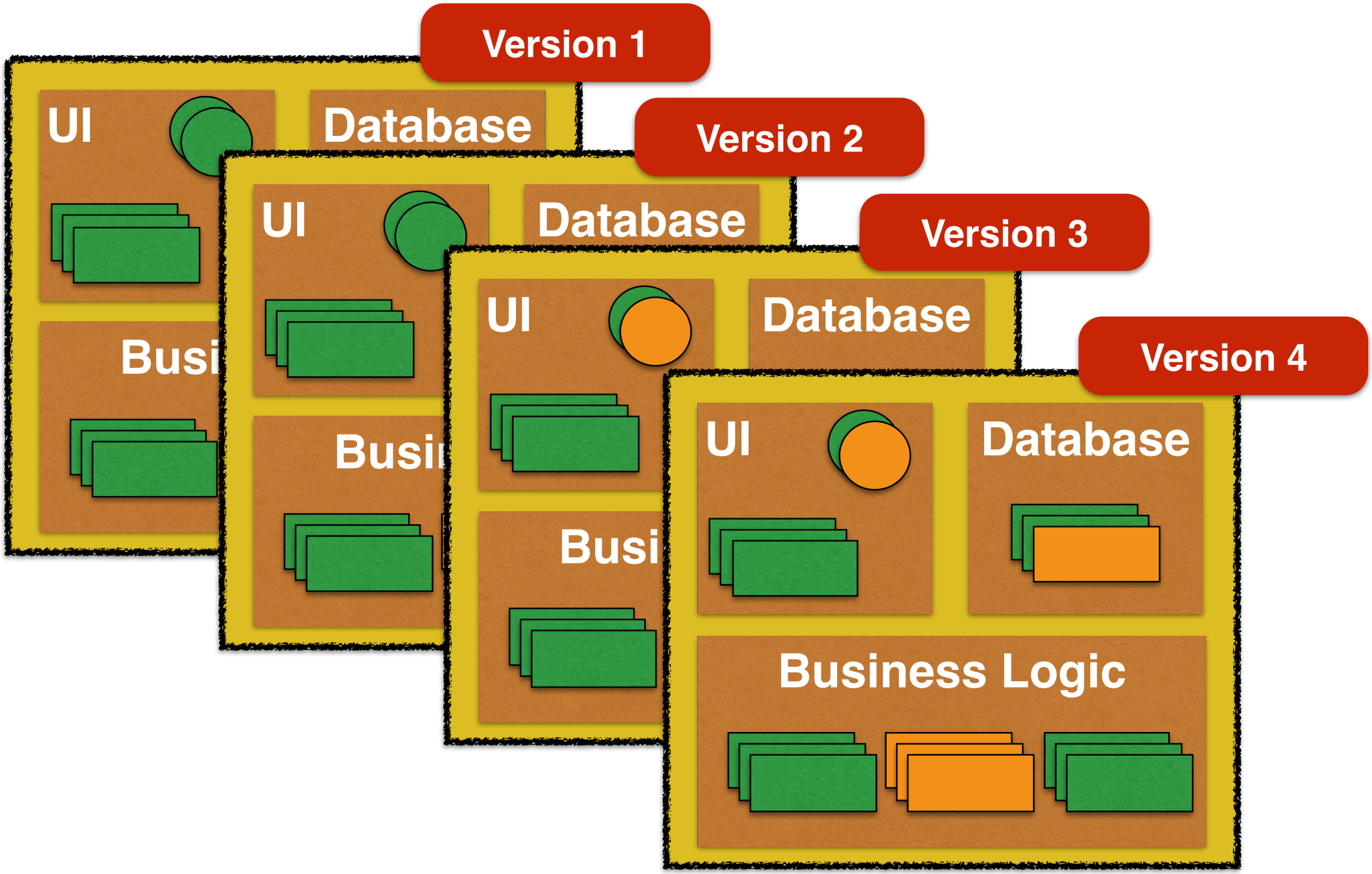
Monolith Application



Monolith Application



Monolith Application



Couchbase

Disadvantages of Monolith Application

Disadvantages of Monolith Application

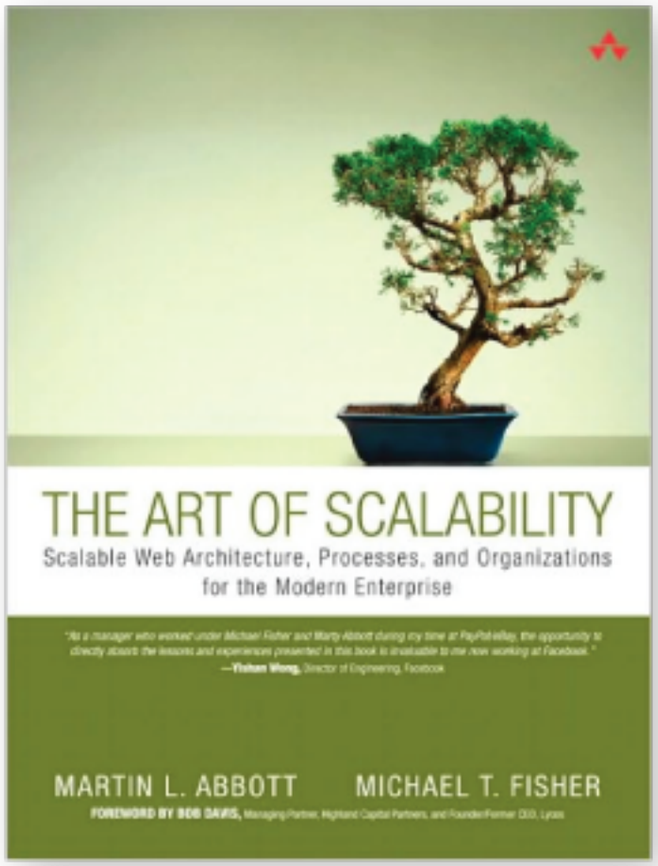
- Difficult to deploy and maintain

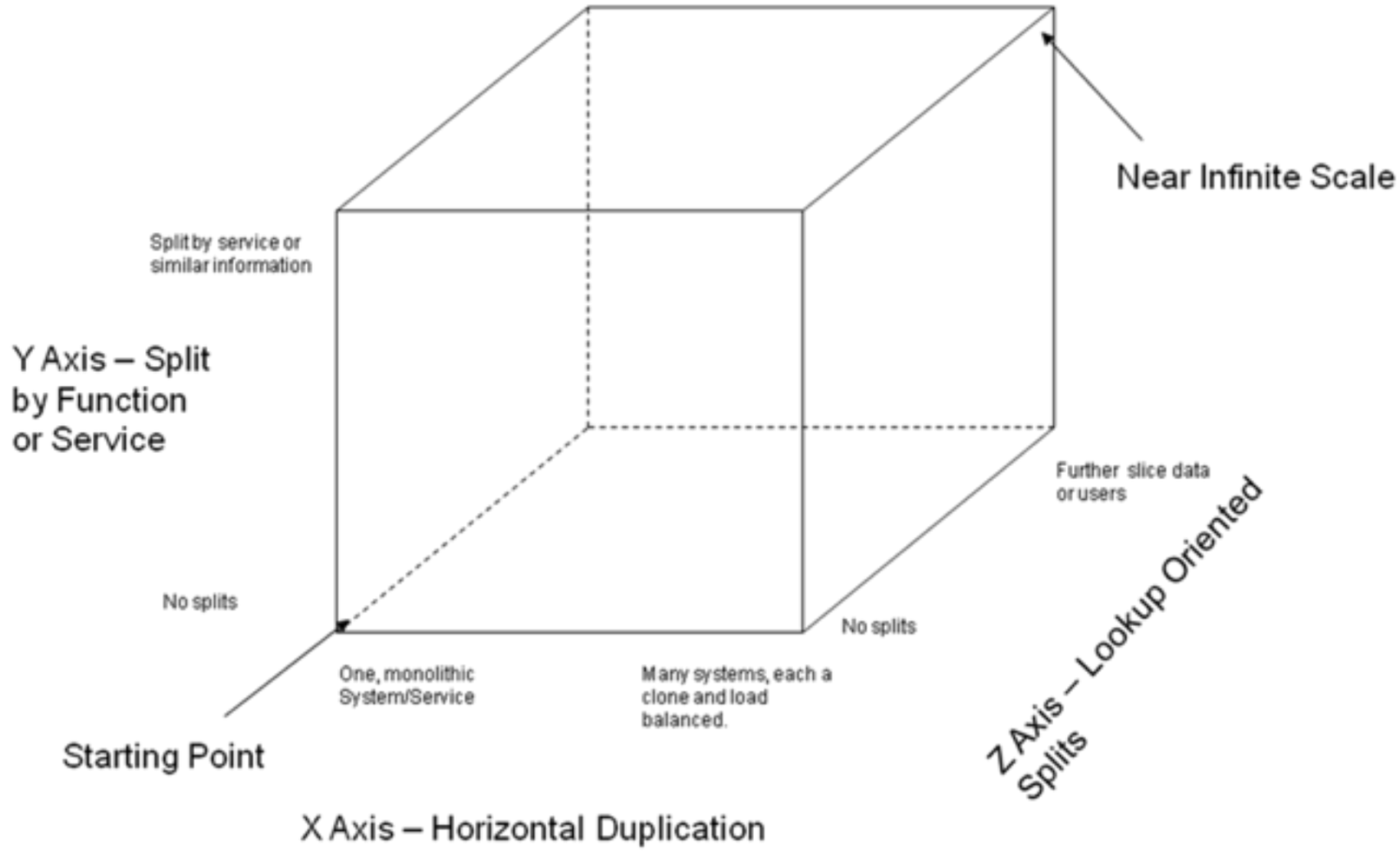
Disadvantages of Monolith Application

- Difficult to deploy and maintain
- Obstacle to frequent deployments

Disadvantages of Monolith Application

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

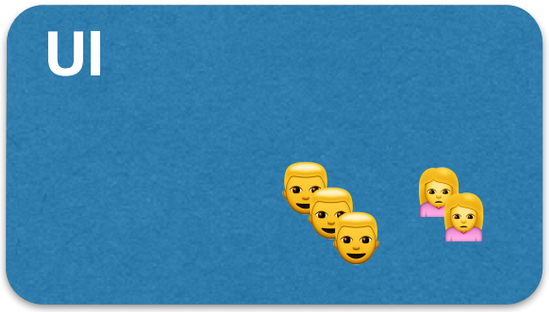




http://akfpartners.com/techblog/2008/05/08/splitting-applications-or-services-for-scale/

*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***

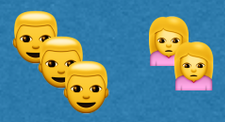

I DONT ALWAYS
BUILD EVERYTHING
BUT WHEN I DO I
USE MICRO SERVICES
imgflip.com


UI


Business Logic

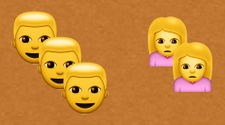

Persistence

UI



Business Logic

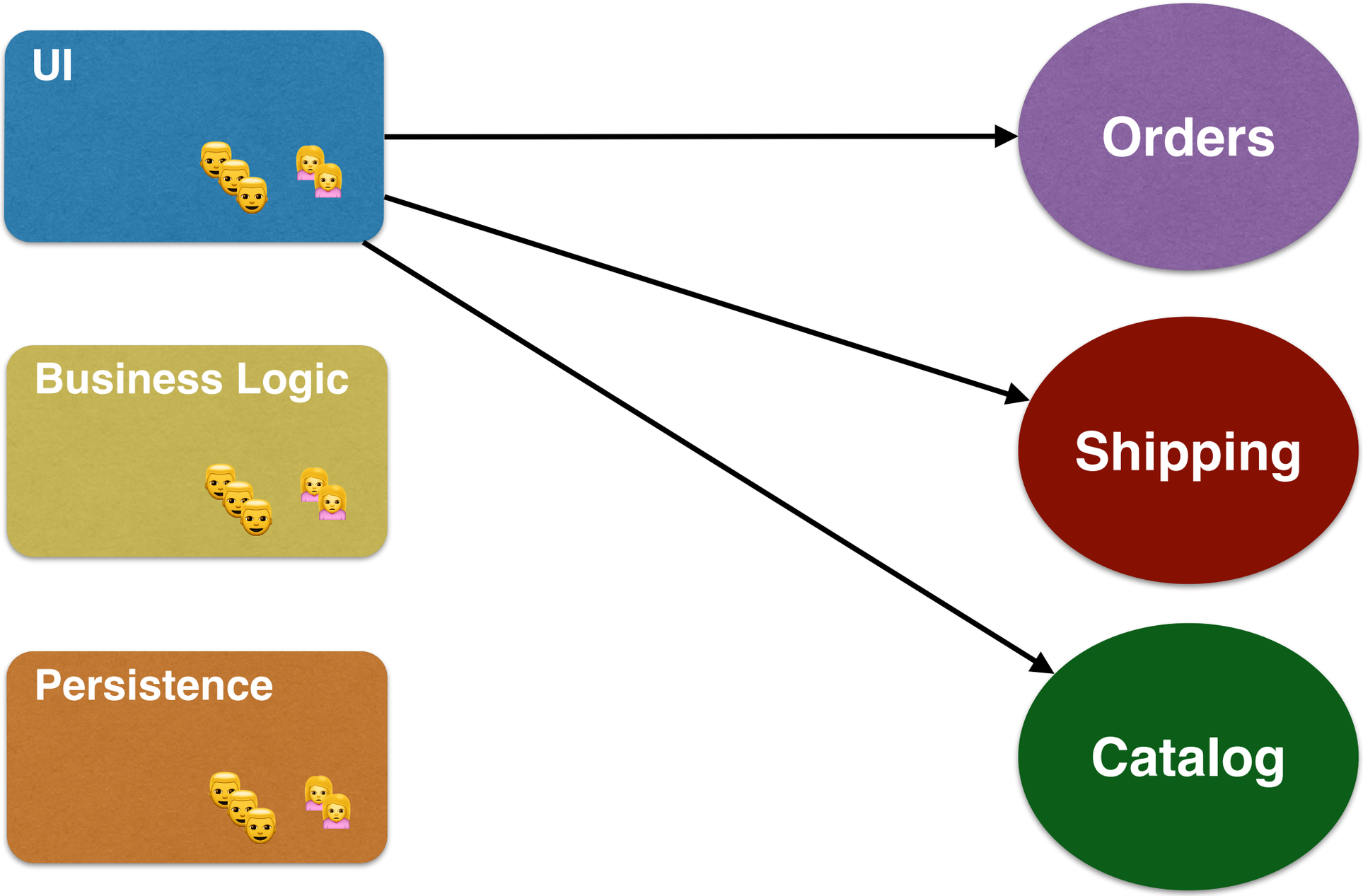
Persistence

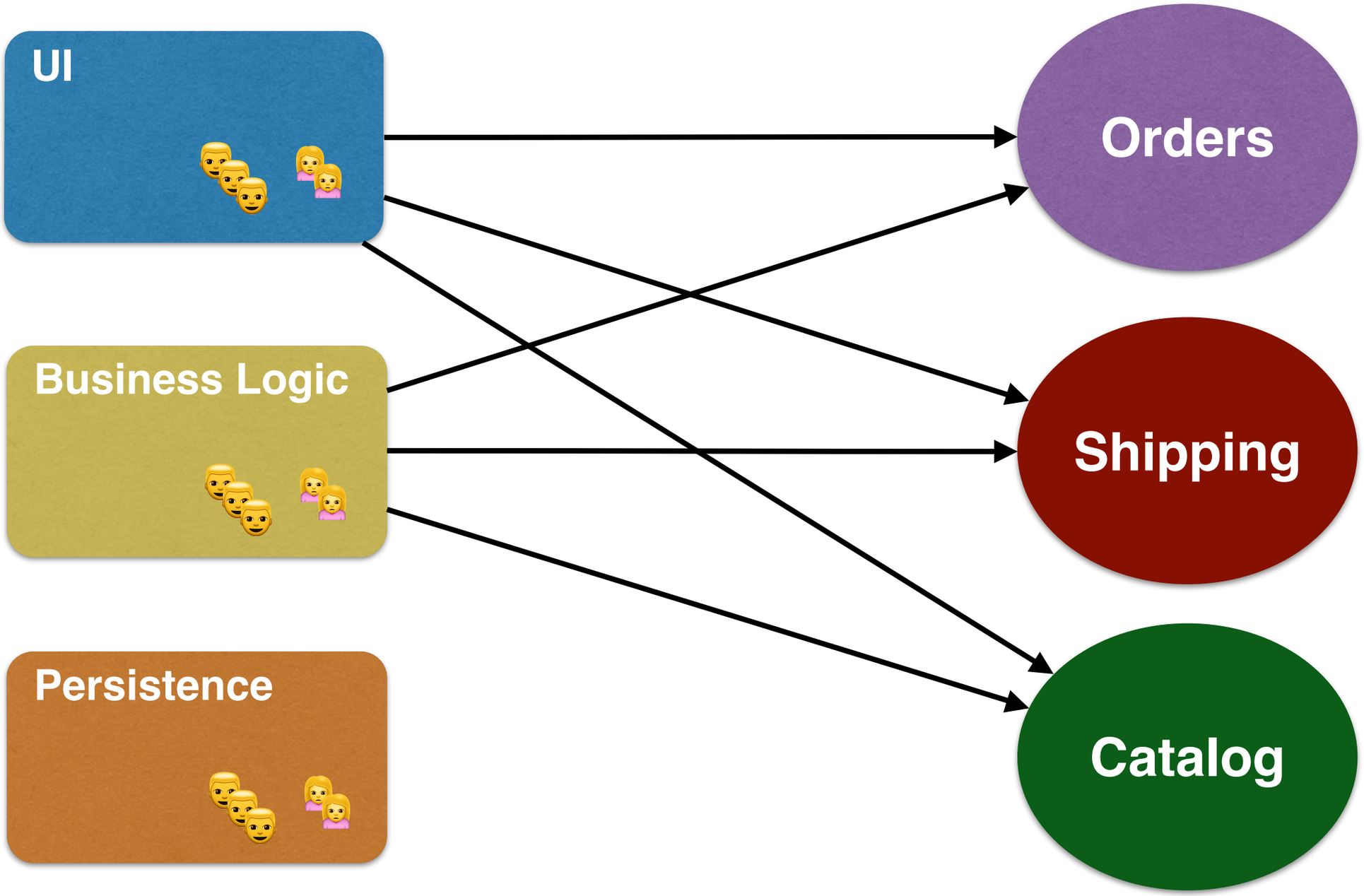


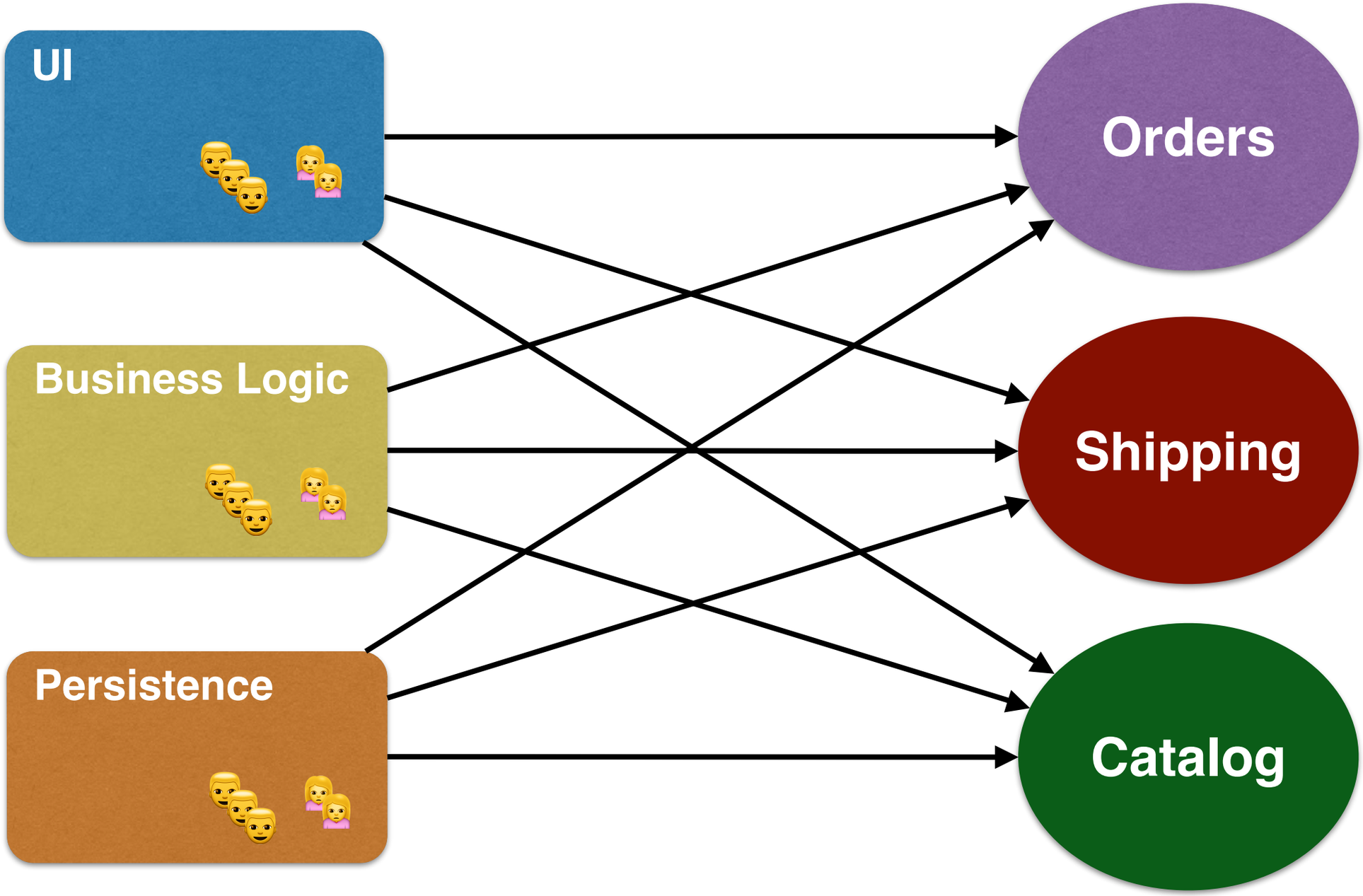
Orders

Shipping

Catalog


UI
Business Logic
Persistence
Orders
Shipping
Catalog


UI
Business Logic
Persistence
Orders
Shipping
Catalog


UI
Business Logic
Persistence
Orders
Shipping
Catalog

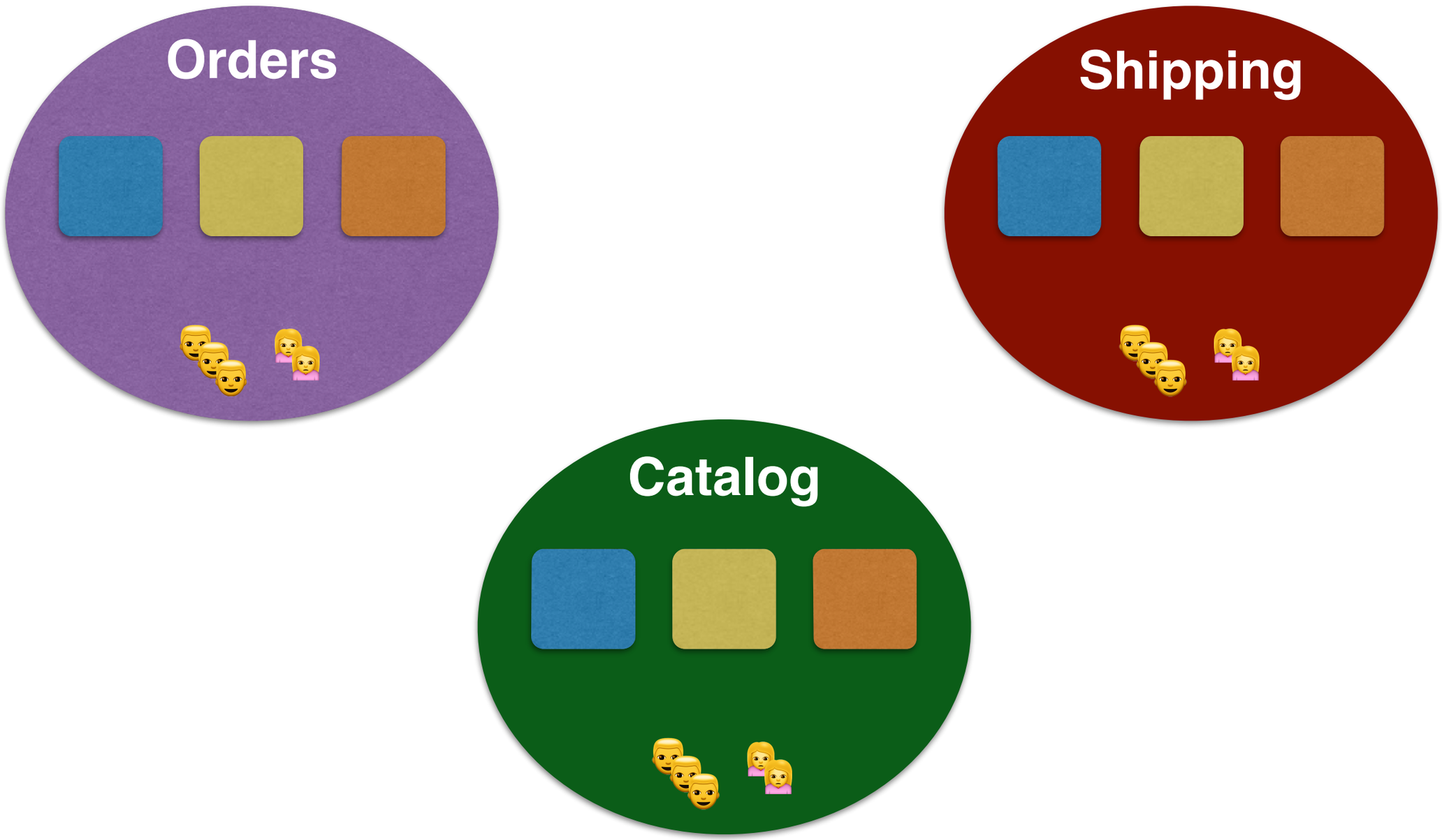
*"Any **organization that designs a system** (defined more broadly here than just information systems) will inevitably produce a design whose structure is a **copy of the organization's communication structure**."*

–Melvin Conway

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

Characteristics

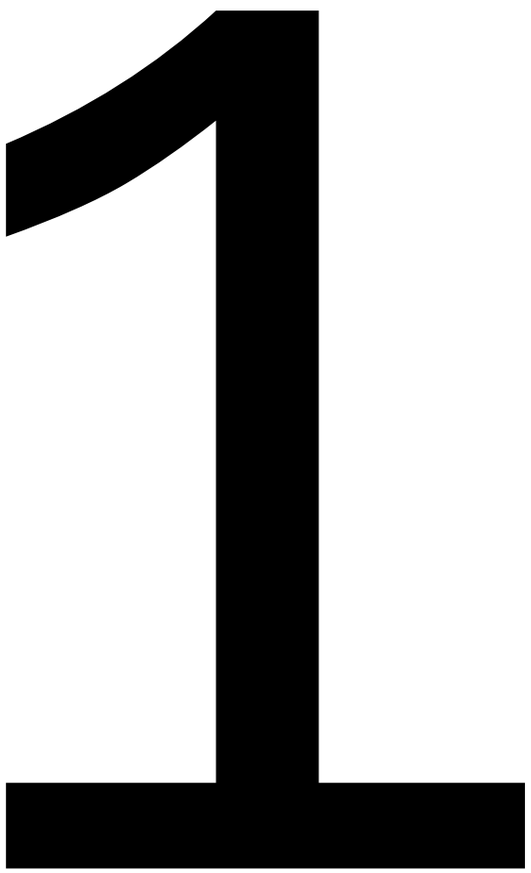
Teams around business capability



Characteristics

Single Responsibility Principle

DO



THING

Characteristics

Explicitly Published interface

Characteristics

Independently replaceable and upgradeable

Characteristics

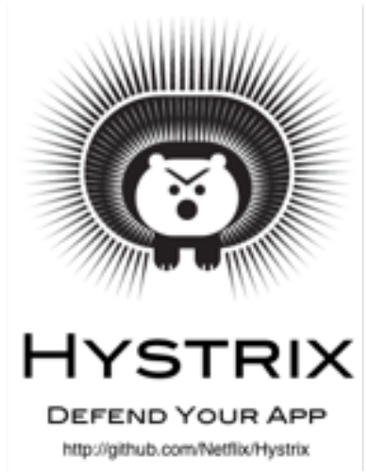
Characteristics

With great power, comes great responsibility

“you build it, you run it!”

Characteristics

Designed for failure



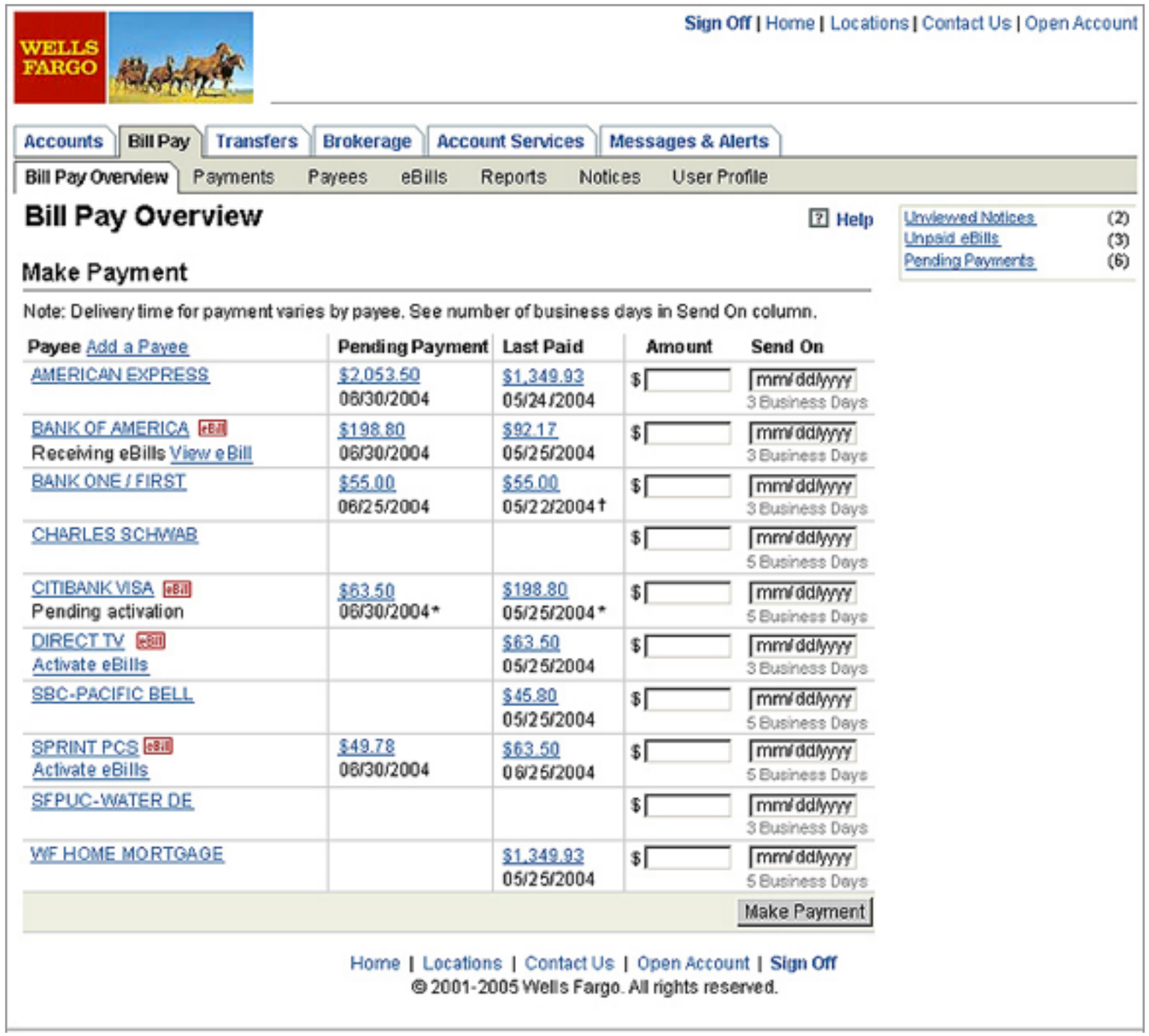
Fault tolerance is a requirement, not a feature

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

Characteristics

Characteristics

100% automated



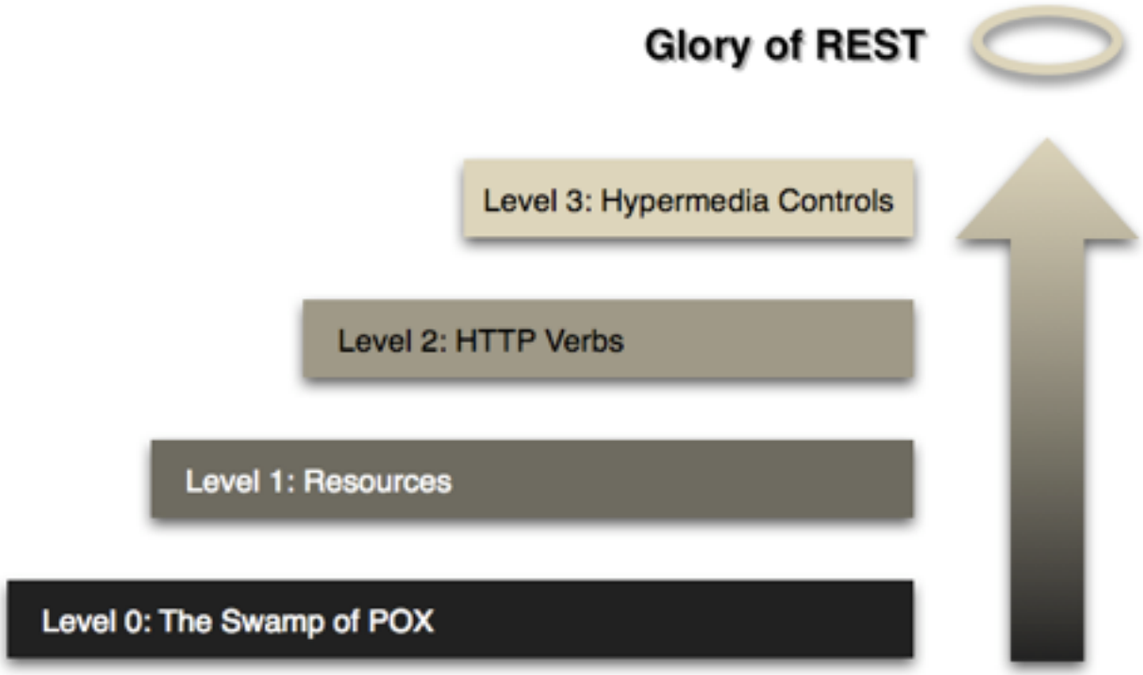
Couchbase

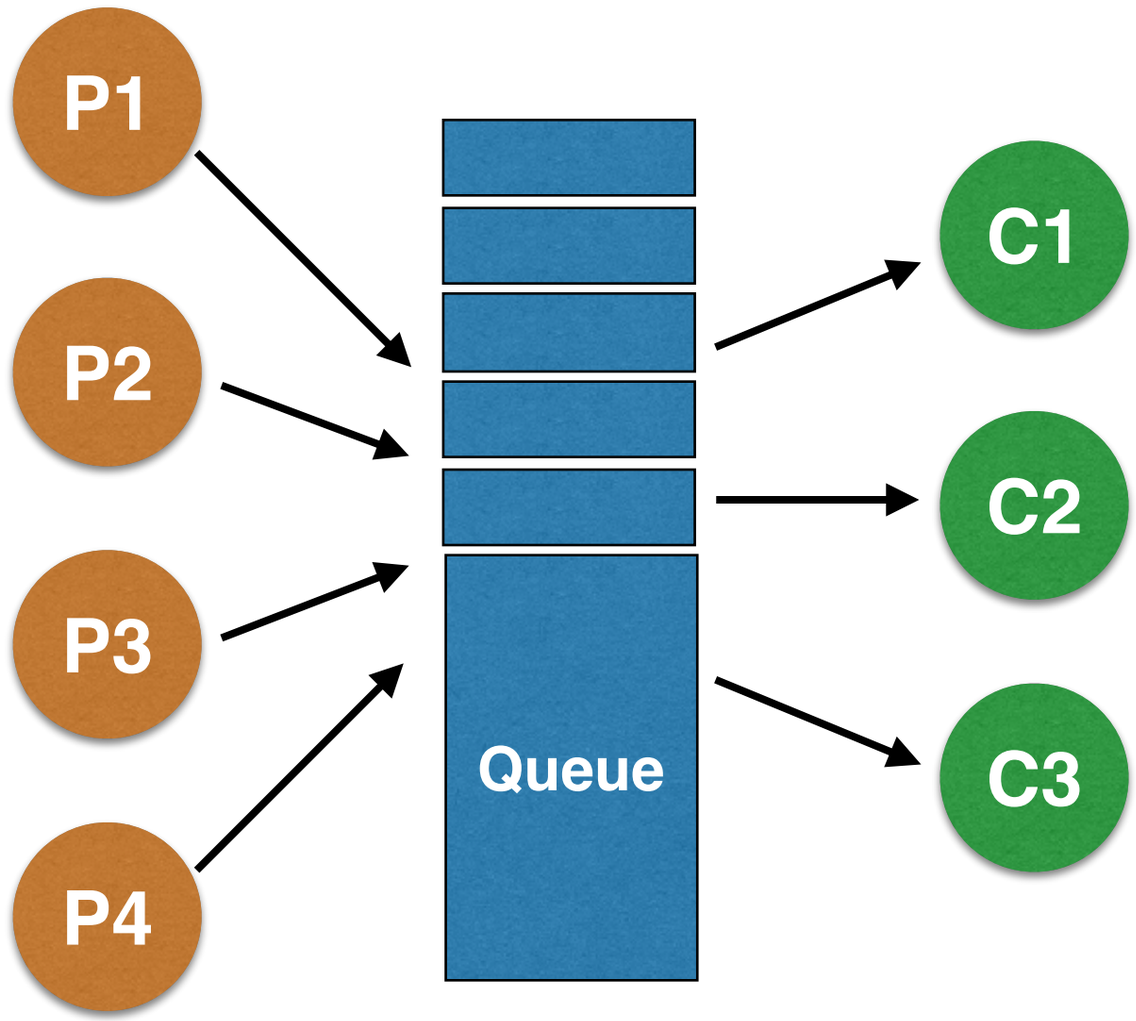
Characteristics

Sync or Async Messaging

Characteristics

REST vs Pub/Sub





Characteristics

“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

SOA 2.0?

- Conway's Law

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

SOA 2.0?

- Conway's Law
- Service Discovery

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

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

Strategies for decomposing

Strategies for decomposing

Strategies for decomposing

- Verb or usecase - e.g. Checkout UI

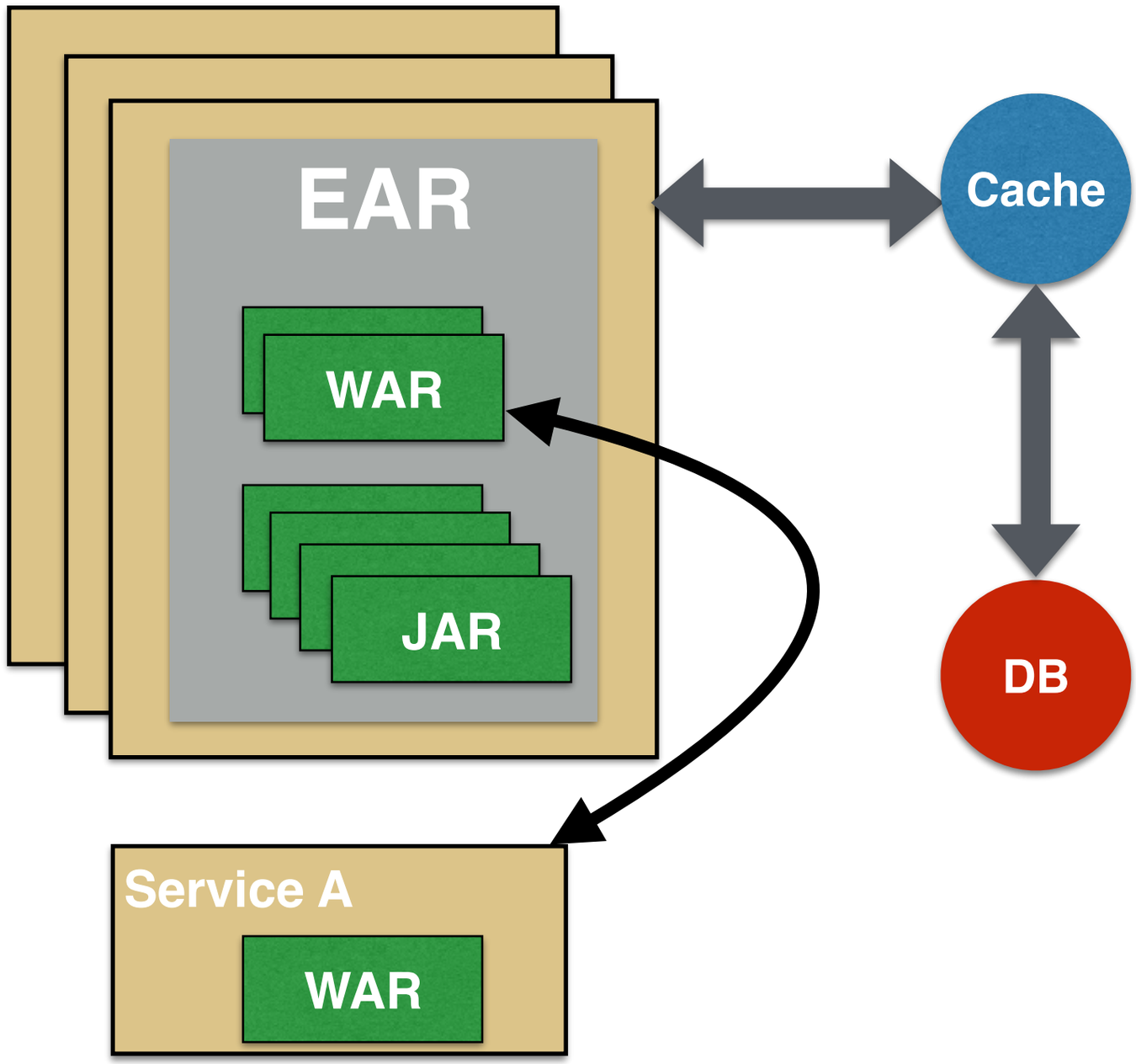
Strategies for decomposing

- Verb or usecase - e.g. Checkout UI
- Noun - e.g. Catalog product service

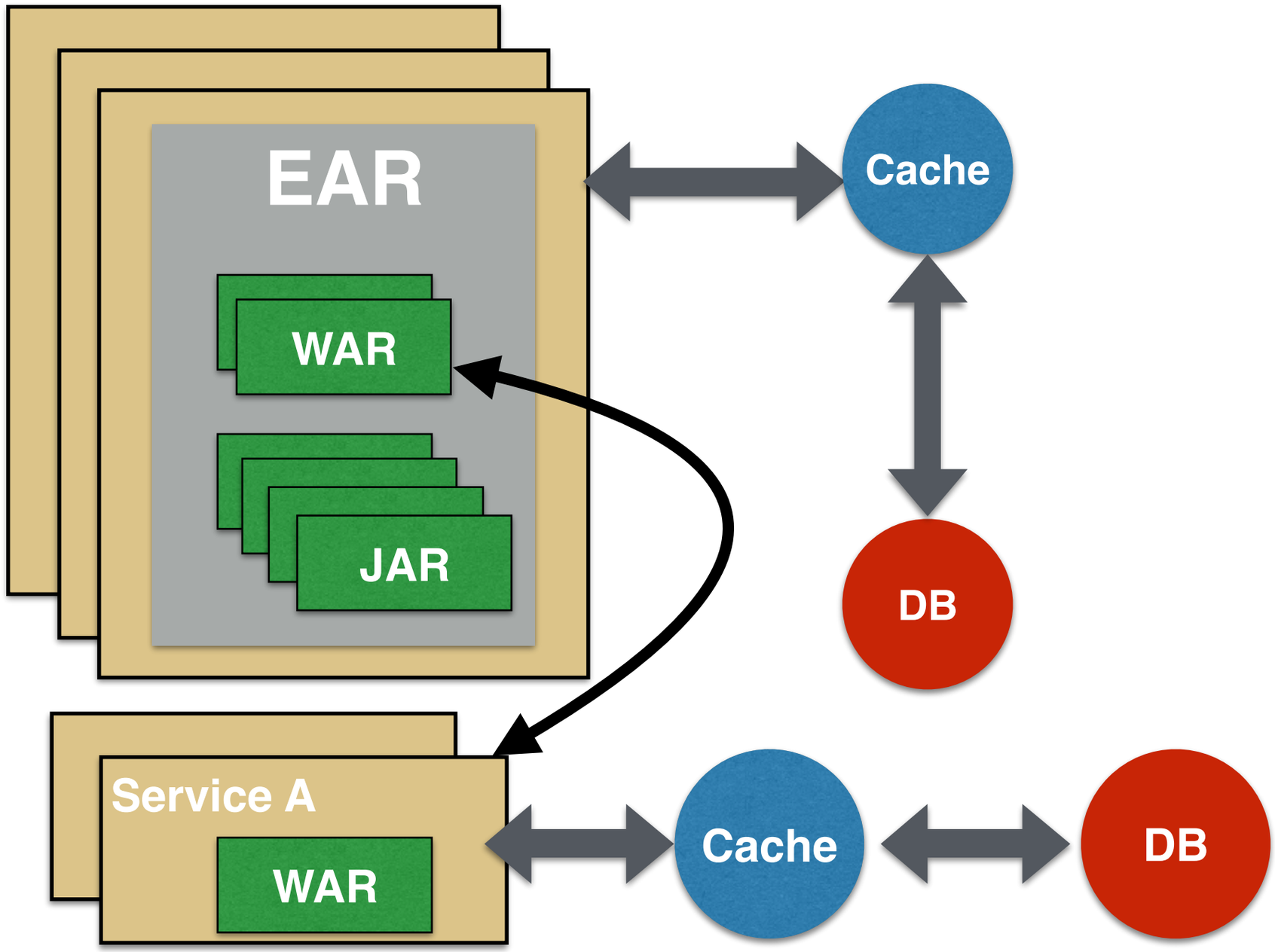
Strategies for decomposing

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

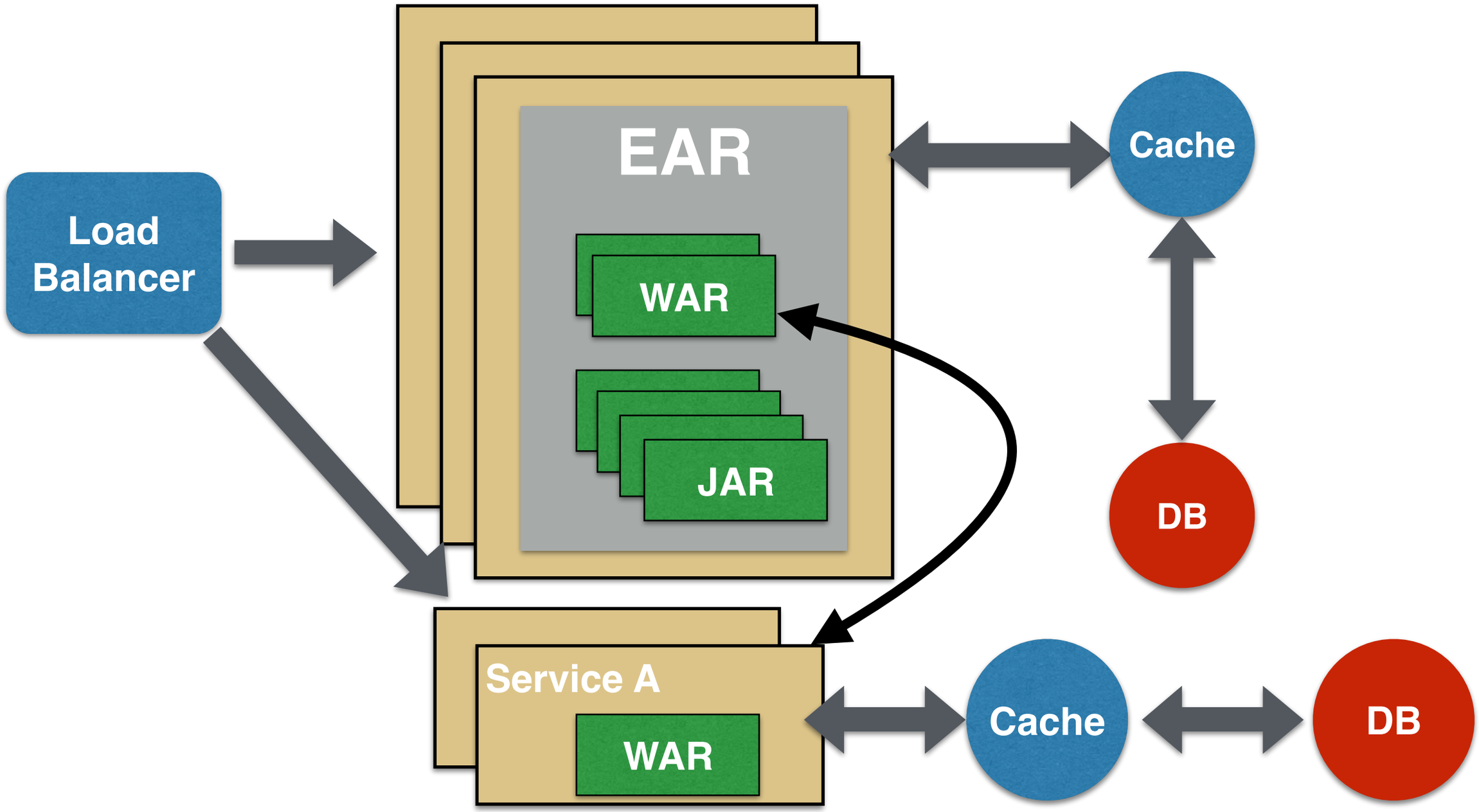
Towards microservices



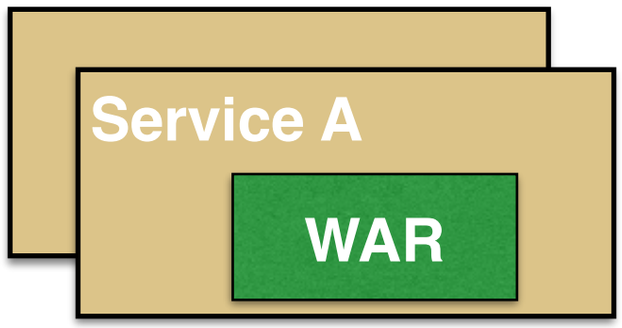
Towards microservices

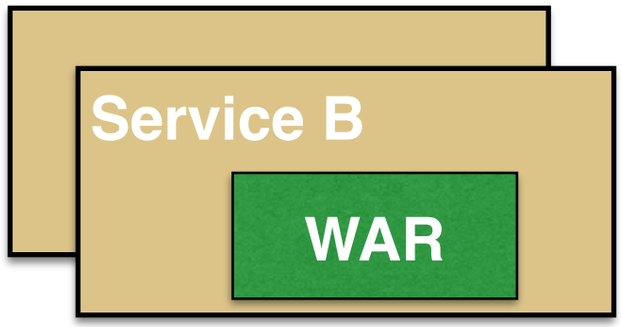


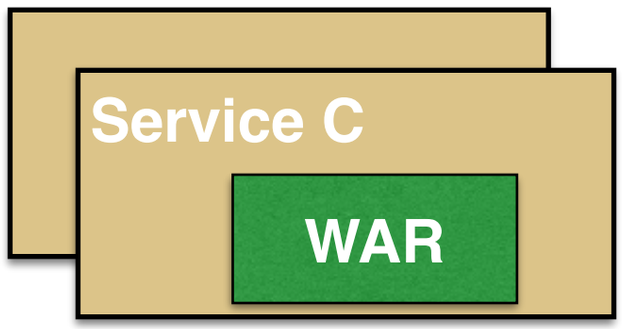
Towards microservices



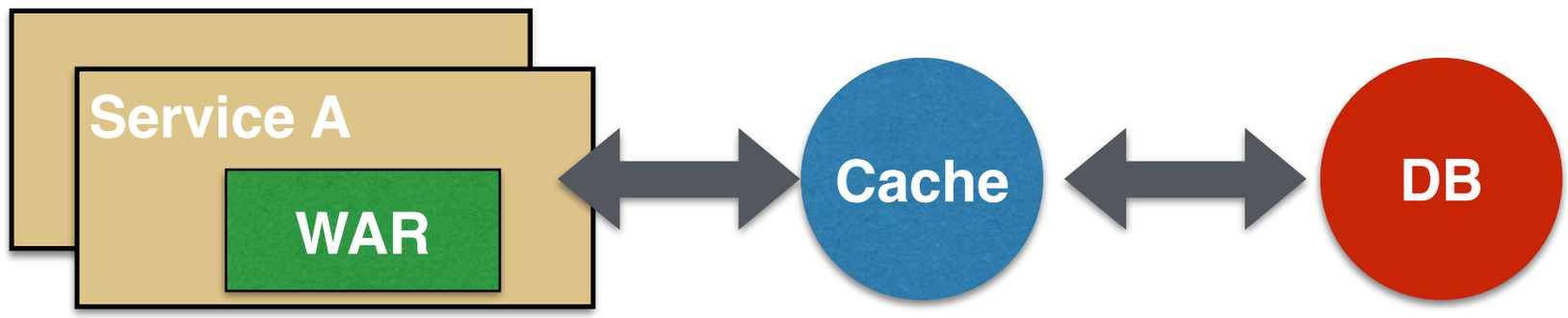
Aggregator Pattern #1

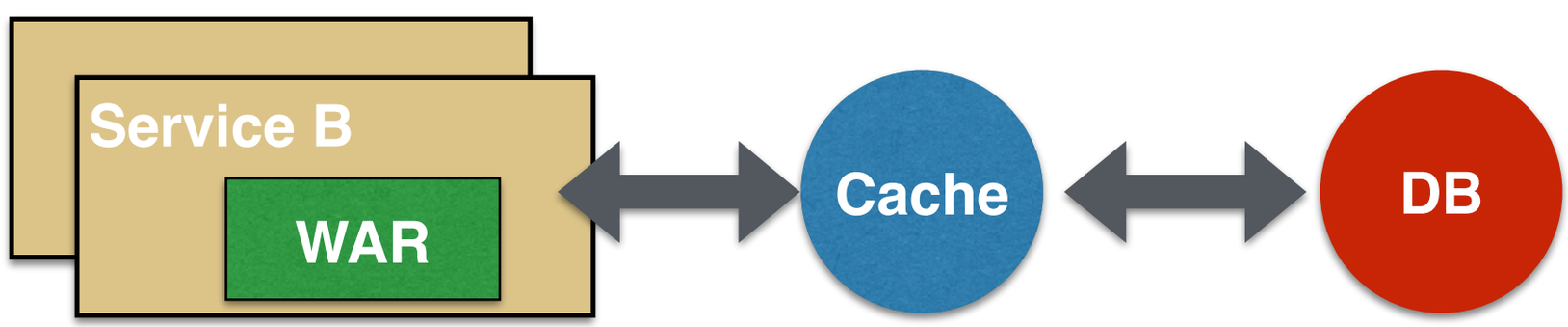


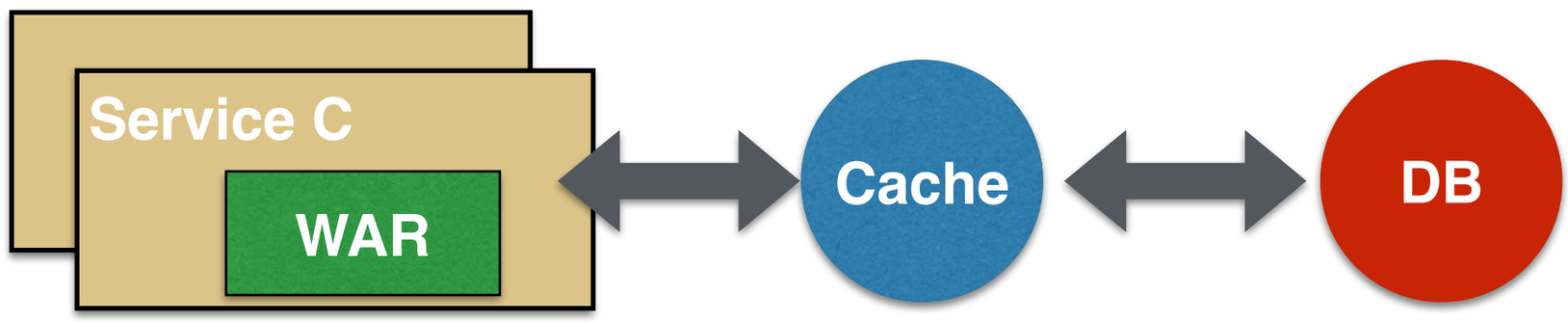




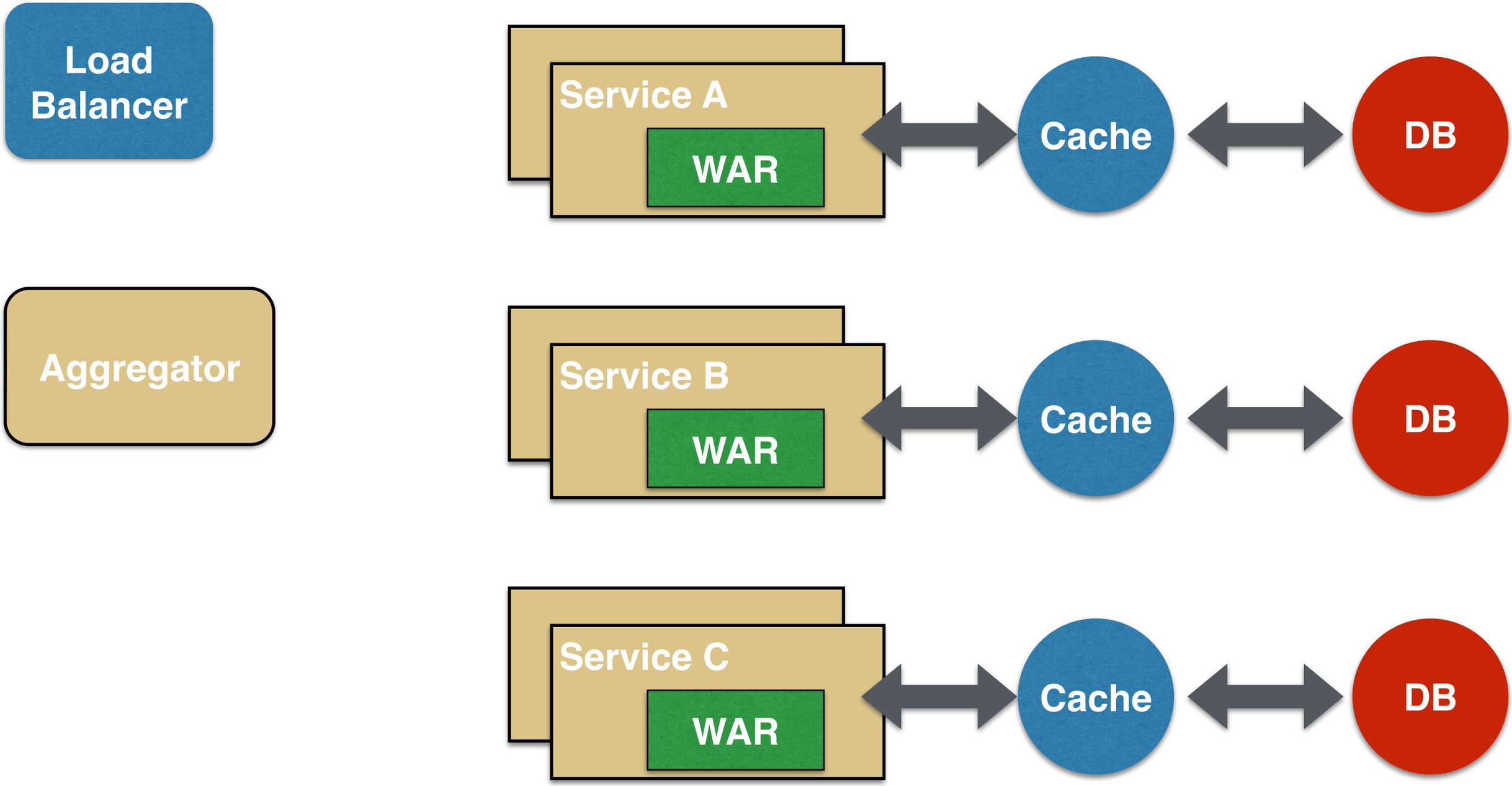
Aggregator Pattern #1



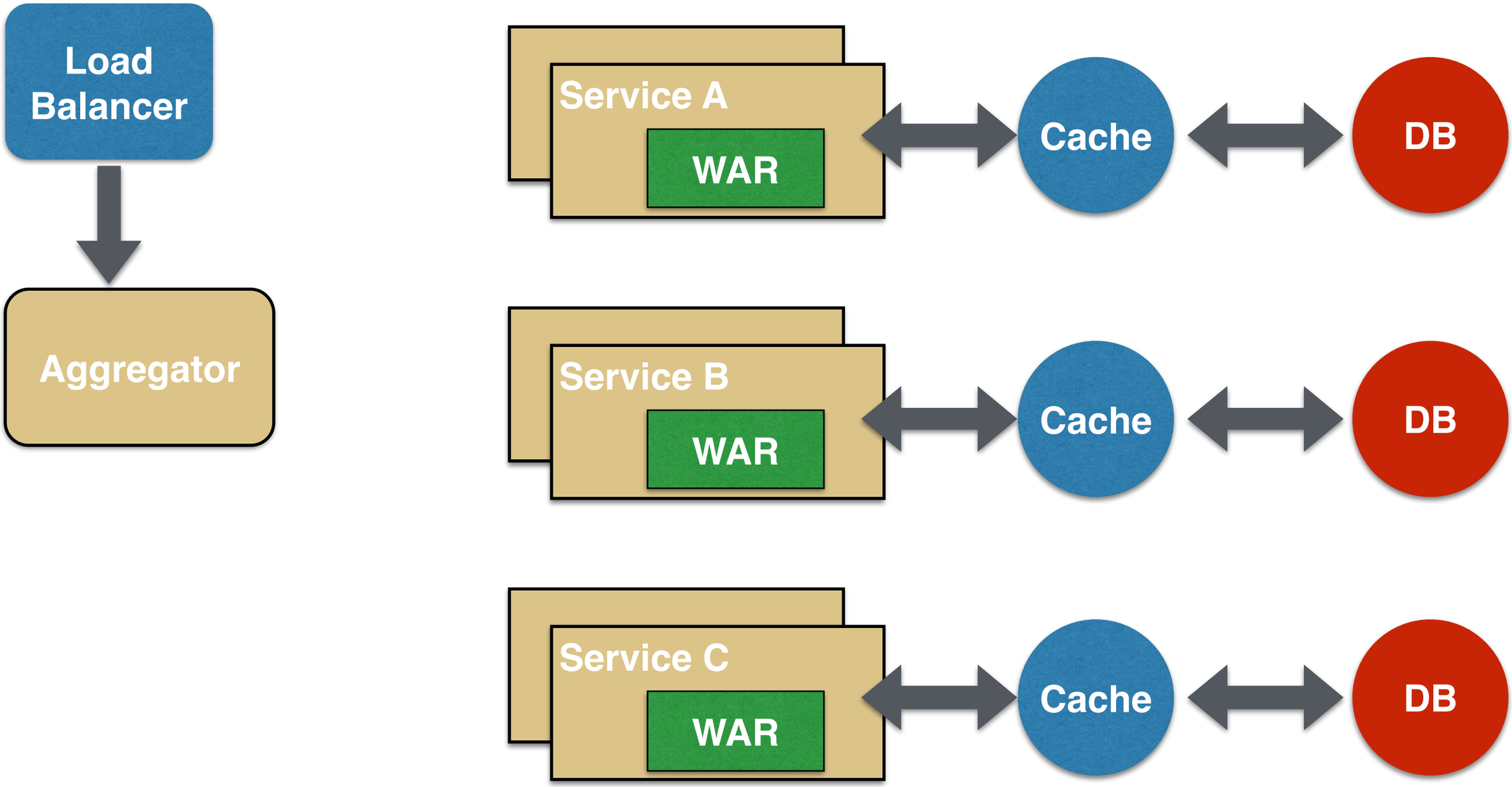




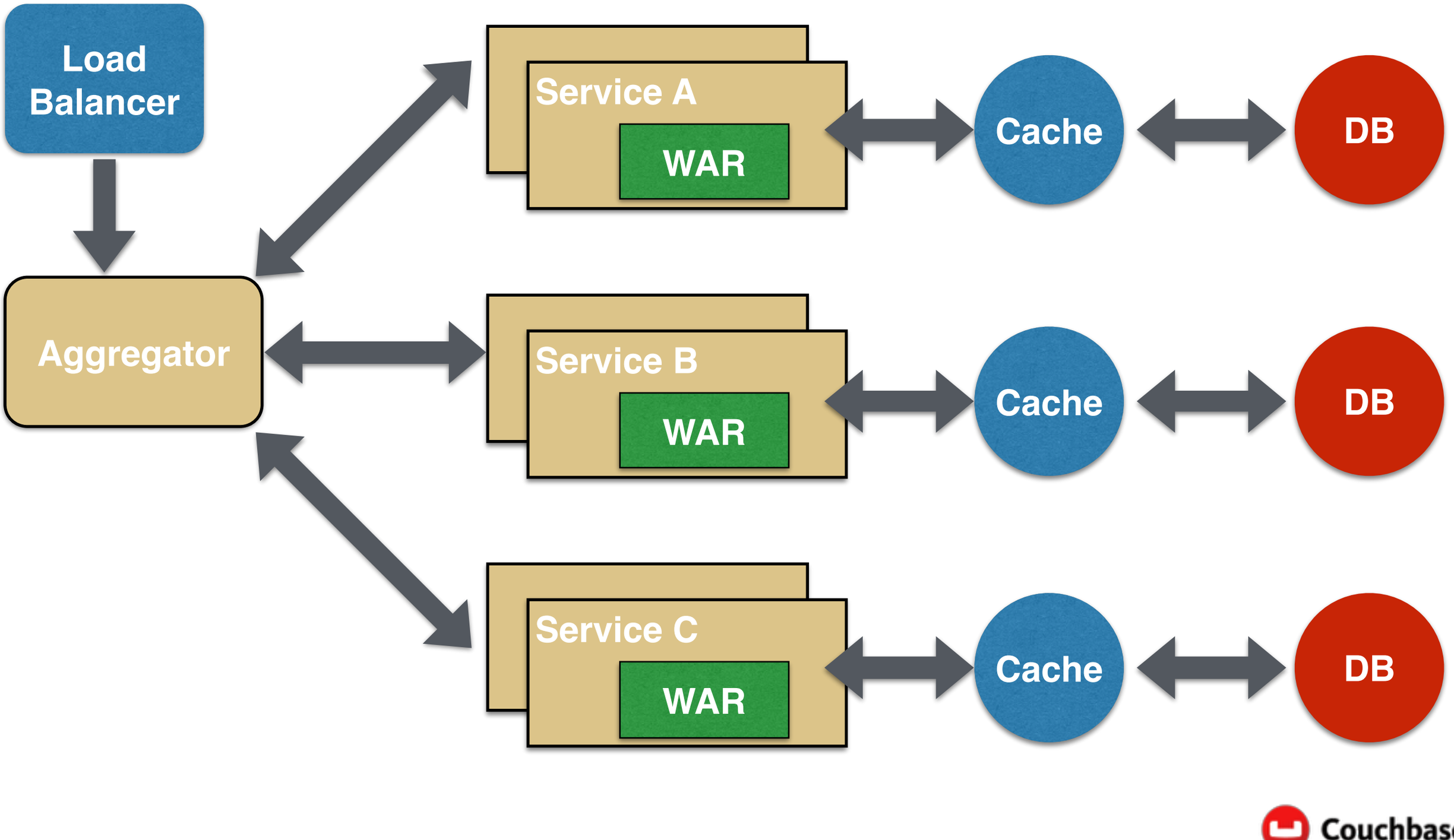
Aggregator Pattern #1



Aggregator Pattern #1

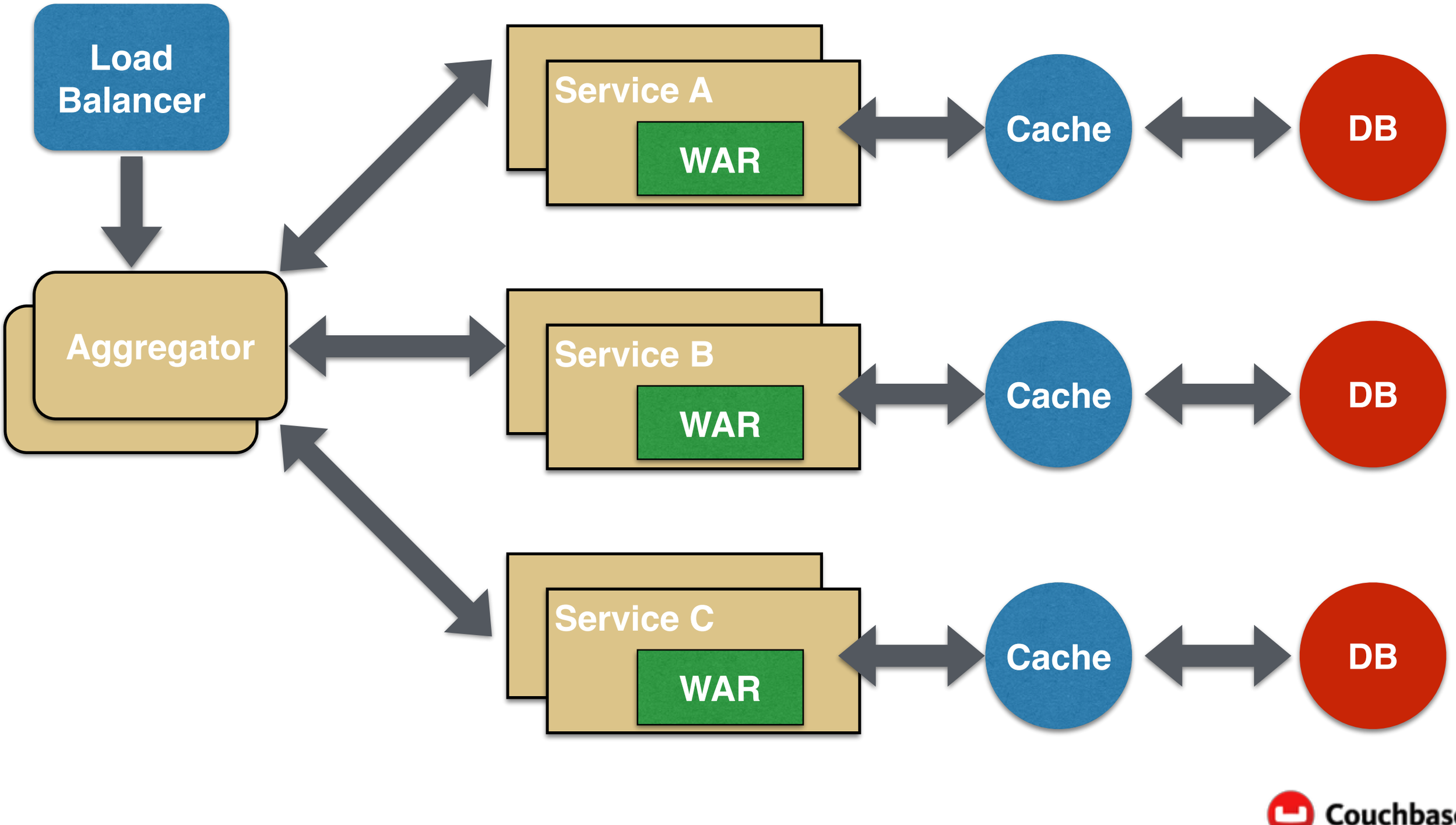


Aggregator Pattern #1



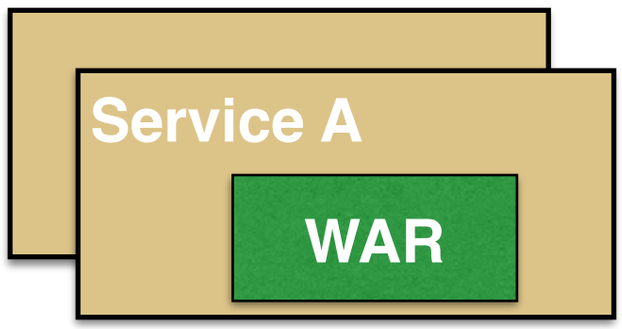
Couchbase

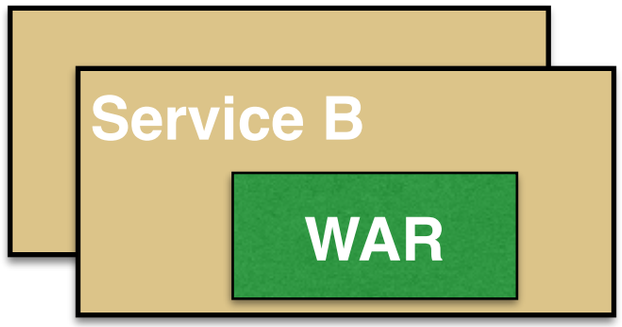
Aggregator Pattern #1

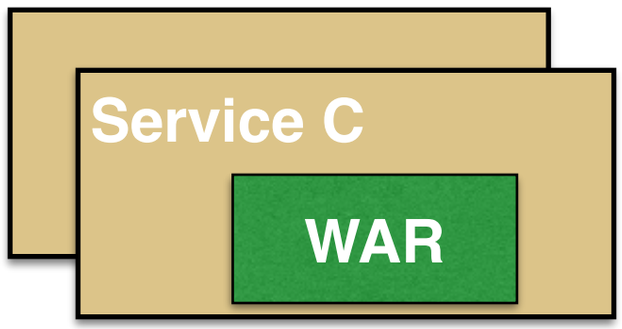


Couchbase

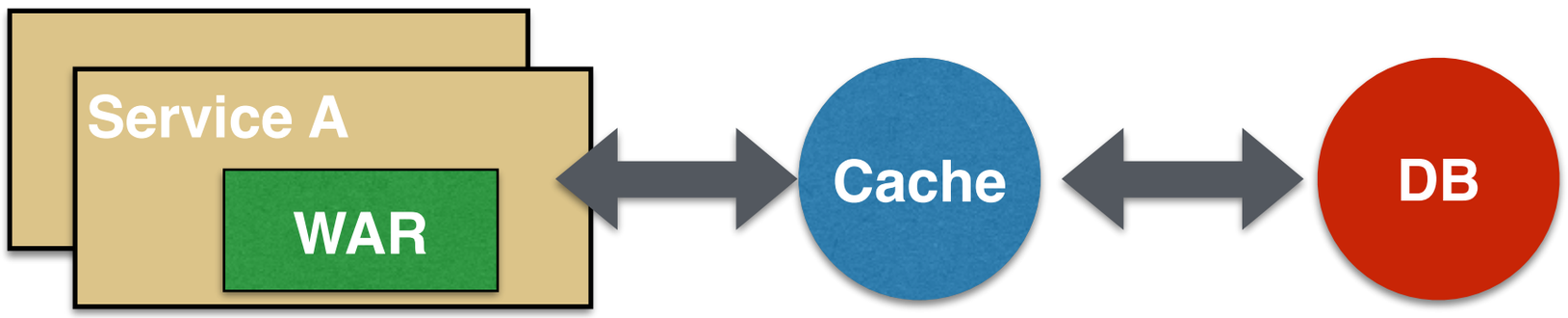
Proxy Pattern #2

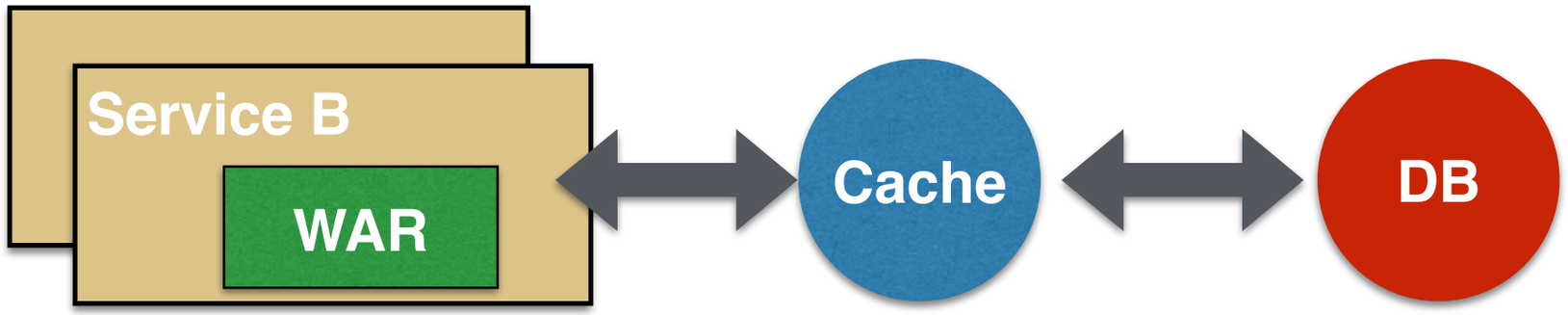


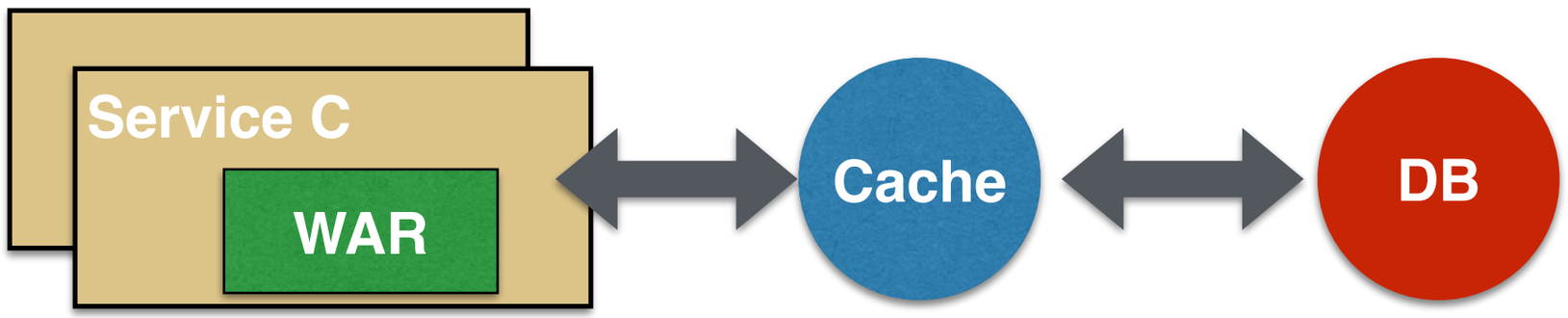




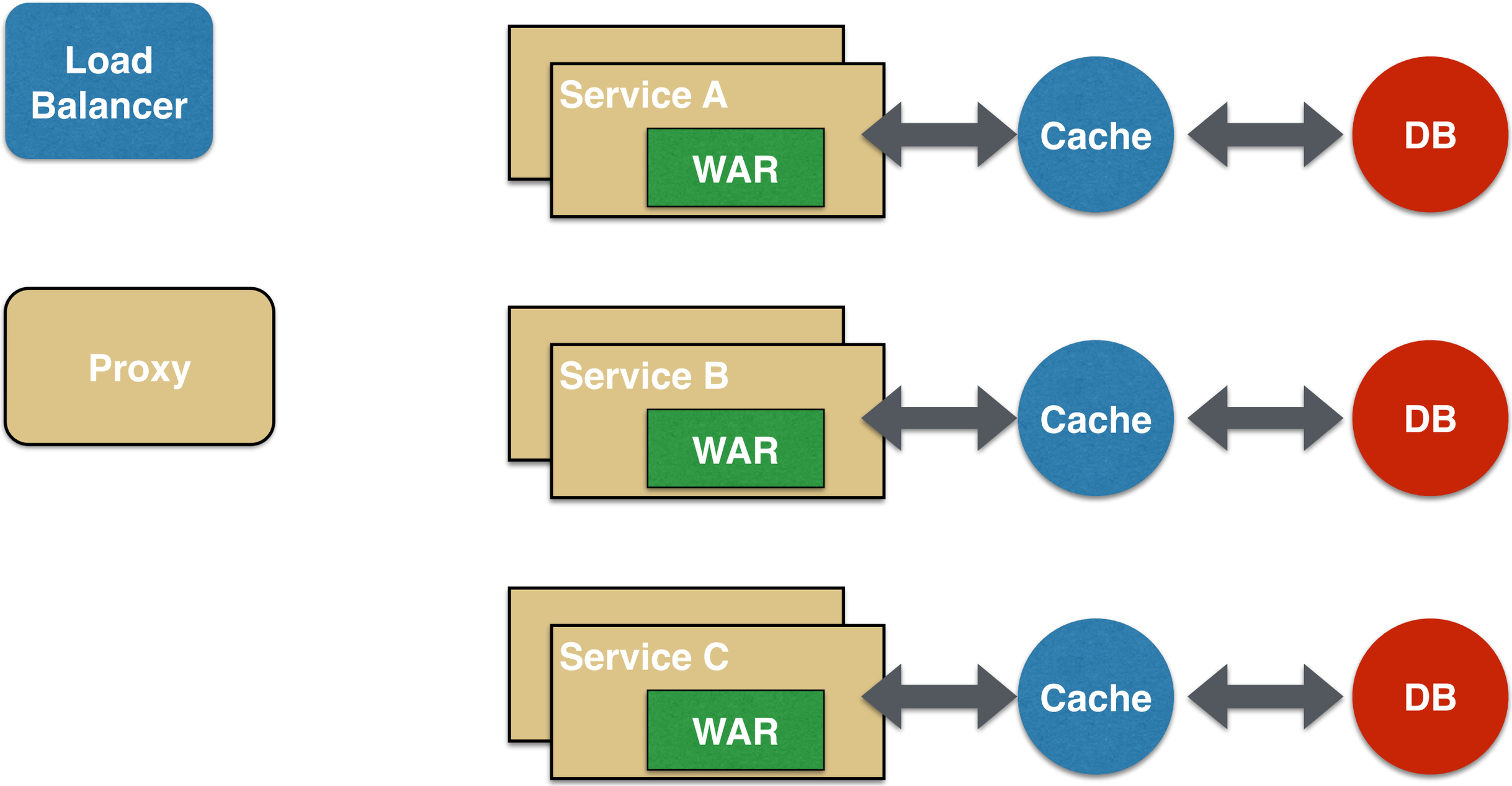
Proxy Pattern #2



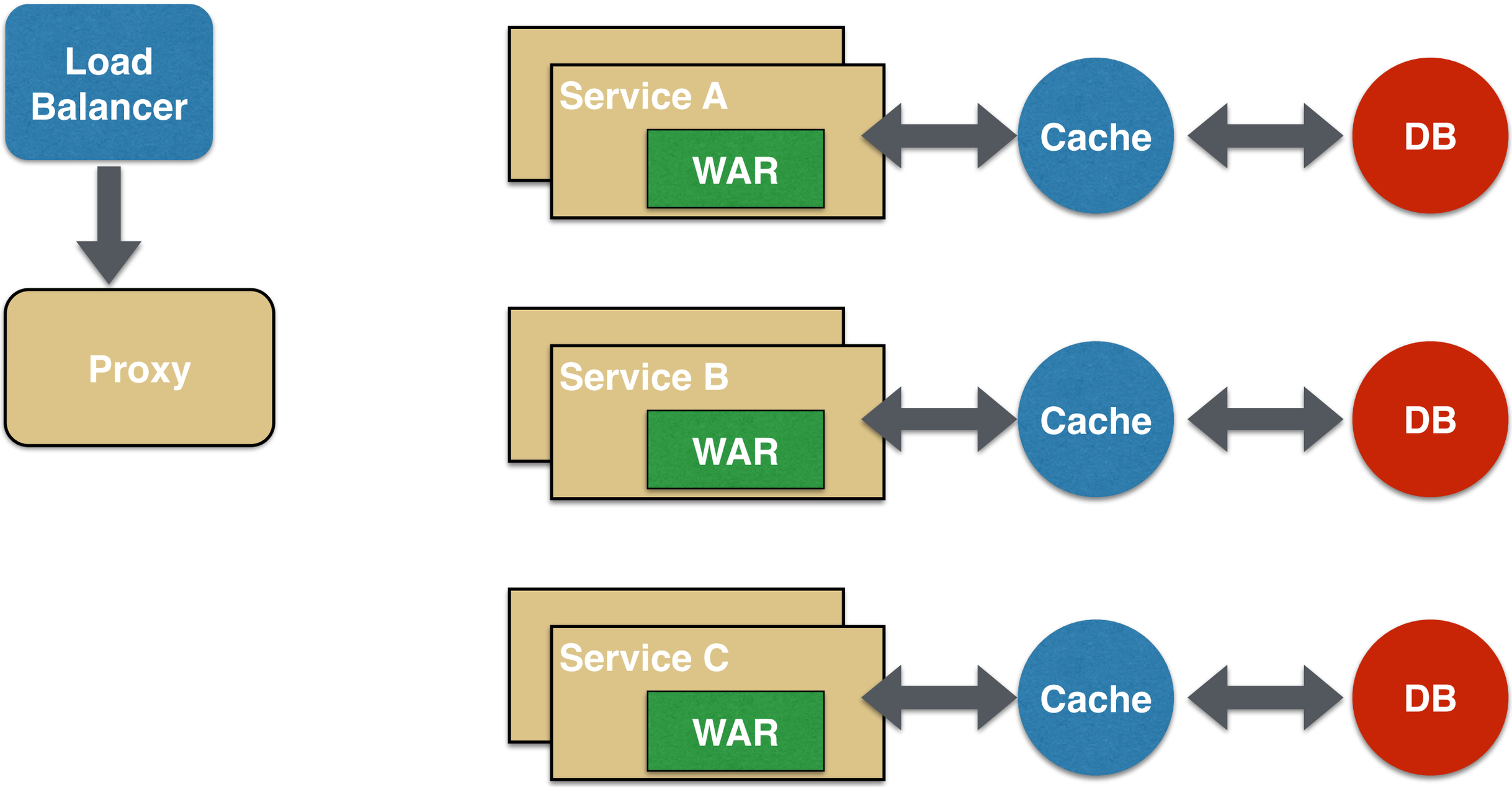




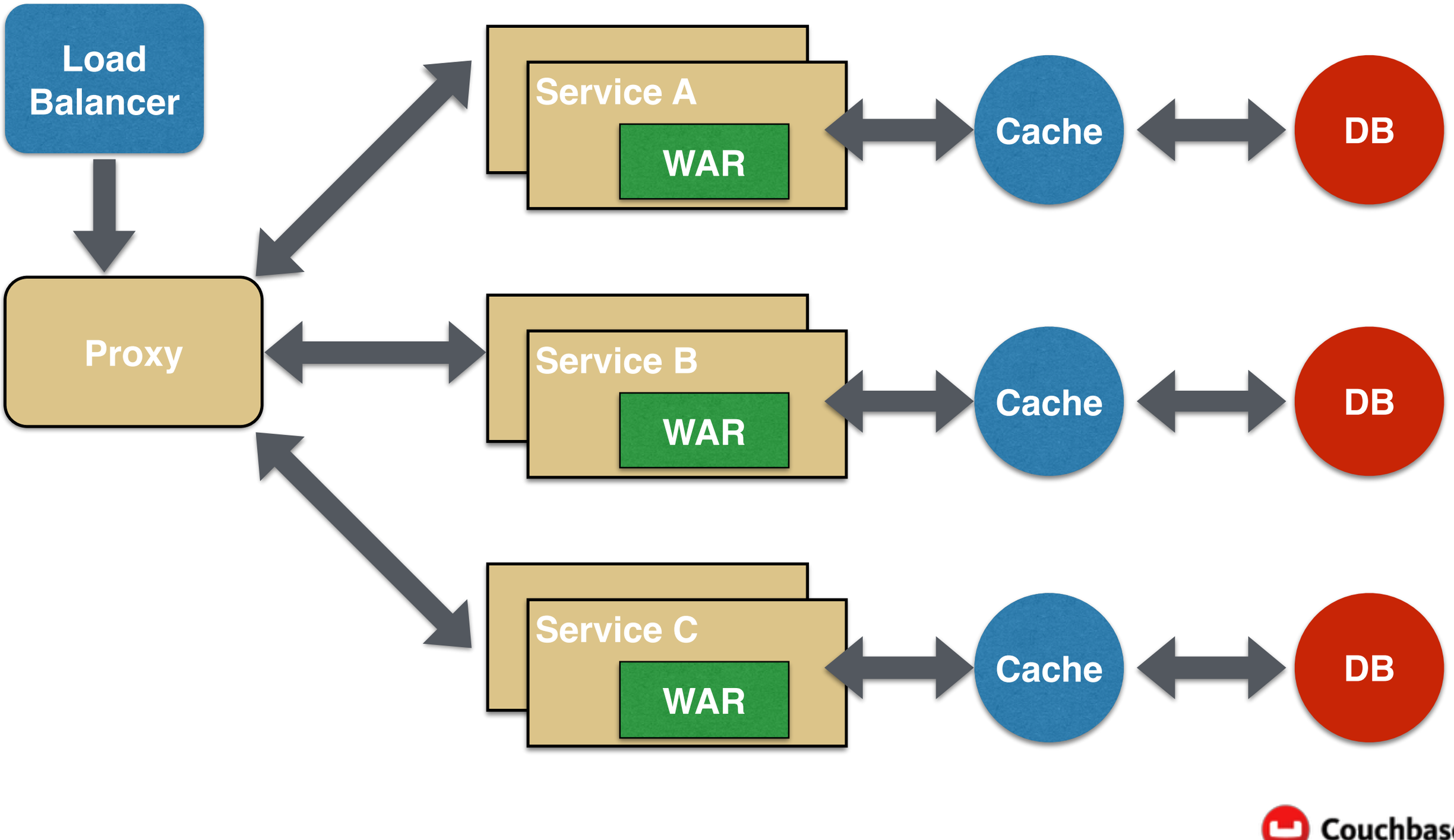
Proxy Pattern #2



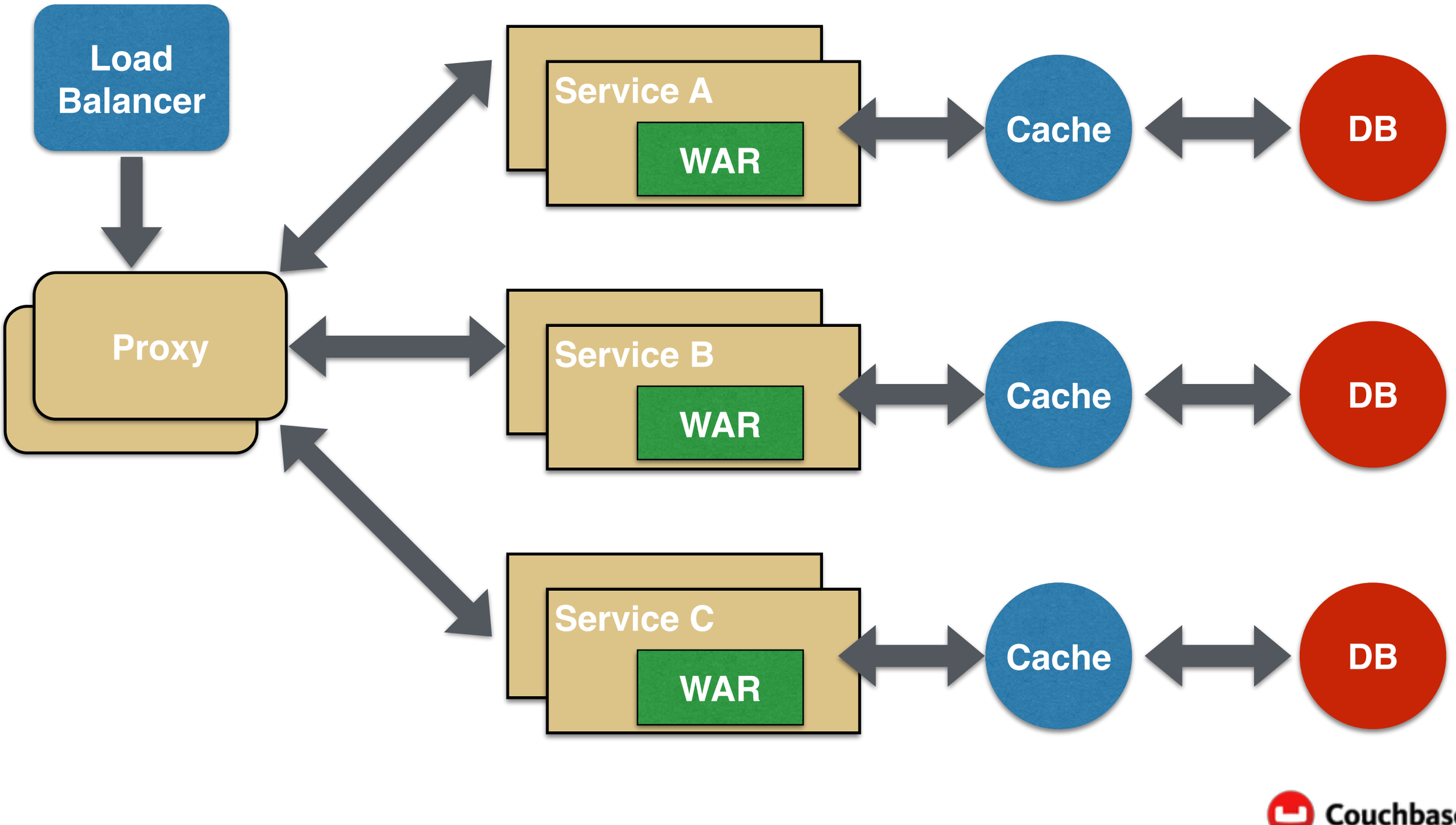
Proxy Pattern #2



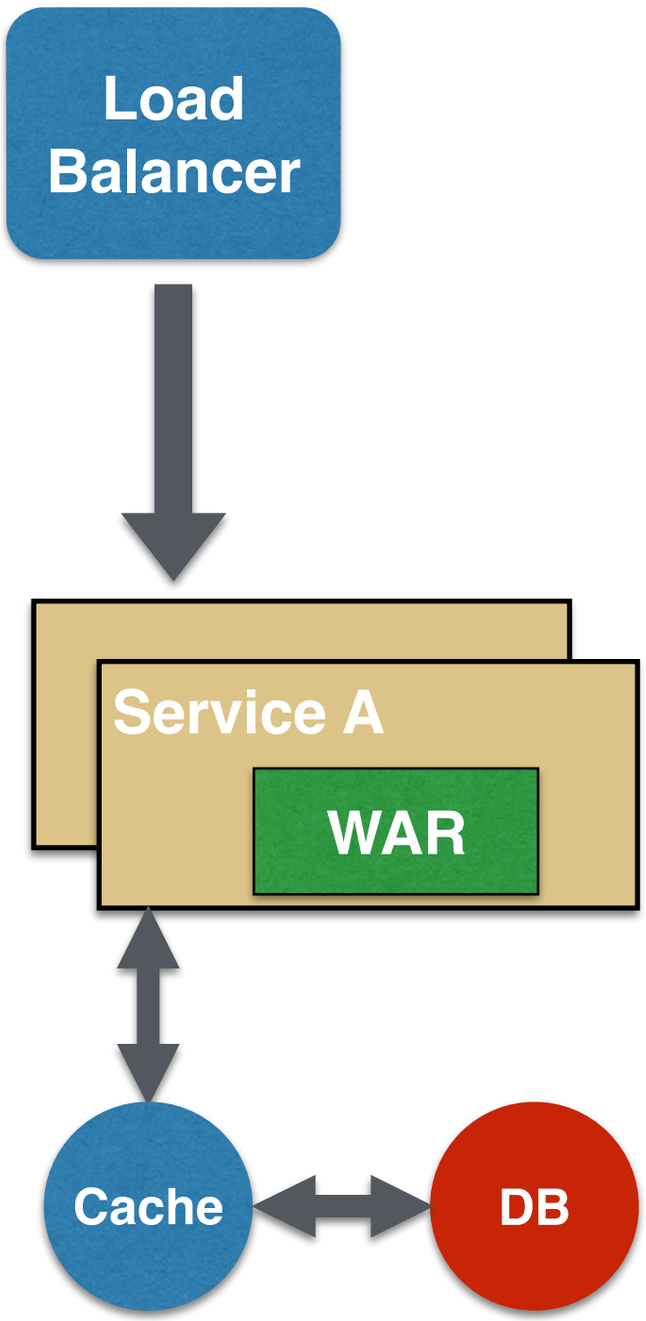
Proxy Pattern #2



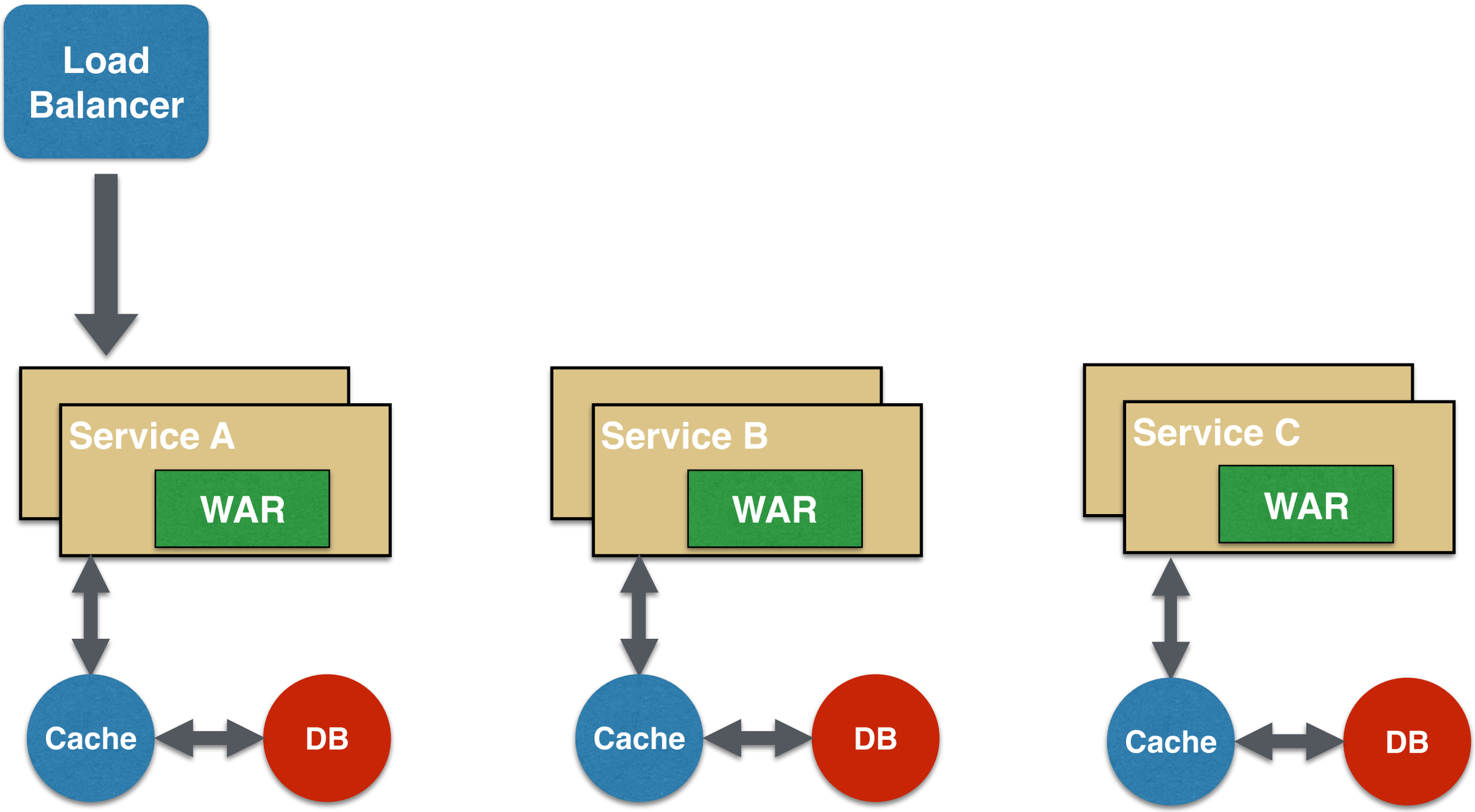
Proxy Pattern #2



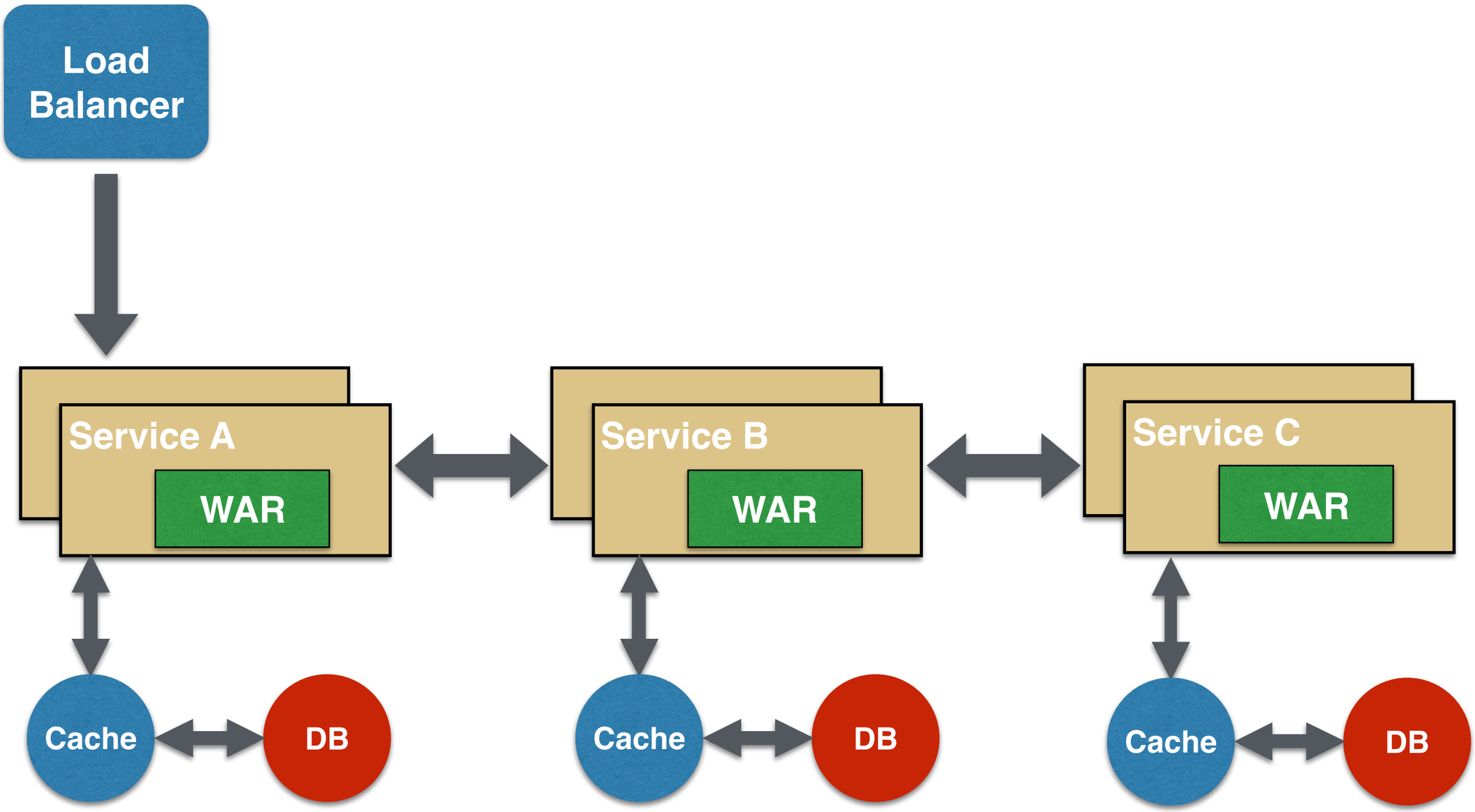
Chained Pattern #3



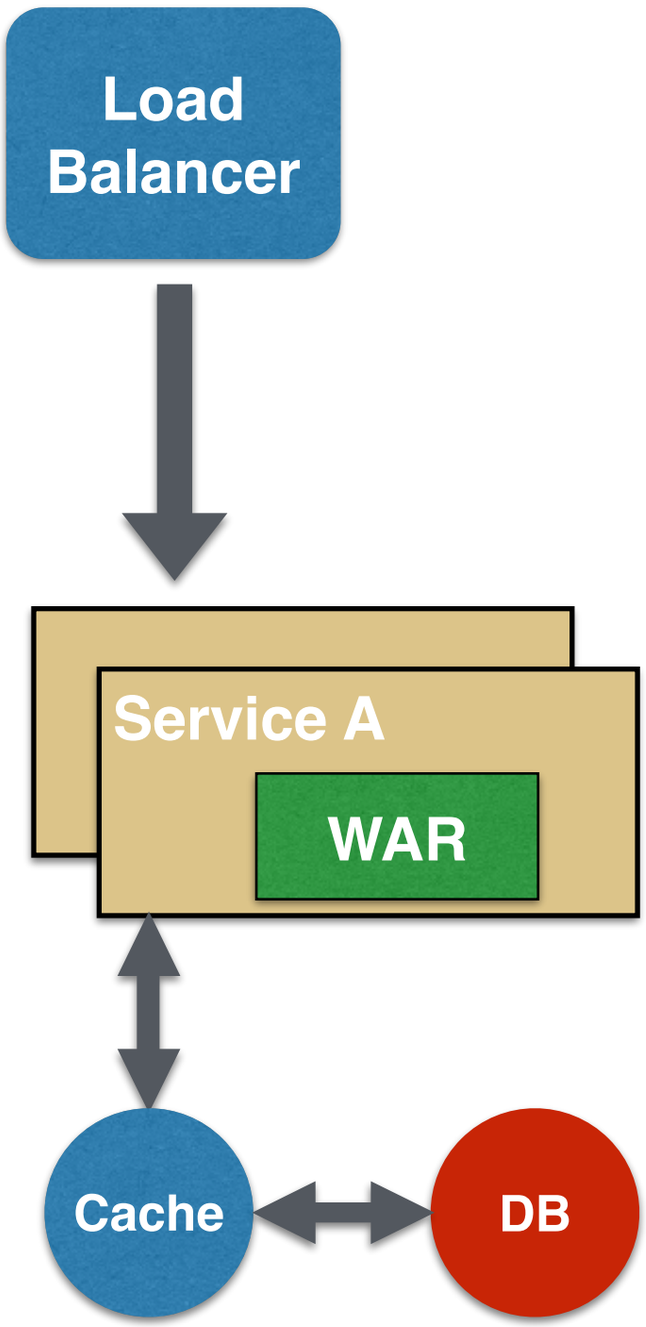
Chained Pattern #3



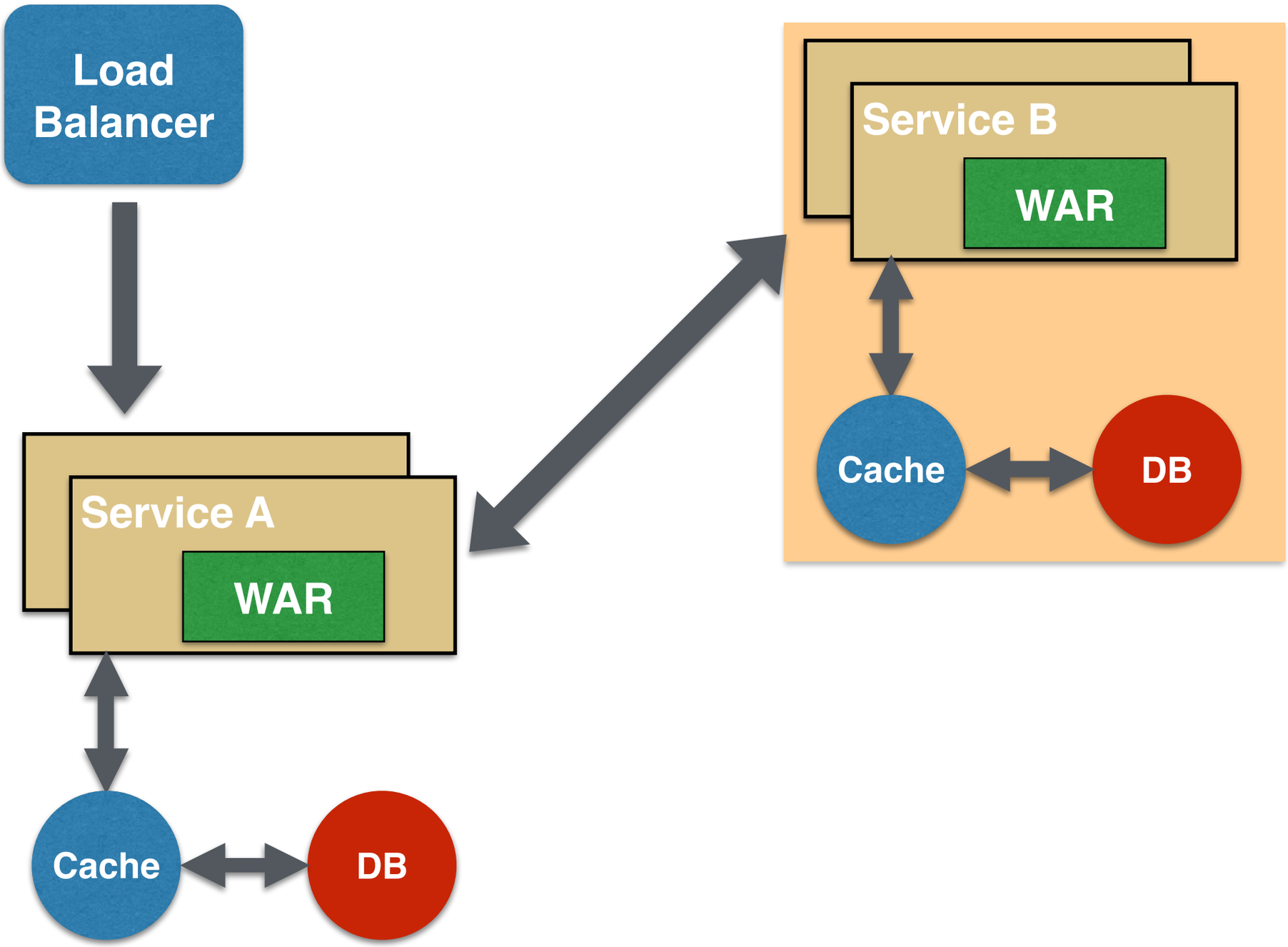
Chained Pattern #3



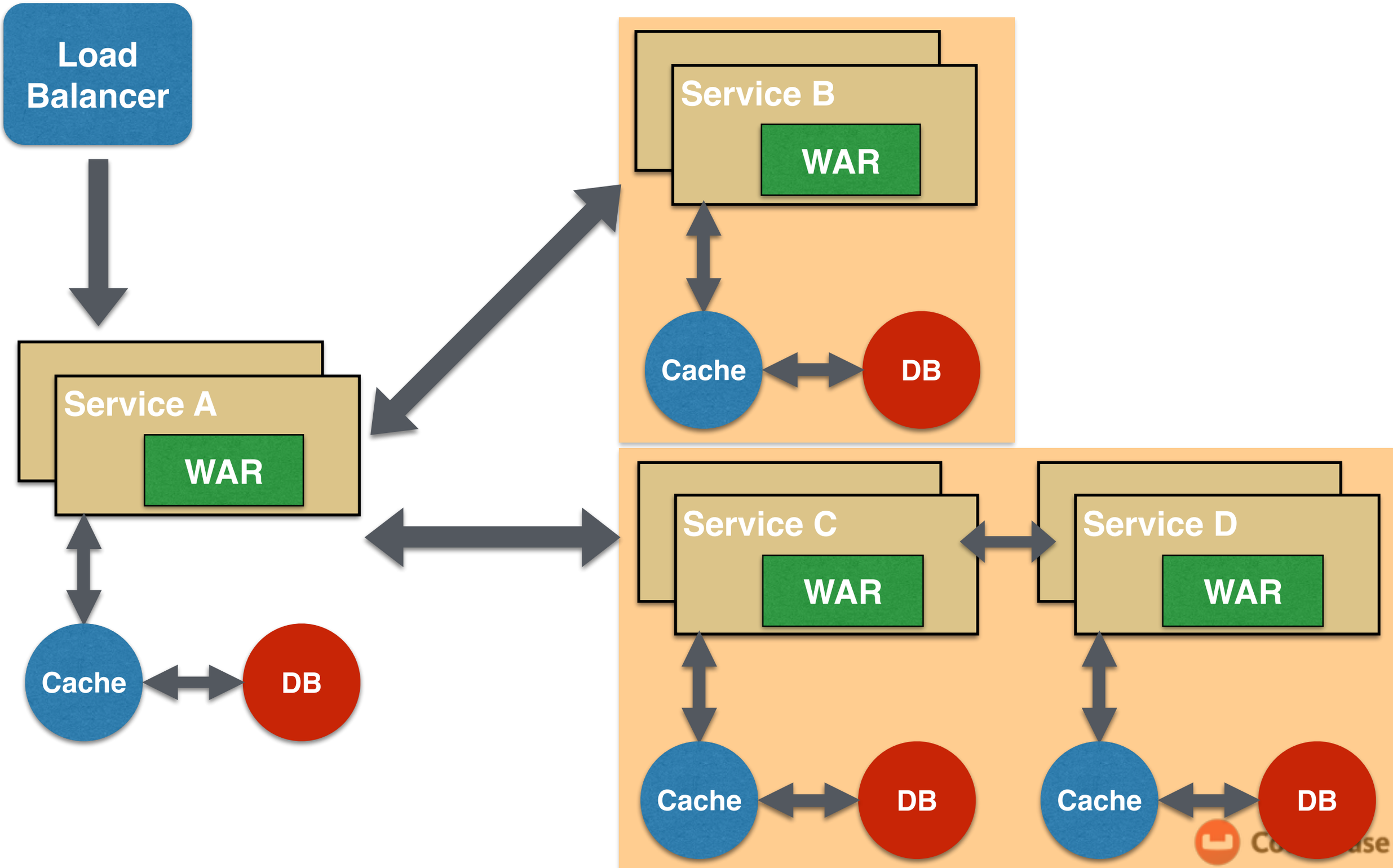
Branch Pattern #4



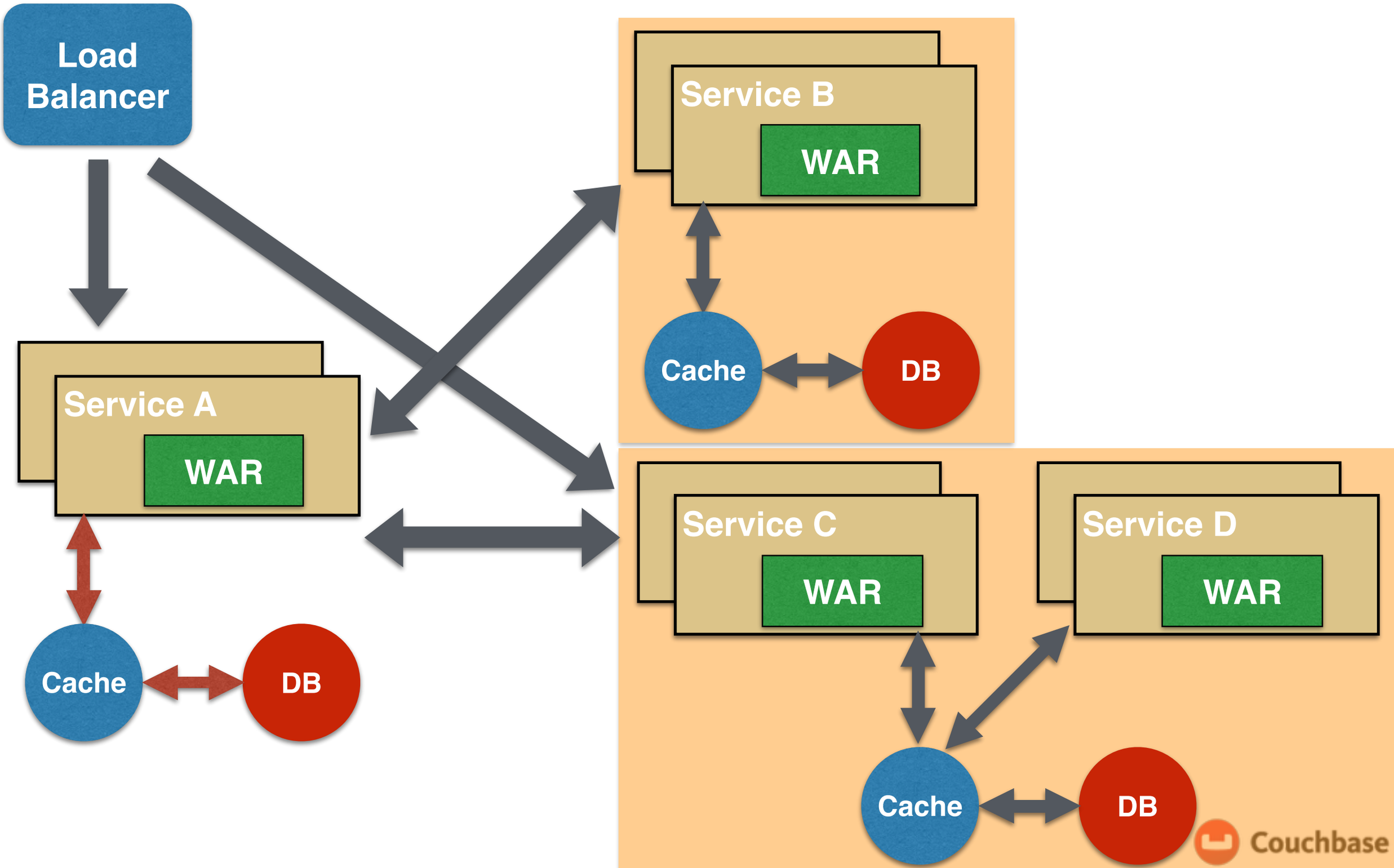
Branch Pattern #4



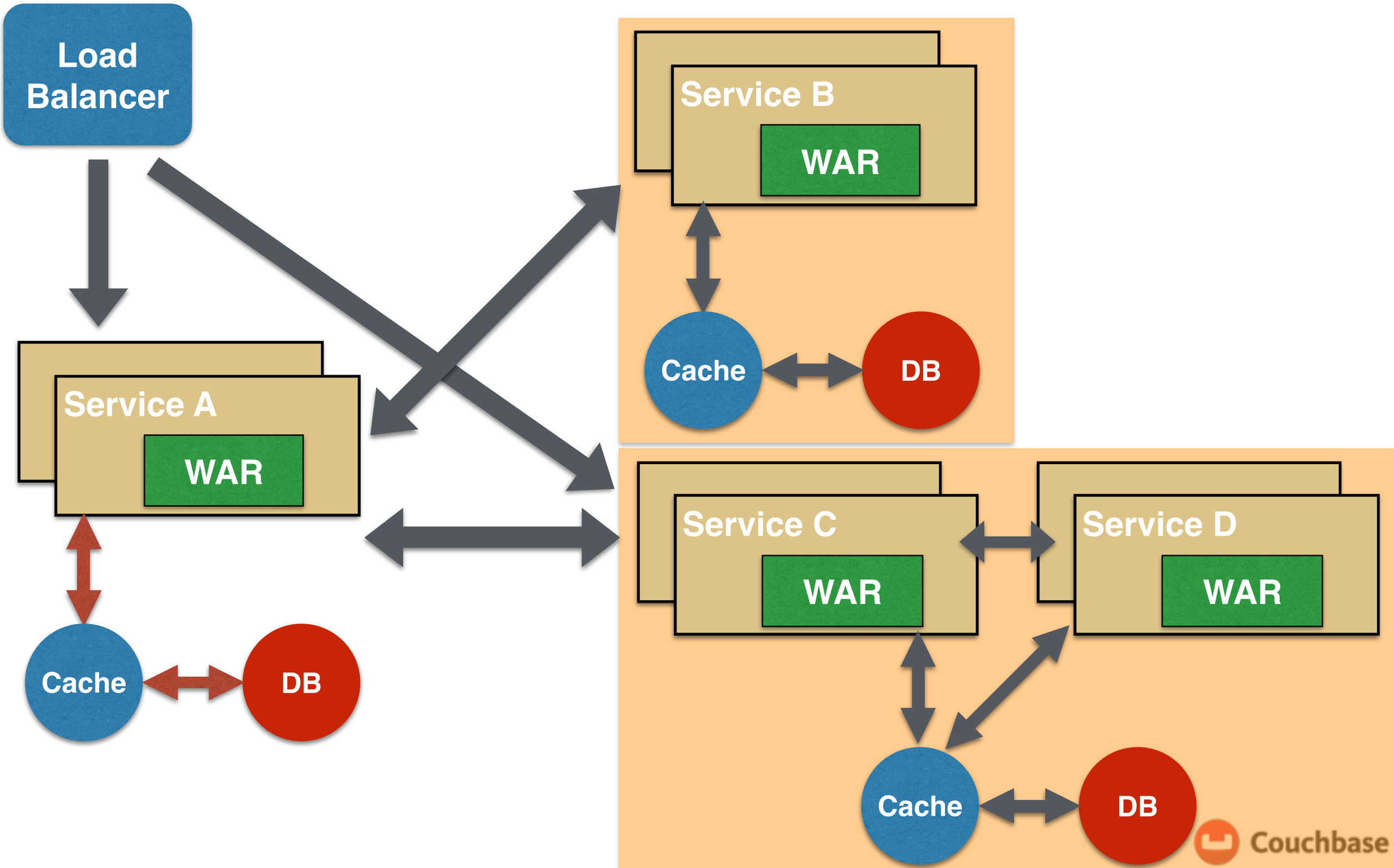
Branch Pattern #4



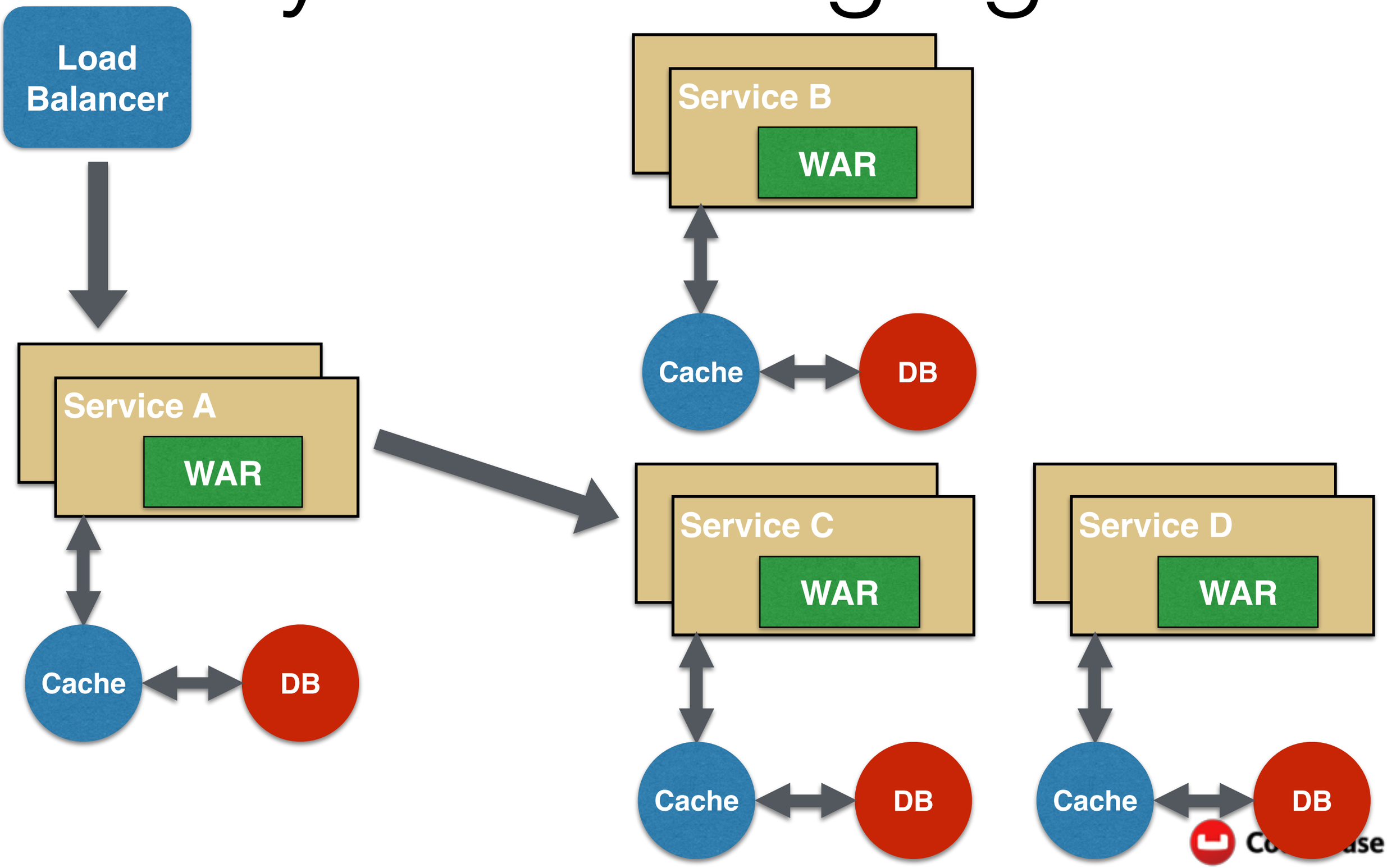
Shared Resources #5



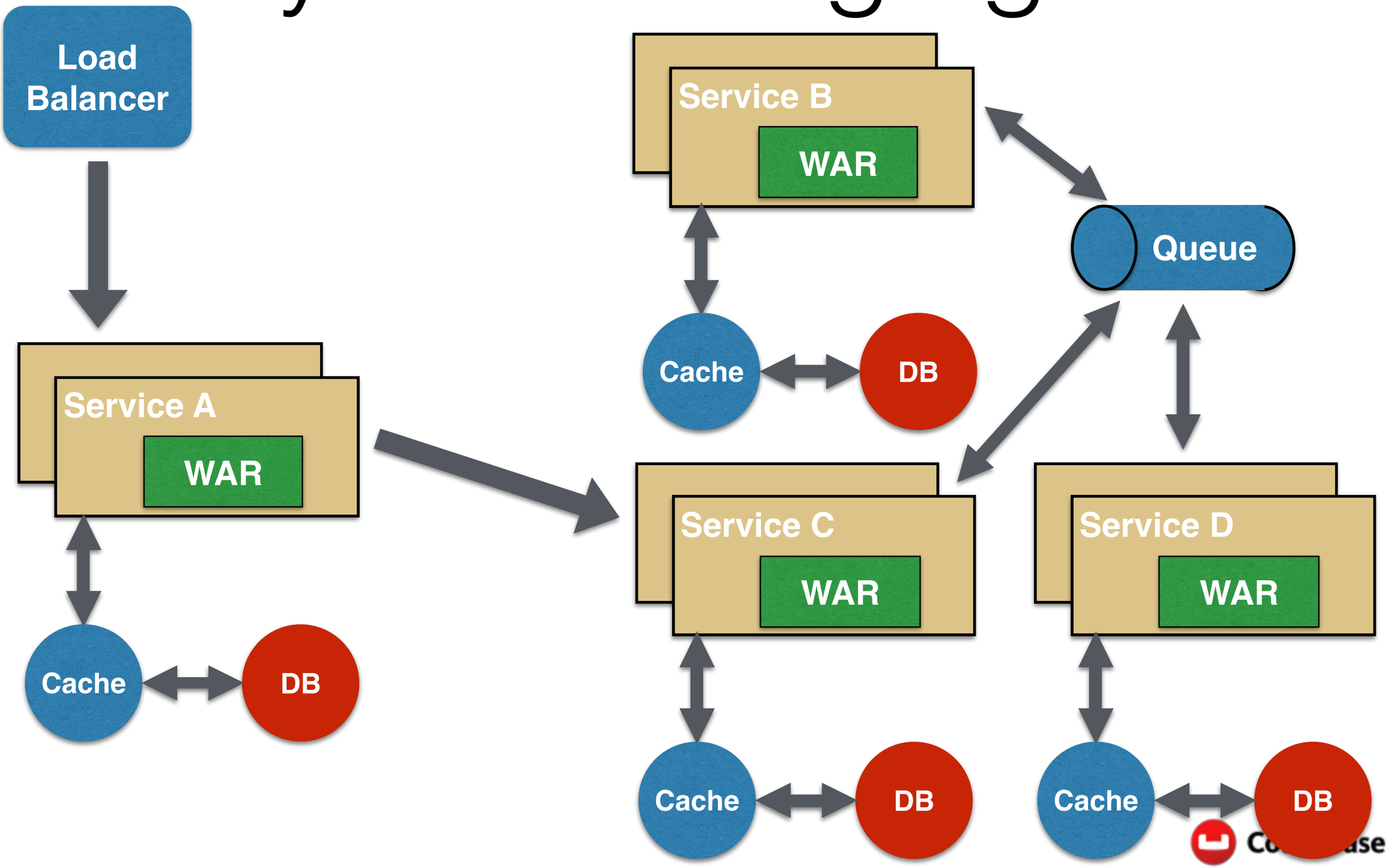
Shared Resources #5



Async Messaging #5



Async Messaging #5



Advantages of microservices

Advantages of microservices

- Easier to develop, understand, maintain

Advantages of microservices

- Easier to develop, understand, maintain
- Starts faster than a monolith, speeds up deployments

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

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

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

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

Advantages of microservices

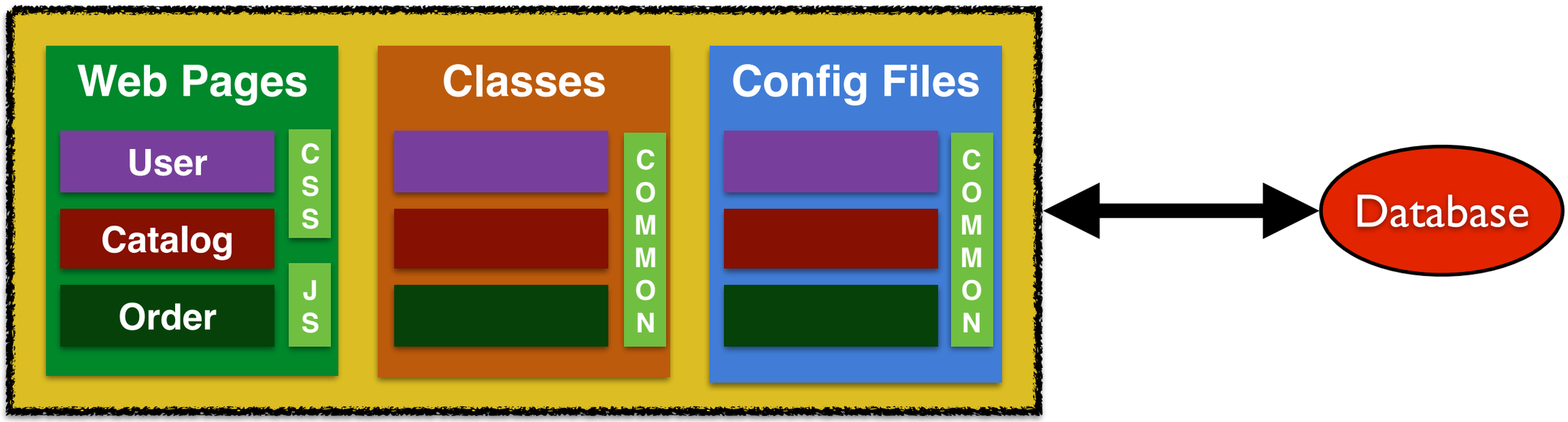
- Easier to develop, understand, maintain
- Starts faster than a monolith, speeds up deployments
- Local change can be easily deployed, great enabler of CD
- Each service can scale on X- and Z-axis
- Improves fault isolation
- Eliminates any long-term commitment to a technology stack
- Freedom of choice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

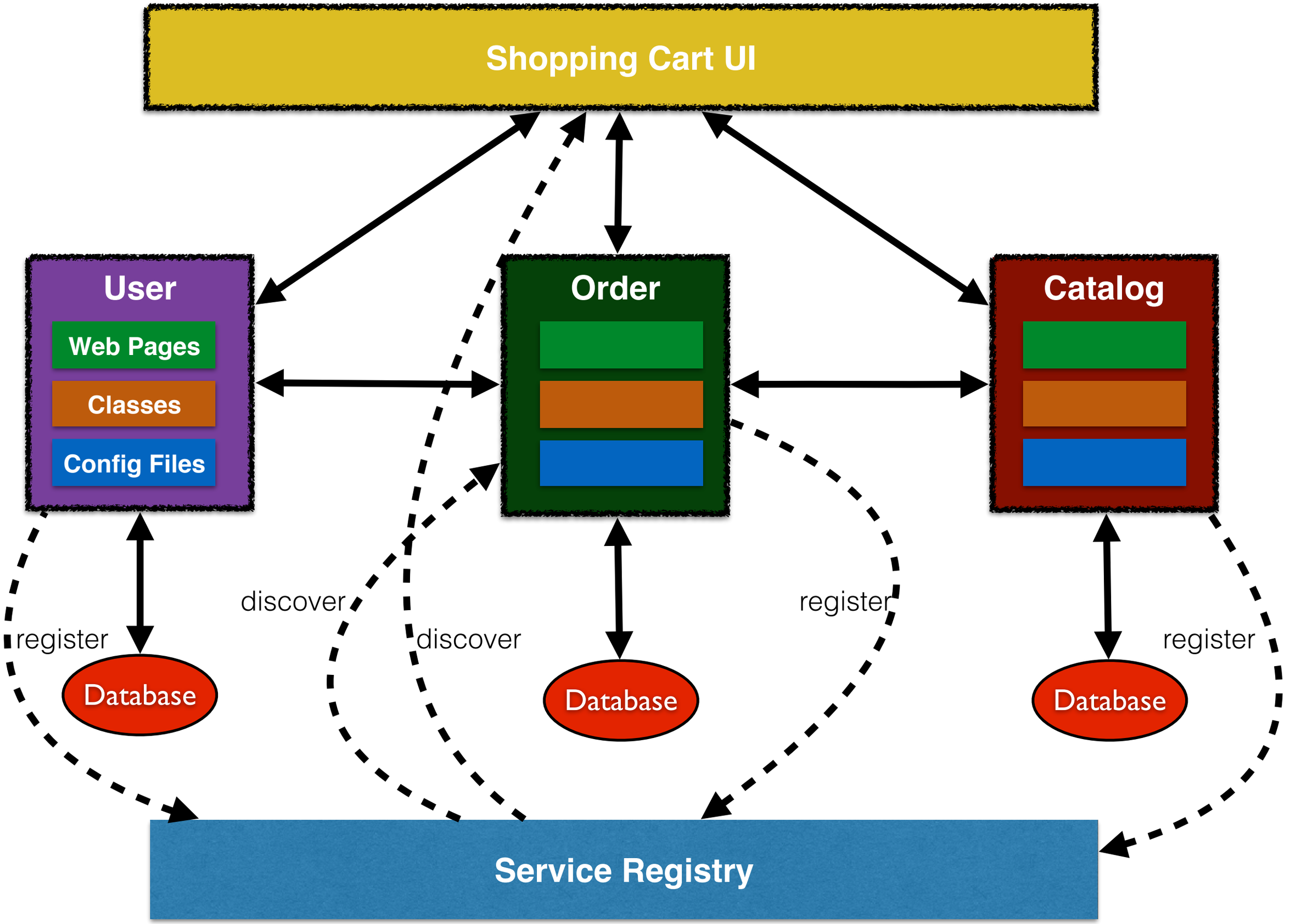
"If you can't build a [well-structured] monolith, what makes you think microservices are the answer?"

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/

Service Registry/Discovery

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

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

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/

Design Considerations

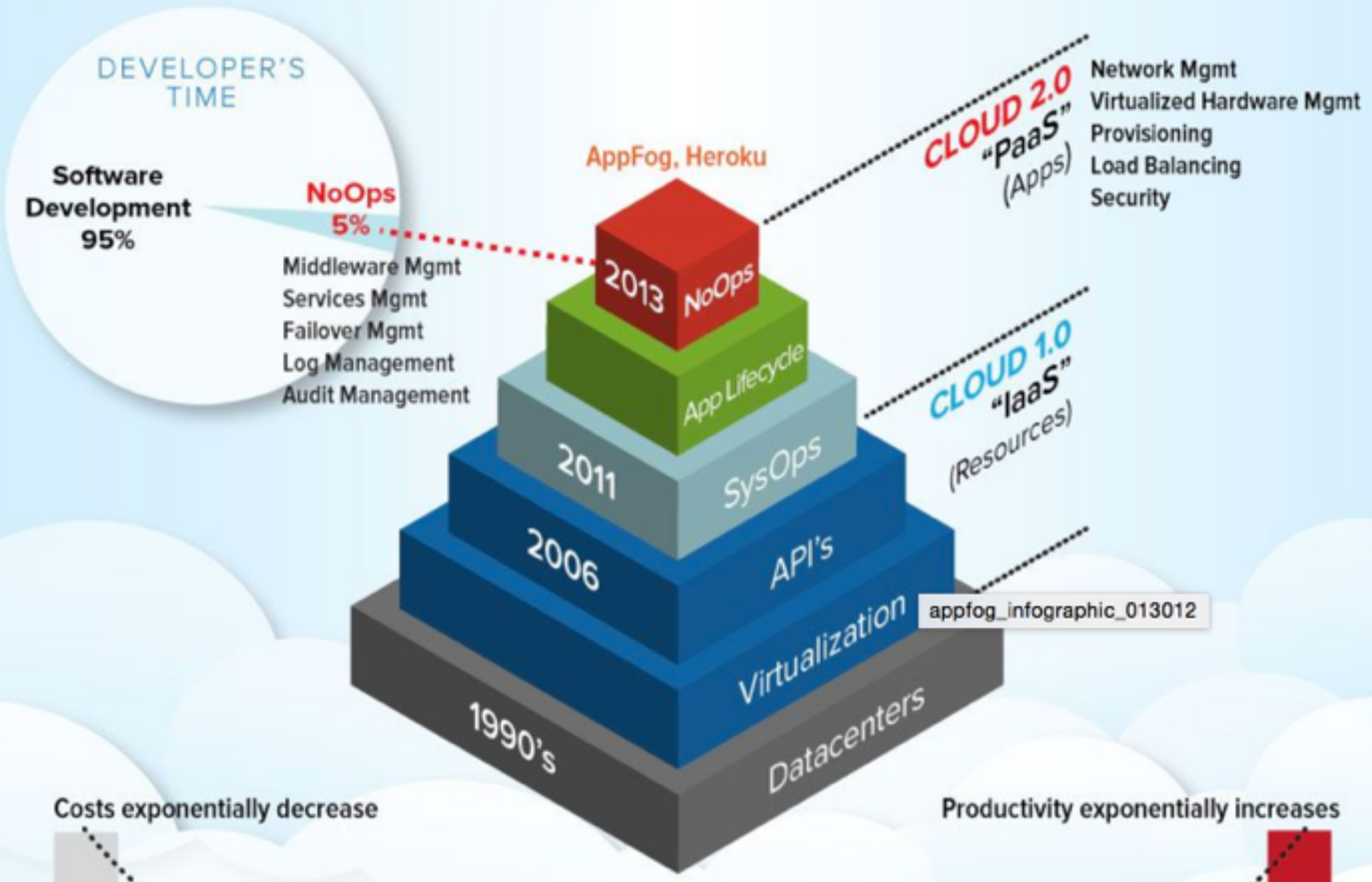
- UI and Full stack
  - Client-side composition (JavaScript?)
  - Server-side HTML generation (JSF?)
    - One service, one UI
- REST Services
- Event sequencing instead of 2PC
- API Management

NoOps

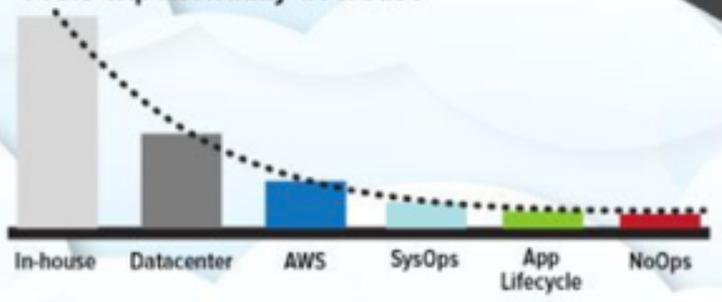
- Service replication (Kubernetes)
- Dependency resolution (Nexus)
- Failover (Circuit Breaker)
- Resiliency (Circuit Breaker)
- Service monitoring, alerts and events (ELK)

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.

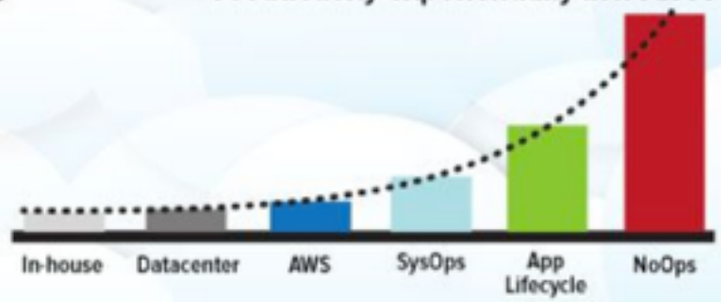


Costs exponentially decrease



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/

Drawbacks of microservices

Drawbacks of microservices

- Additional complexity of distributed systems

Drawbacks of microservices

- Additional complexity of distributed systems
- Significant operational complexity, need high-level of automation

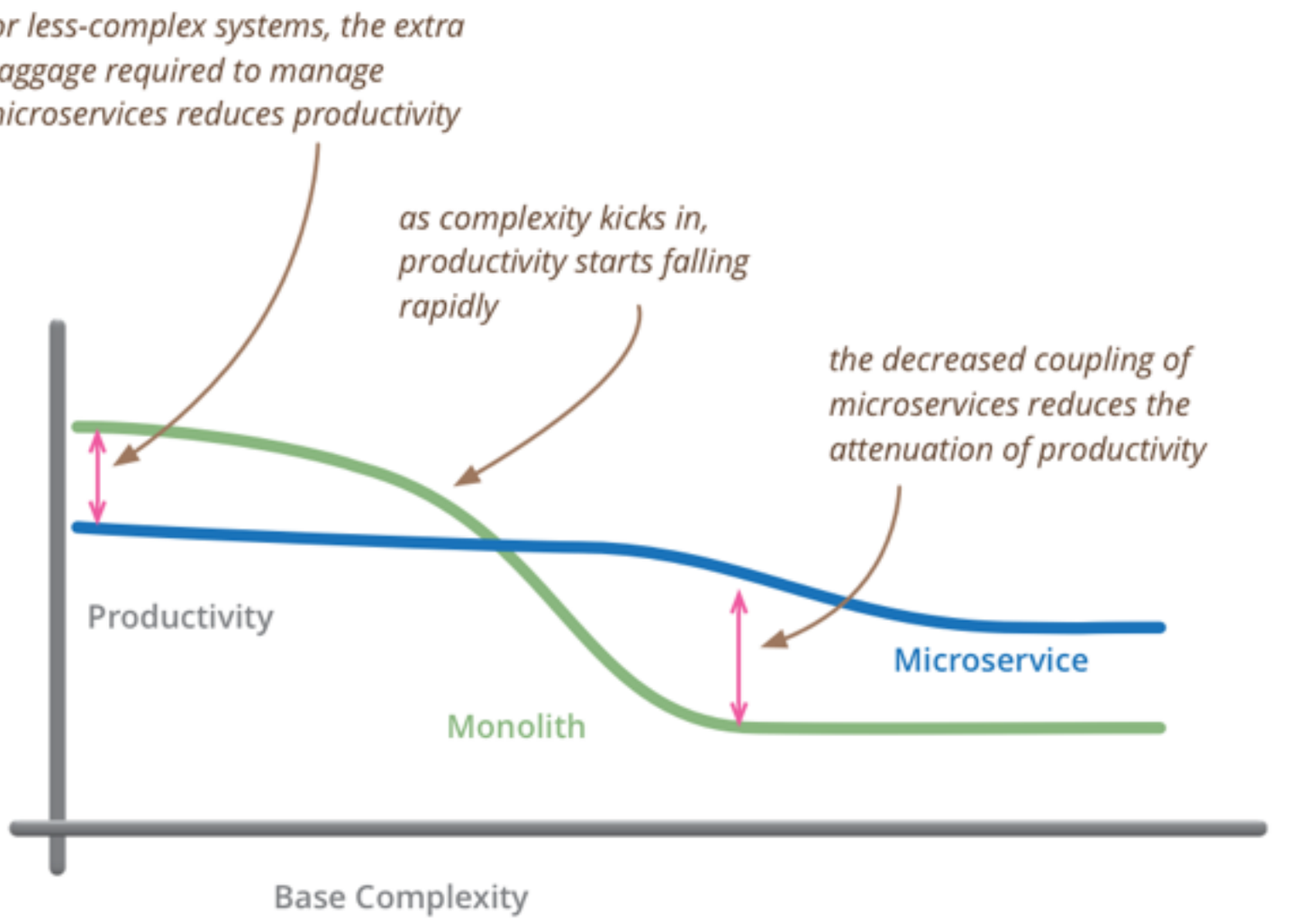
Drawbacks of microservices

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

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"

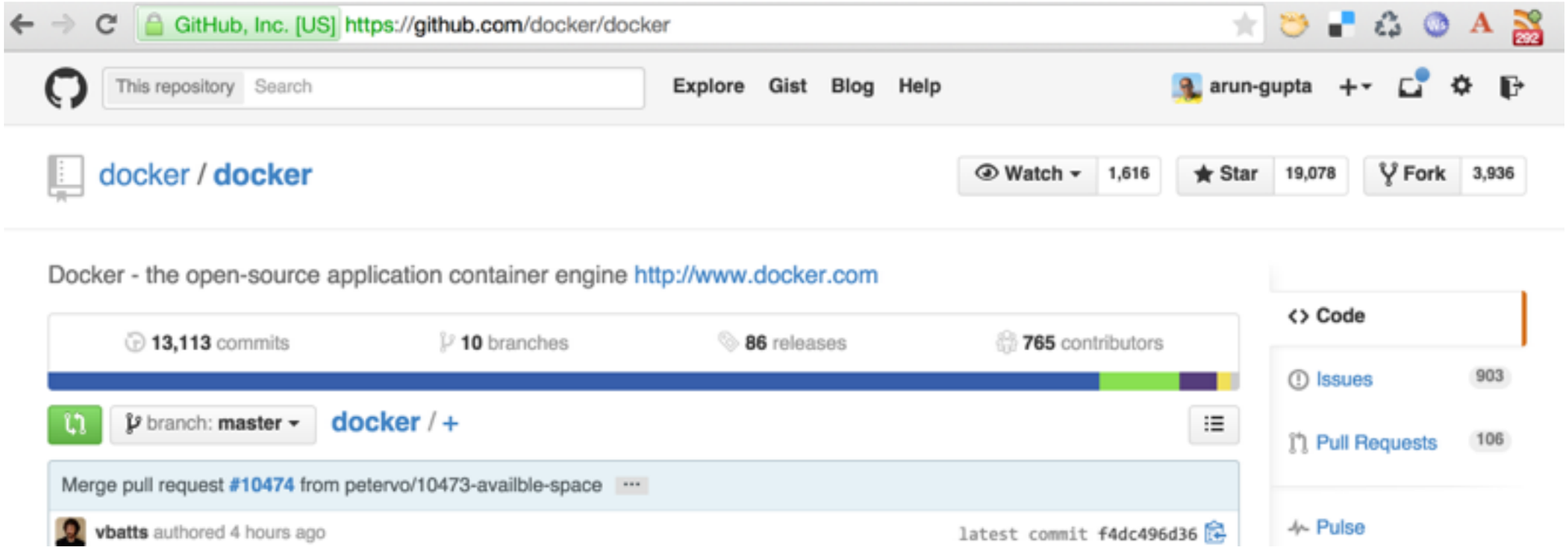
http://martinfowler.com/bliki/MicroservicePremium.html

Couchbase

What is Docker?

What is Docker?

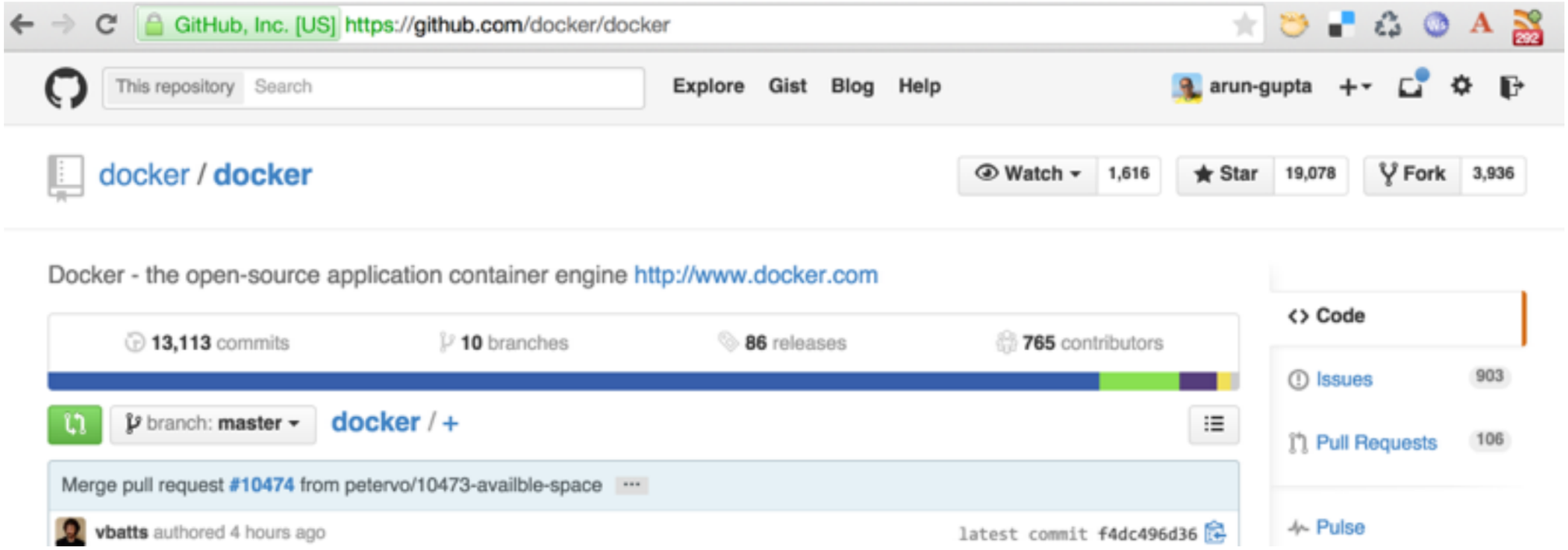
- Open source project and company



Couchbase

What is Docker?

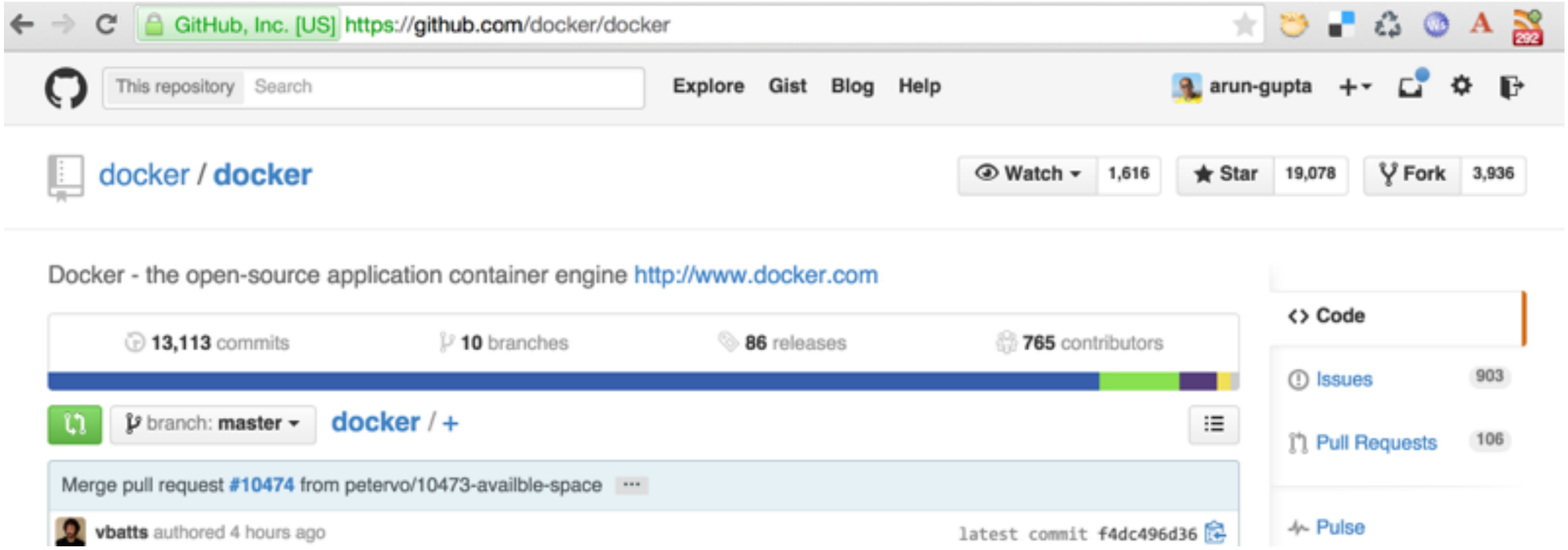
- Open source project and company



- Used to create containers for software applications

What is Docker?

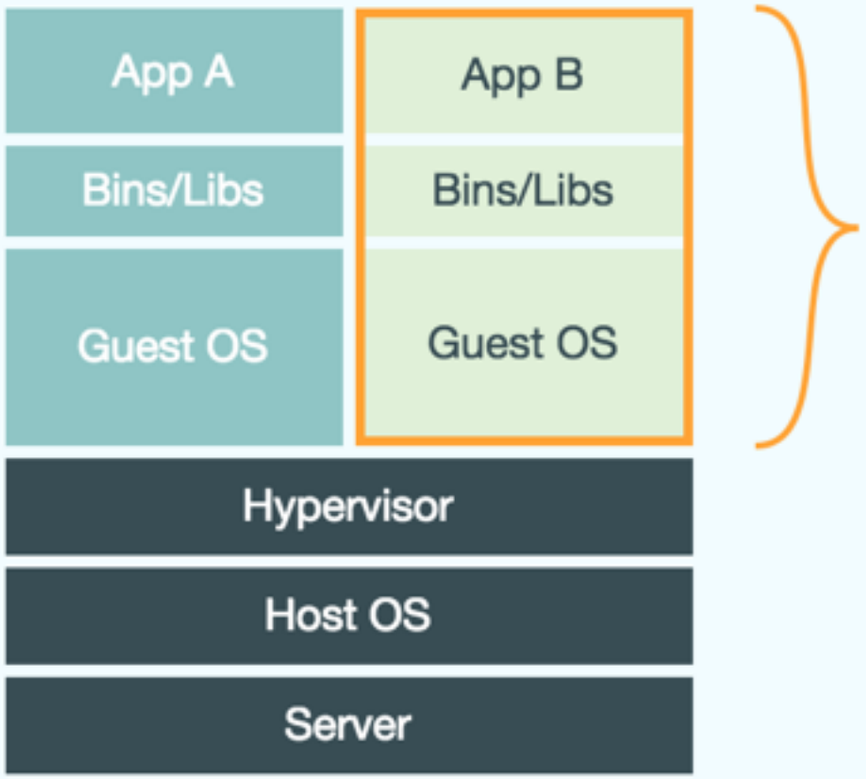
- Open source project and company



- Used to create containers for software applications

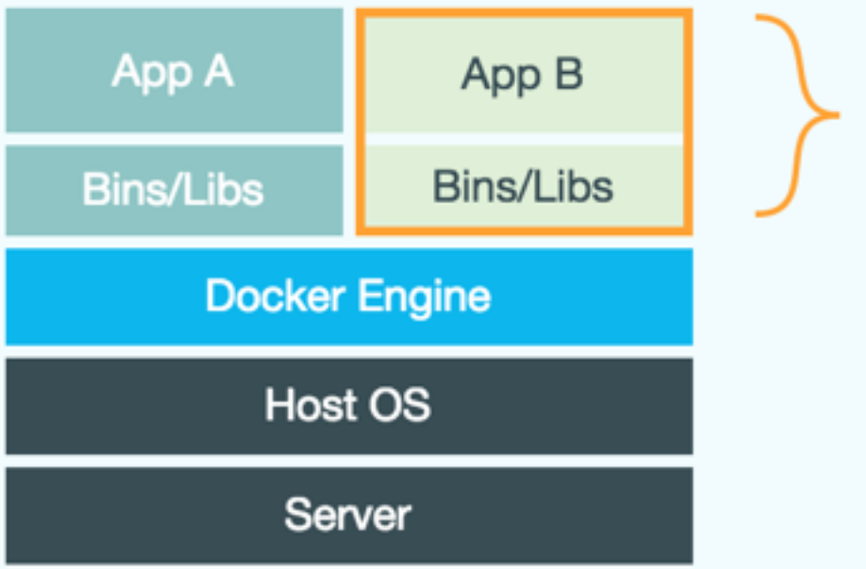
- Package Once Deploy Anywhere (PODA)

Couchbase



Virtual Machines

Each virtualized application includes not only the application - which may be only 10s of MB - and the necessary binaries and libraries, but also an entire guest operating system - which may weigh 10s of GB.



Docker

The Docker Engine container comprises just the application and its dependencies. It runs as an isolated process in userspace on the host operating system, sharing the kernel with other containers. Thus, it enjoys the resource isolation and allocation benefits of VMs but is much more portable and efficient.

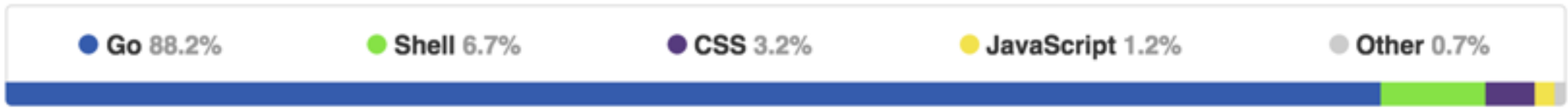
Underlying Technology

Underlying Technology

- Written in Go

Underlying Technology

- Written in Go



- Uses several Linux features

Underlying Technology

- Written in Go

- Uses several Linux features
- **Namespaces** to provide isolation

Underlying Technology

- Written in Go

- Uses several Linux features
  - **Namespaces** to provide isolation
  - **Control groups** to share/limit hardware resources

Underlying Technology

- Written in Go

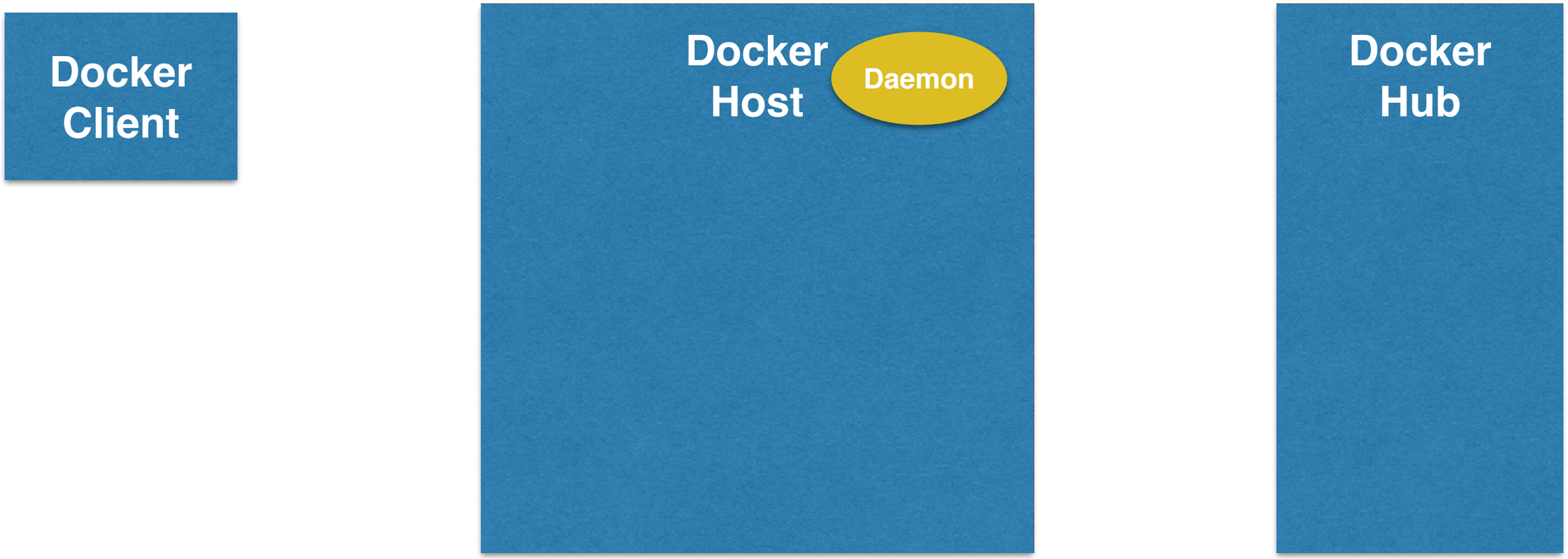
- Uses several Linux features
  - **Namespaces** to provide isolation
  - **Control groups** to share/limit hardware resources
  - **Union File System** makes it light and fast

Underlying Technology

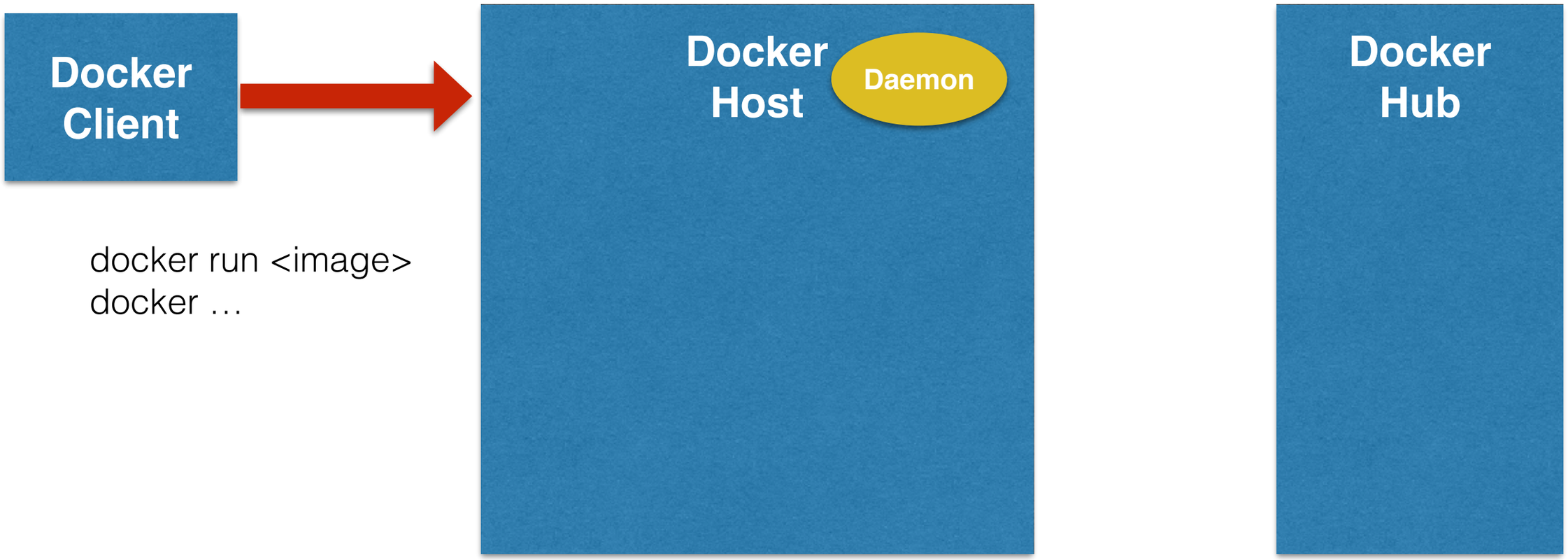
- Written in Go

- Uses several Linux features
  - **Namespaces** to provide isolation
  - **Control groups** to share/limit hardware resources
  - **Union File System** makes it light and fast
  - **libcontainer** defines container format

Docker Workflow

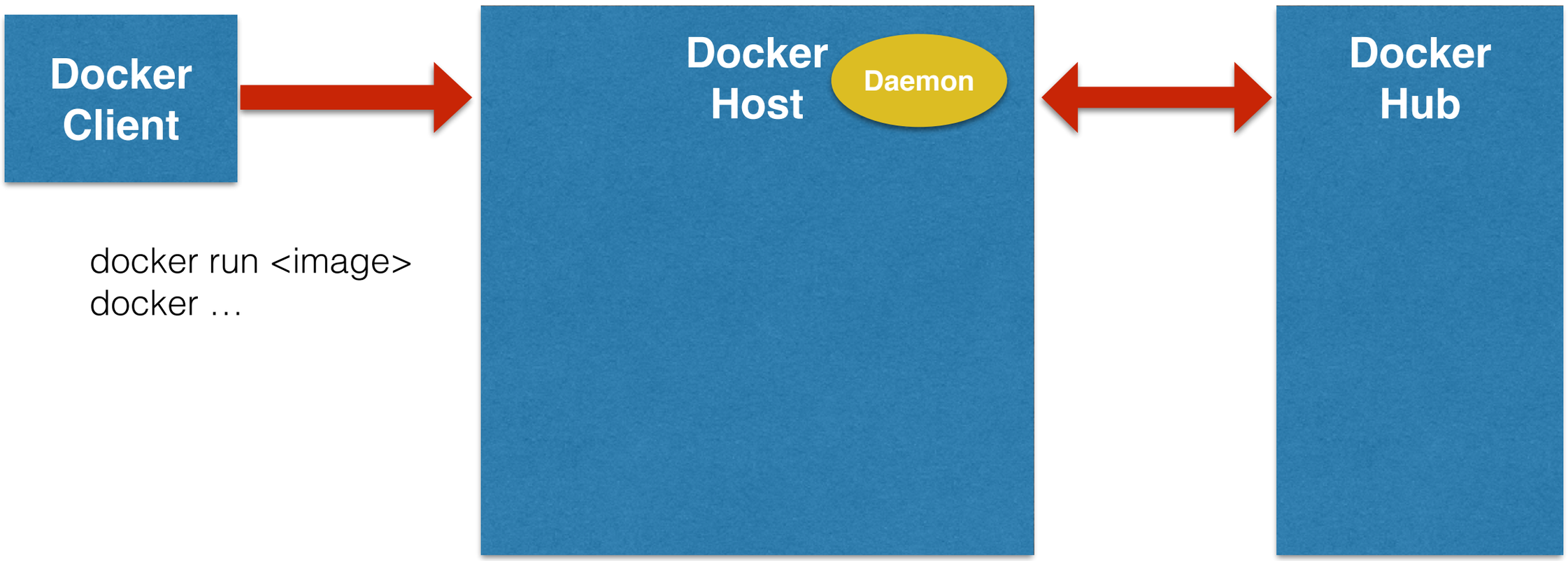


Docker Workflow



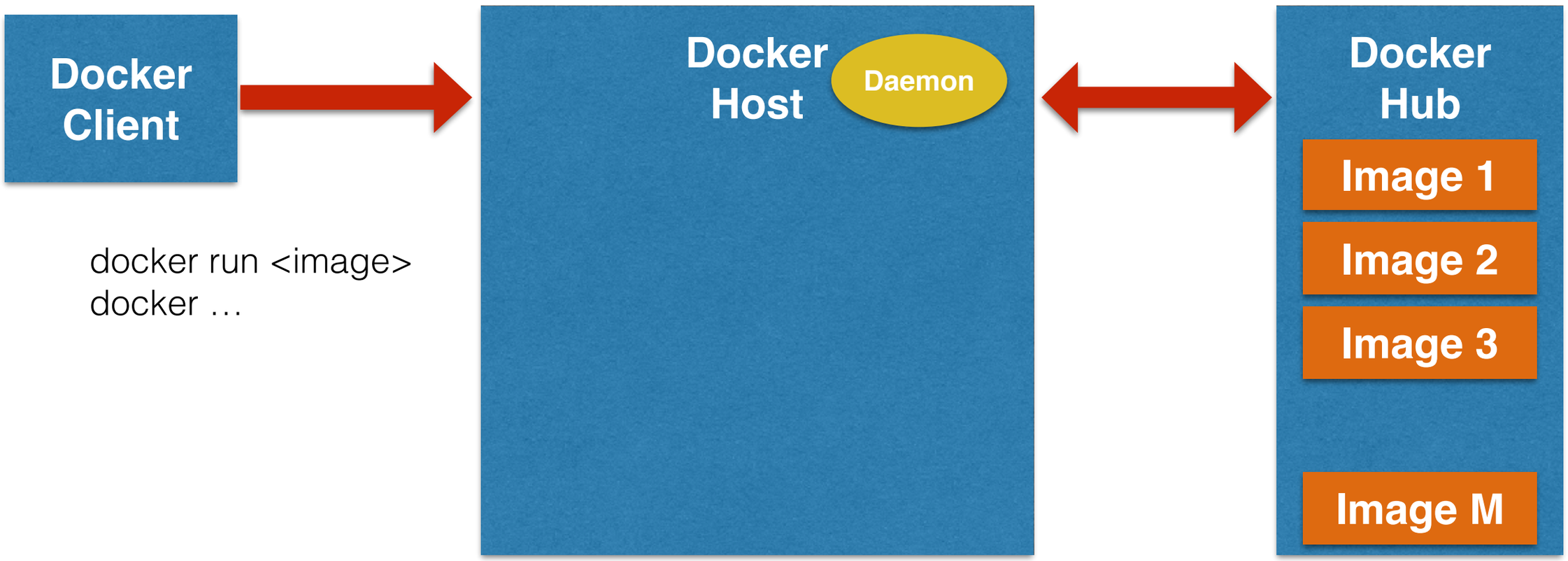
docker run <image>
docker …

Docker Workflow



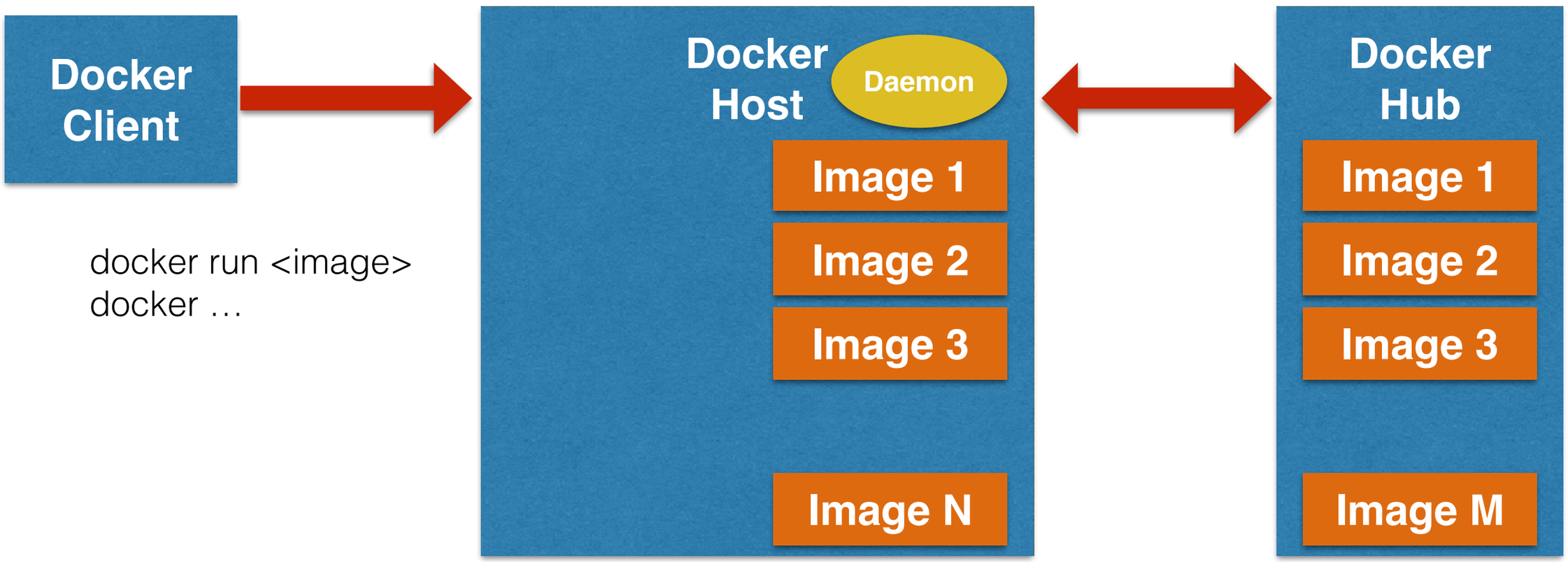
docker run <image>
docker …

Docker Workflow



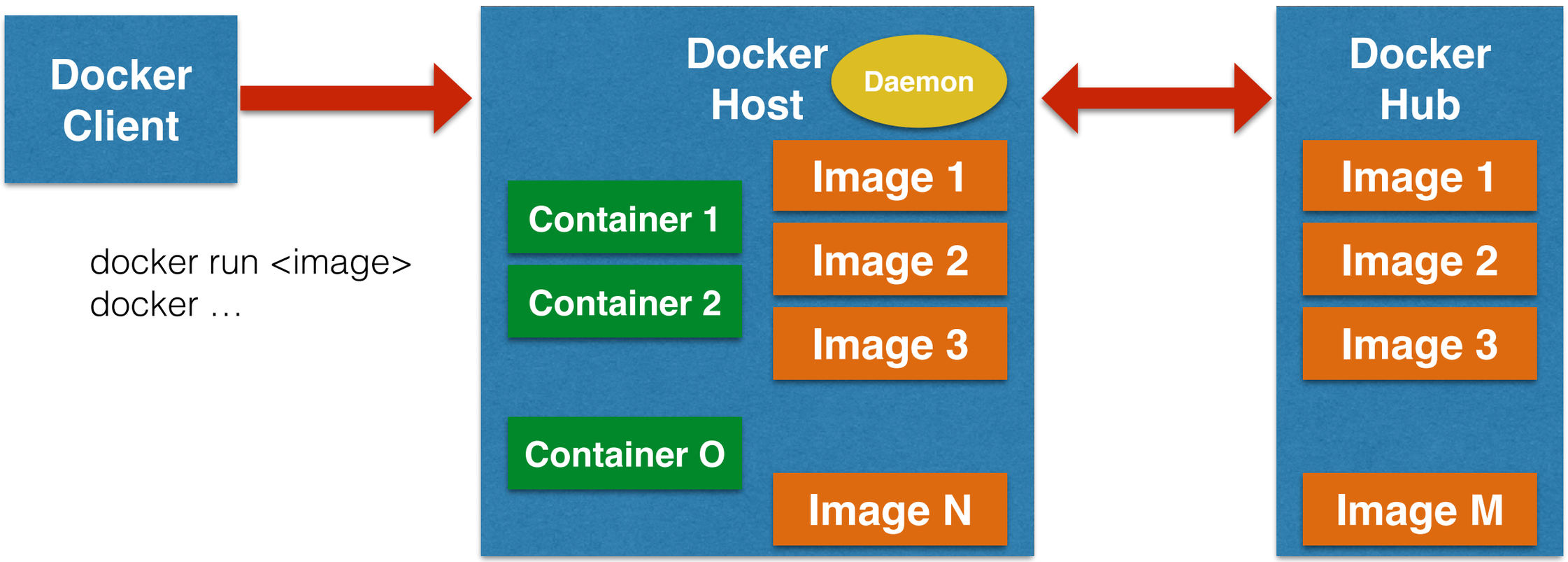
docker run <image>
docker …

Docker Workflow



docker run <image>
docker …

Docker Workflow



docker run <image>
docker …

Build

Develop an app using Docker containers with any language and any toolchain.

- Image defined in text-based **Dockerfile**

- Image defined in text-based **Dockerfile**
- List of commands to build the image

- Image defined in text-based **Dockerfile**
- List of commands to build the image

Build

Develop an app using Docker containers with any language and any toolchain.

```
FROM fedora:latest

CMD echo “Hello world”
```

- Image defined in text-based **Dockerfile**
- List of commands to build the image


```
FROM fedora:latest

CMD echo “Hello world”
```

```
FROM jboss/wildfly

RUN curl -L https://github.com/javaee-samples/javaee7-hol/raw/master/solution/
movieplex7-1.0-SNAPSHOT.war -o /opt/jboss/wildfly/standalone/deployments/
movieplex7-1.0-SNAPSHOT.war
```

Kubernetes

Kubernetes

- Open source orchestration system for Docker containers

Kubernetes

- Open source orchestration system for Docker containers

- Provide declarative primitives for the “desired state”

- Self-healing

- Auto-restarting

- Schedule across hosts

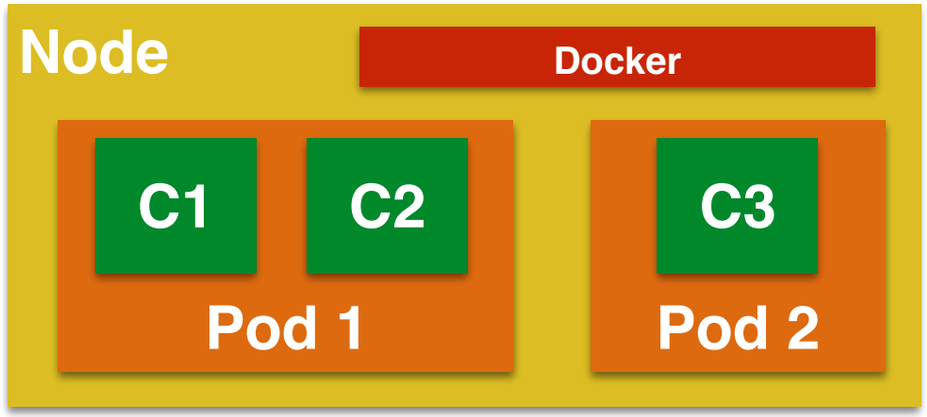
- Replicating

Couchbase

Concepts

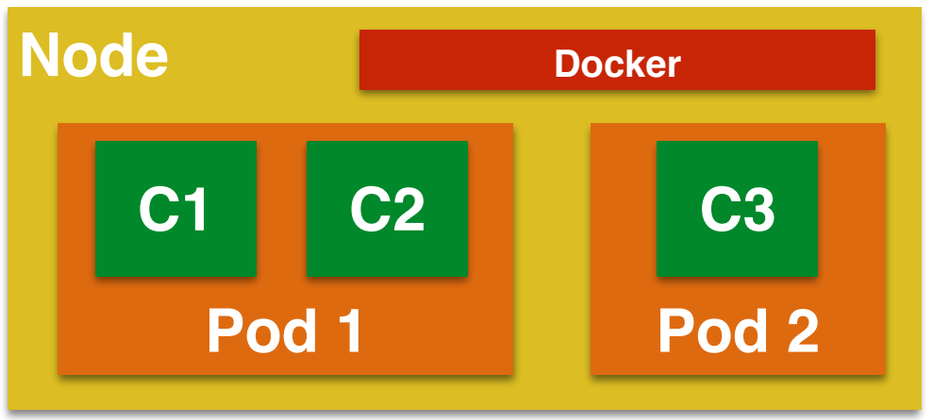
Concepts

- **Pods**: collocated group of Docker containers that share an IP and storage volume

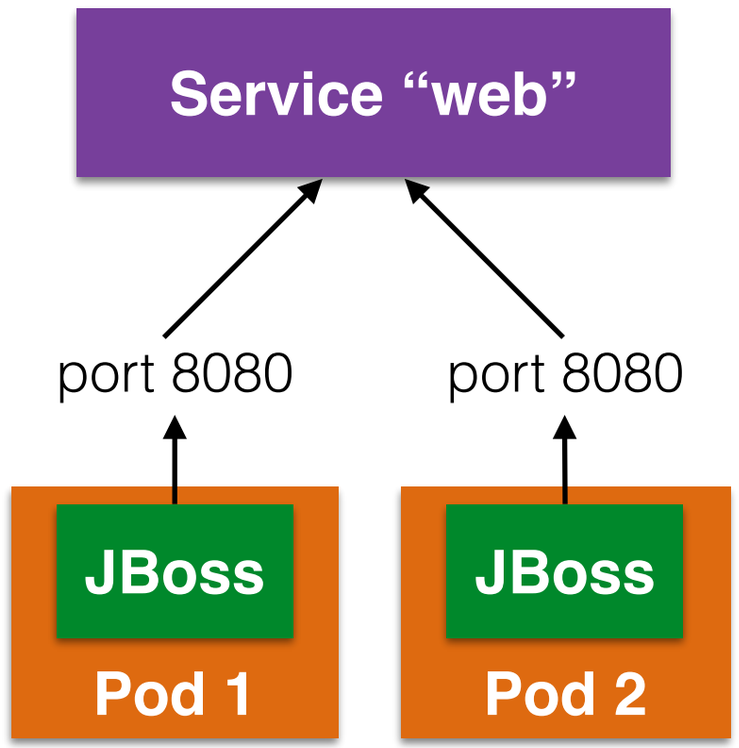


Concepts

- **Pods**: collocated group of Docker containers that share an IP and storage volume

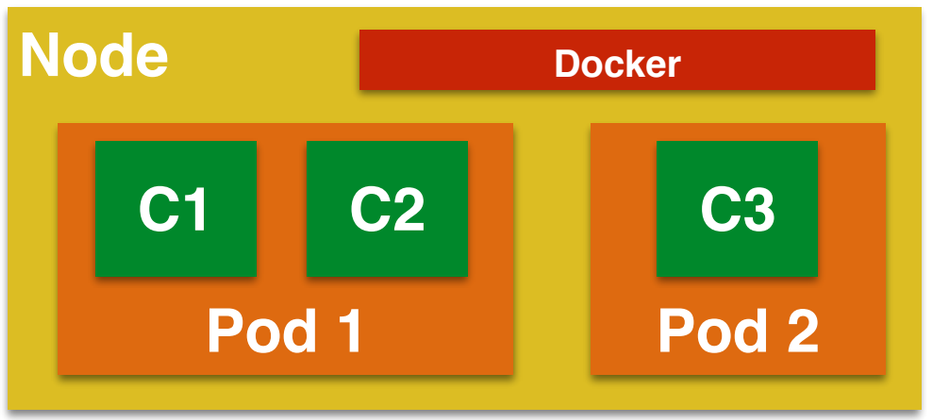


- **Service**: Single, stable name for a set of pods, also acts as LB



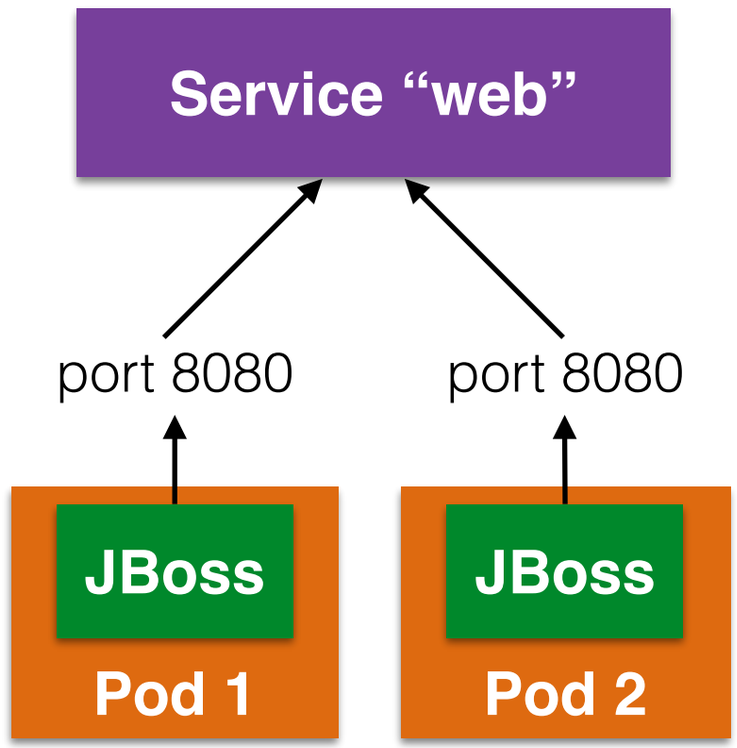
Concepts

- **Pods**: collocated group of Docker containers that share an IP and storage volume



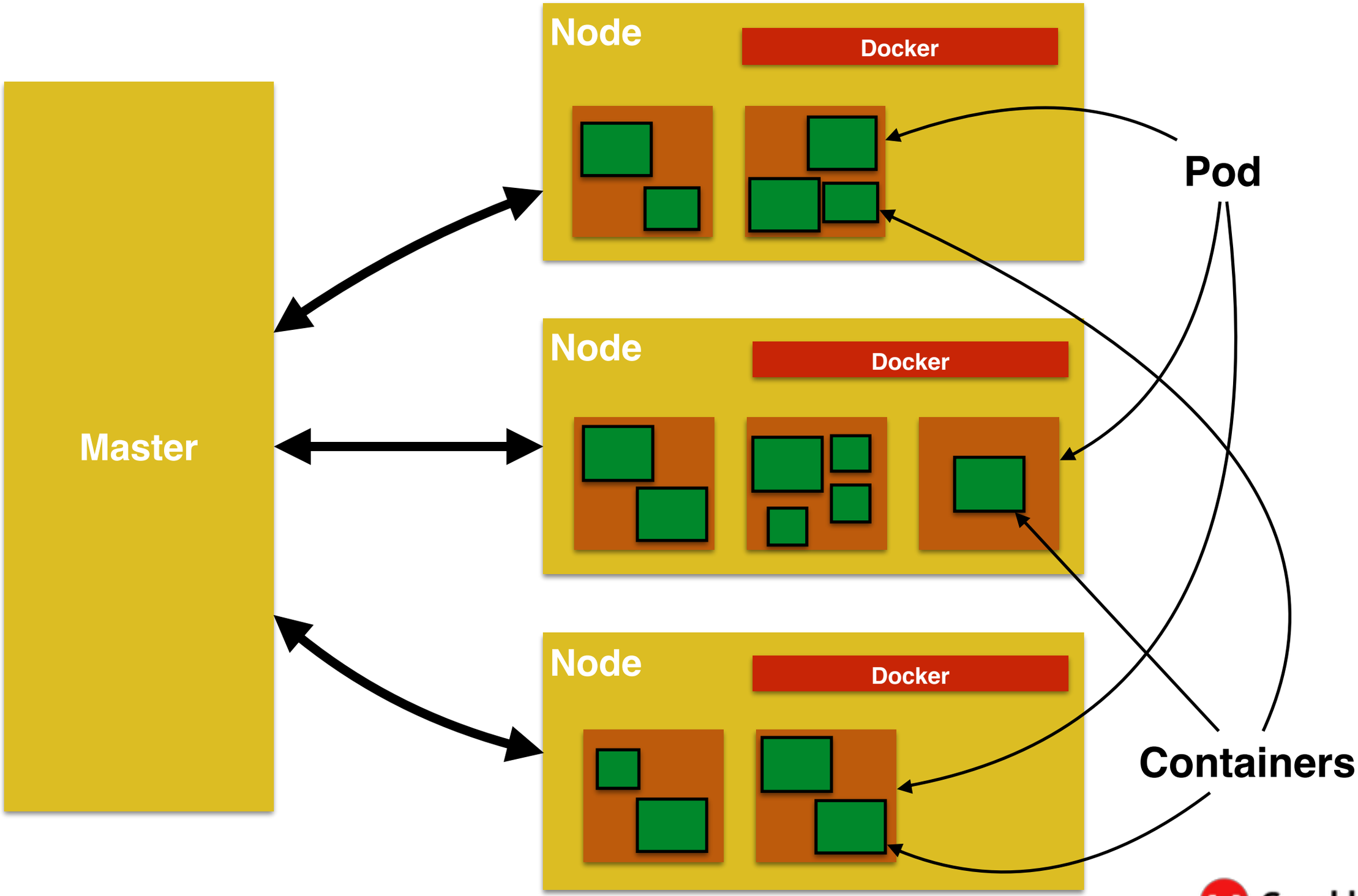
- **Service**: Single, stable name for a set of pods, also acts as LB

- **Label**: used to organize and select group of objects



Couchbase

Kubernetes Architecture



Couchbase

Couchbase

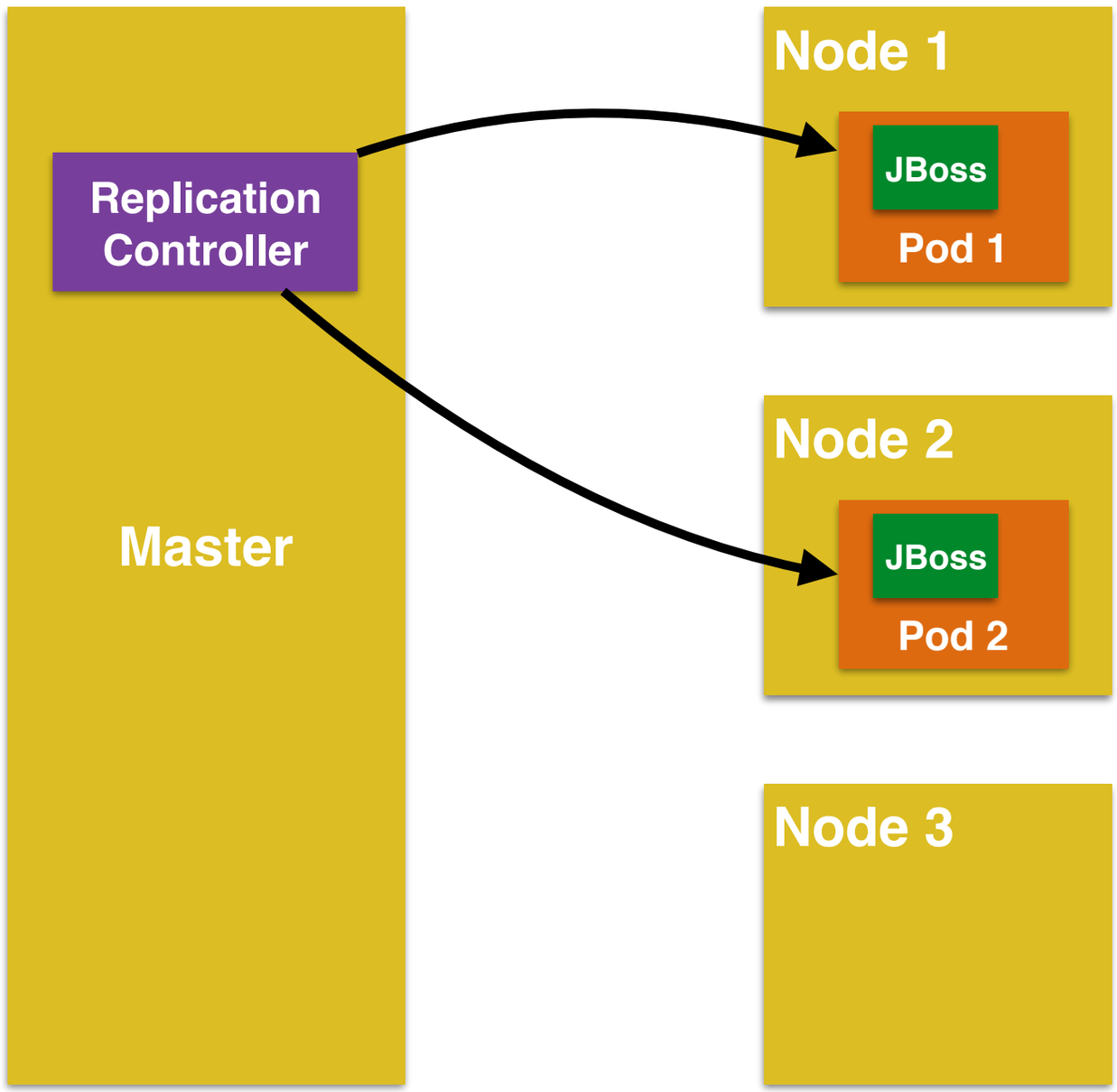
Replication Controller

- Ensures that a specified number of pod "replicas" are running at any one time

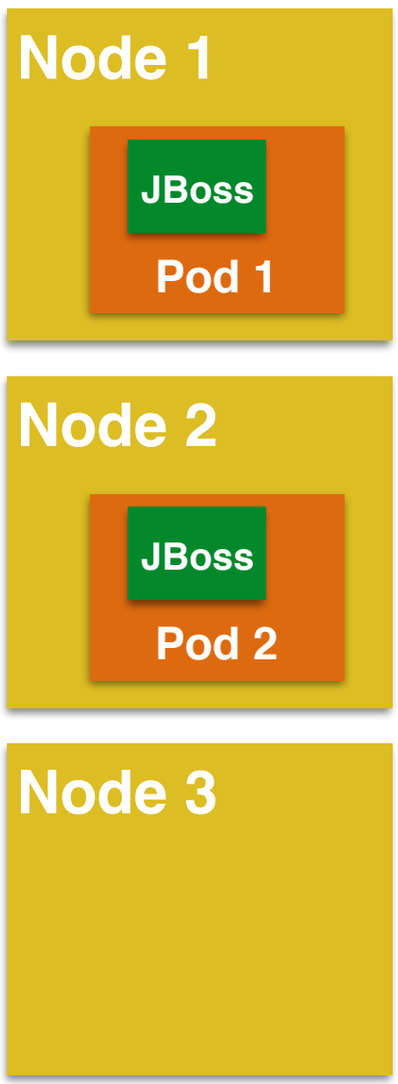
- Recommended to wrap a Pod in a RC

- Only appropriate for Pods with `Restart=Always` policy (default)

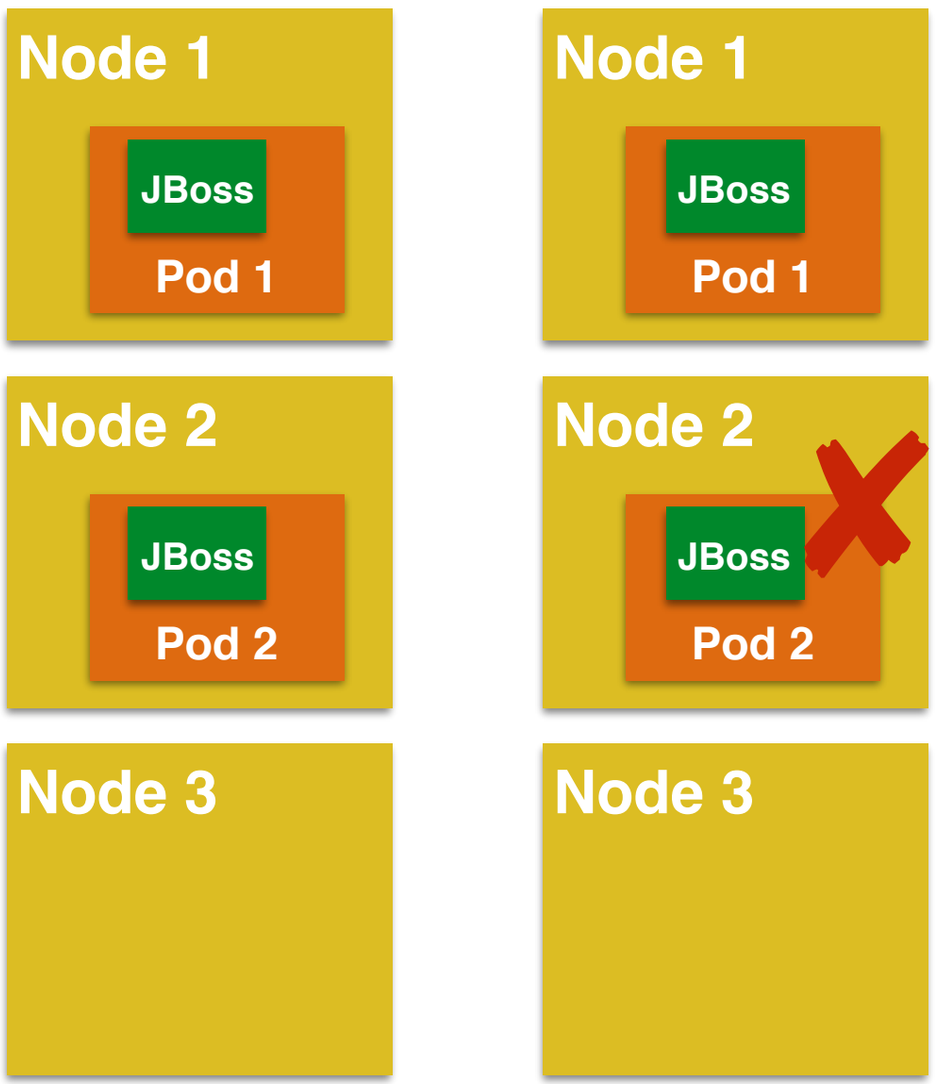
Replication Controller



Replication Controller: Automatic Rescheduling

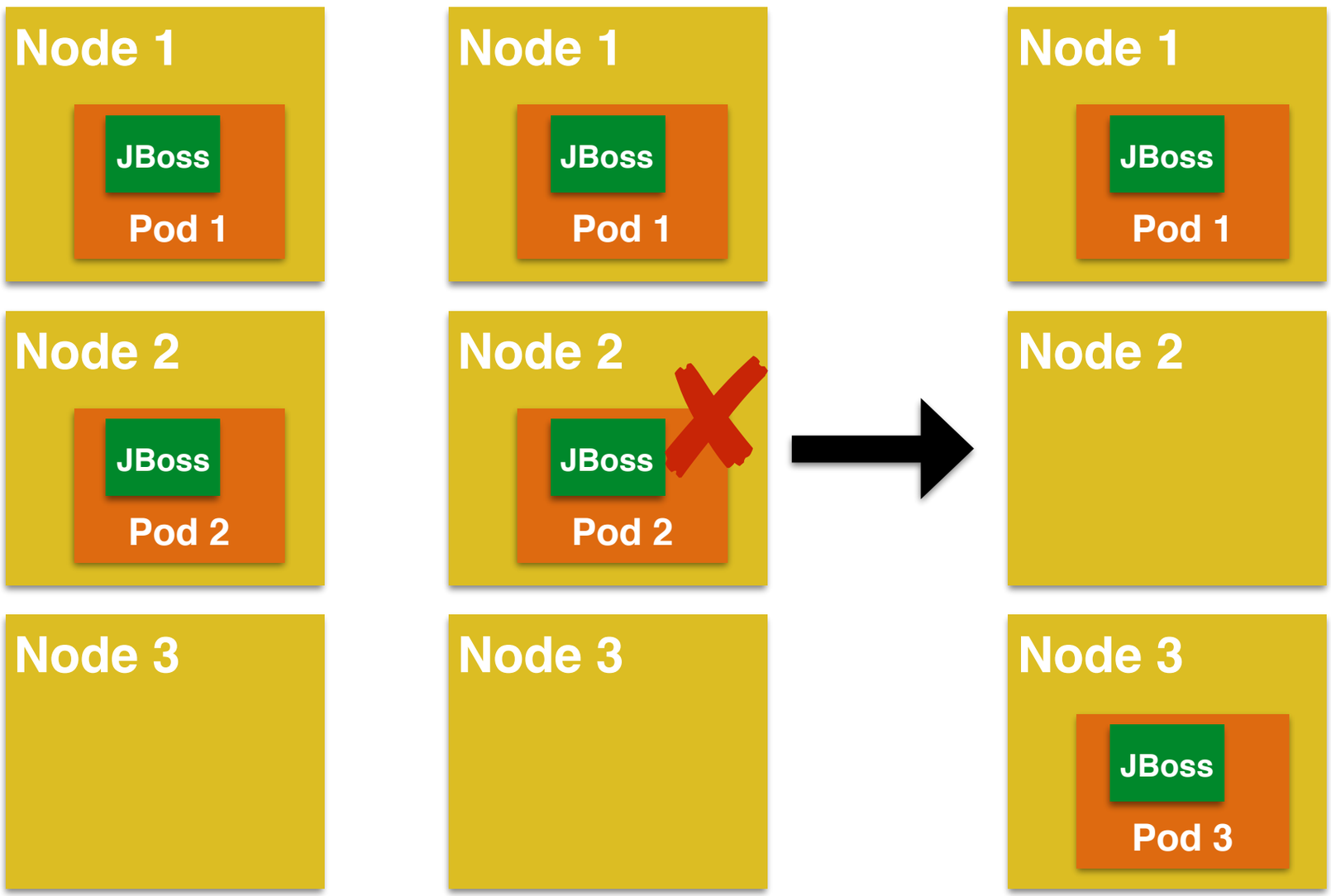


Replication Controller: Automatic Rescheduling

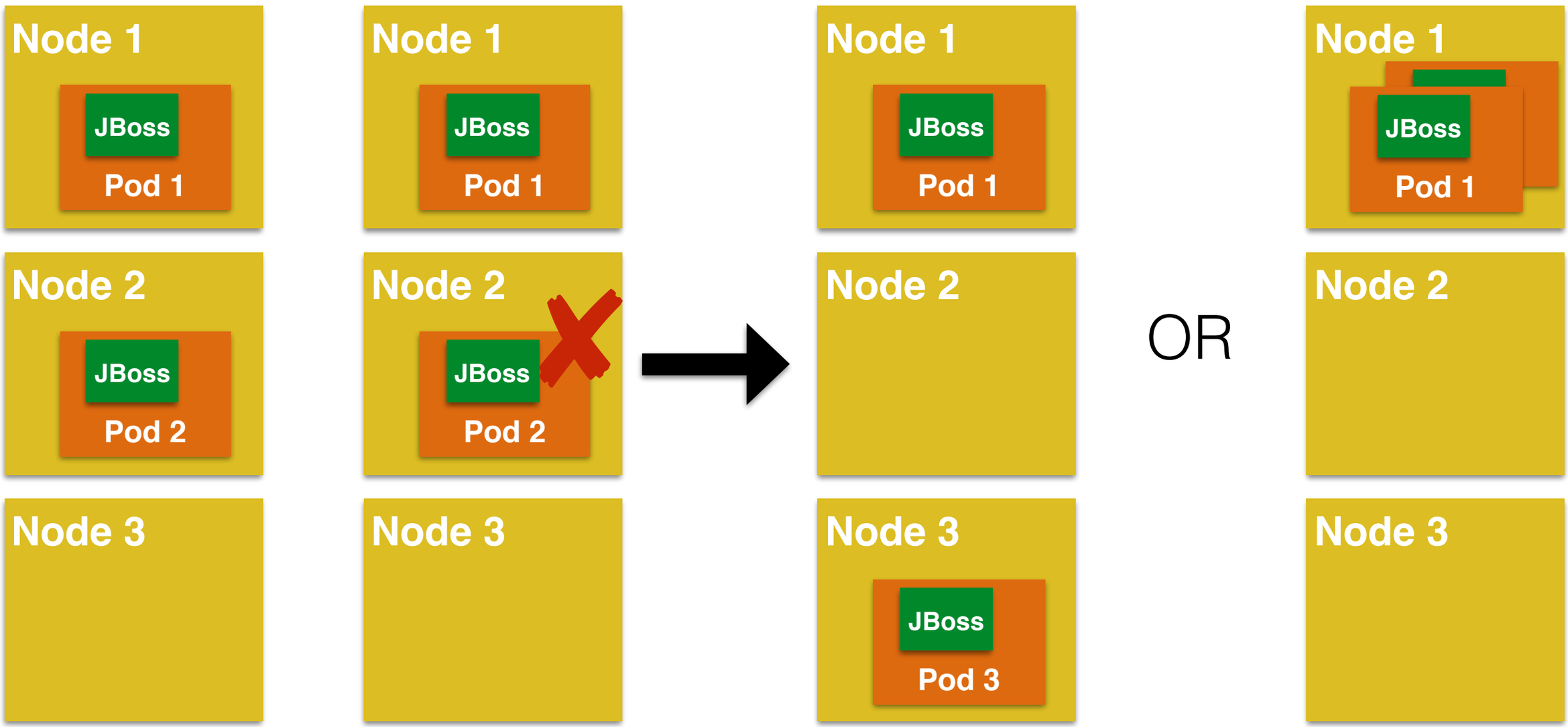


Couchbase

Replication Controller: Automatic Rescheduling

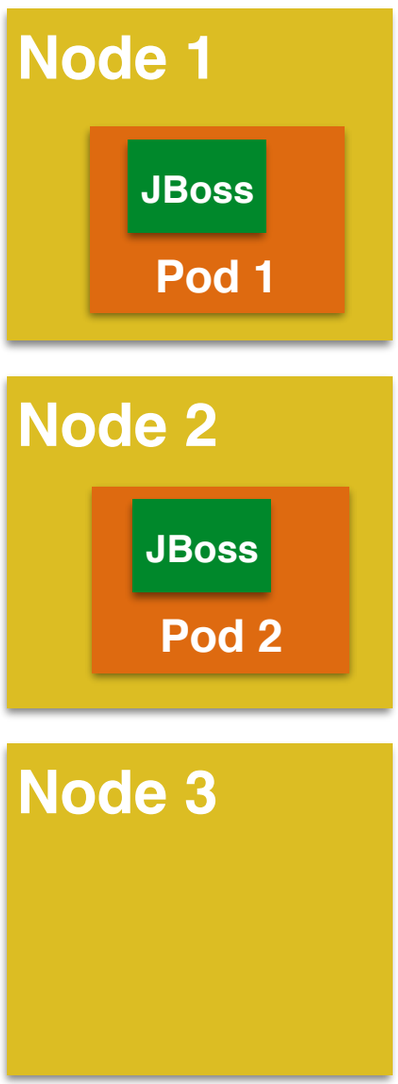


Replication Controller: Automatic Rescheduling



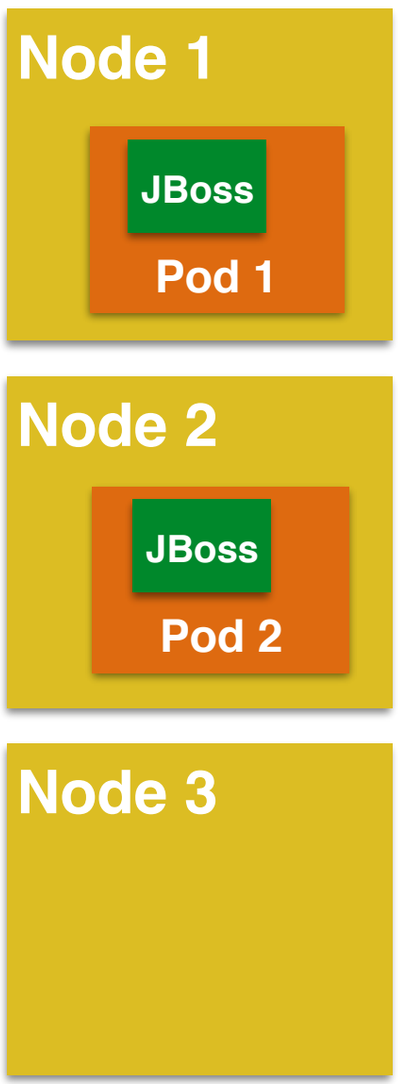
Couchbase

Replication Controller: Scaling

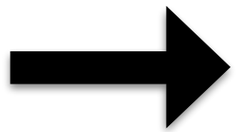


Couchbase

Replication Controller: Scaling

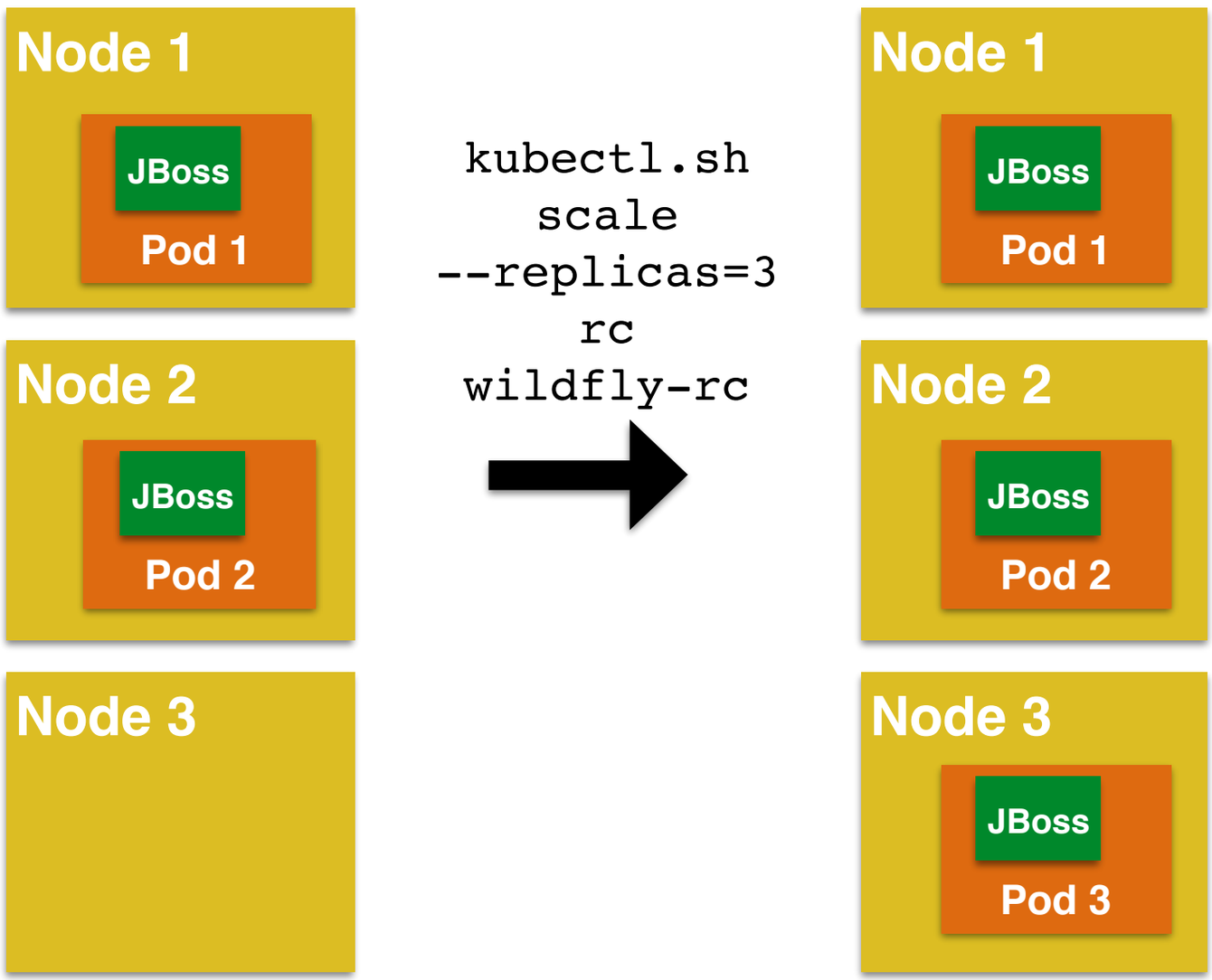



```
kubectl.sh
   scale
--replicas=3
   rc
wildfly-rc
```

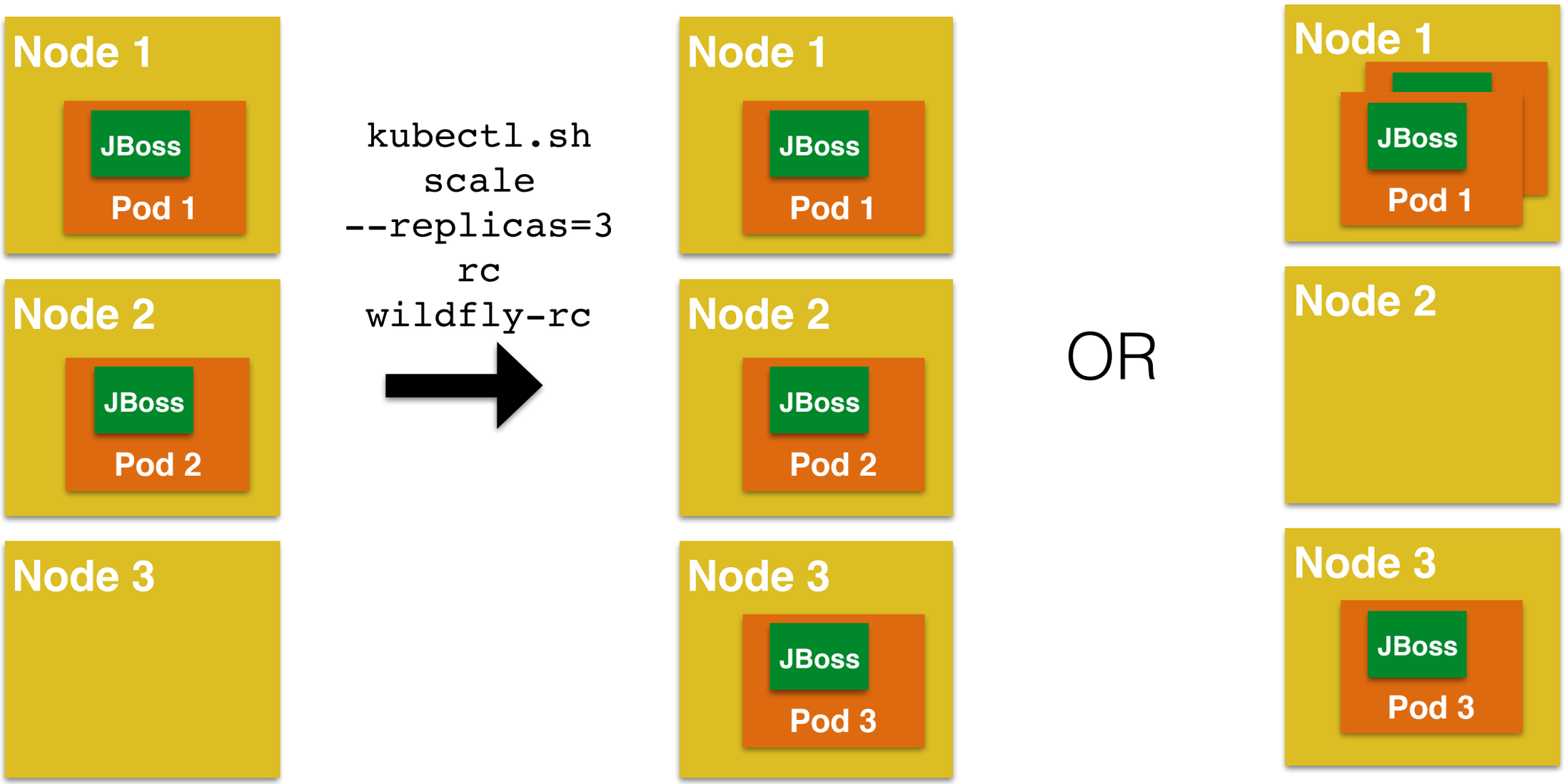


Couchbase

Replication Controller: Scaling



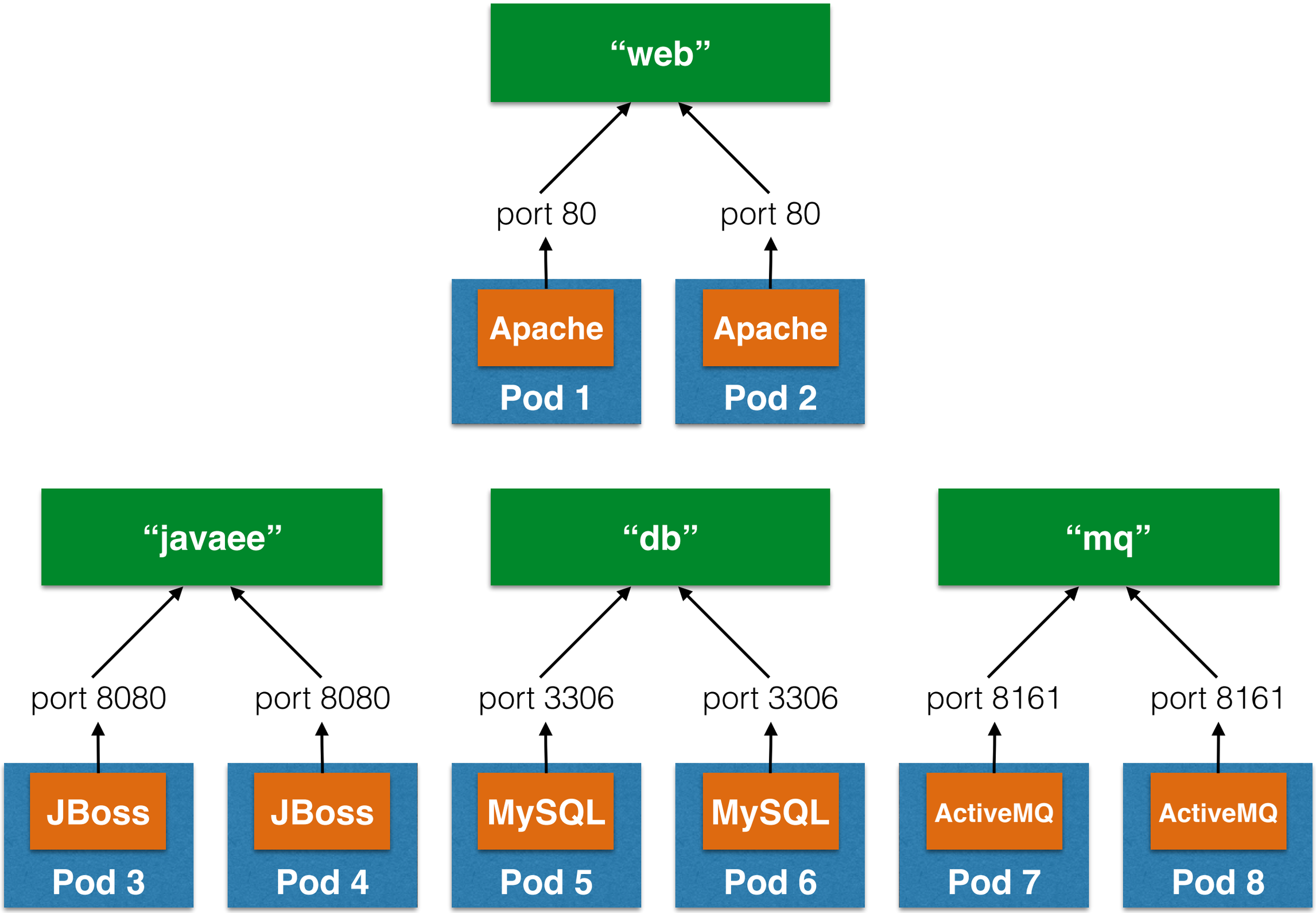
Replication Controller: Scaling



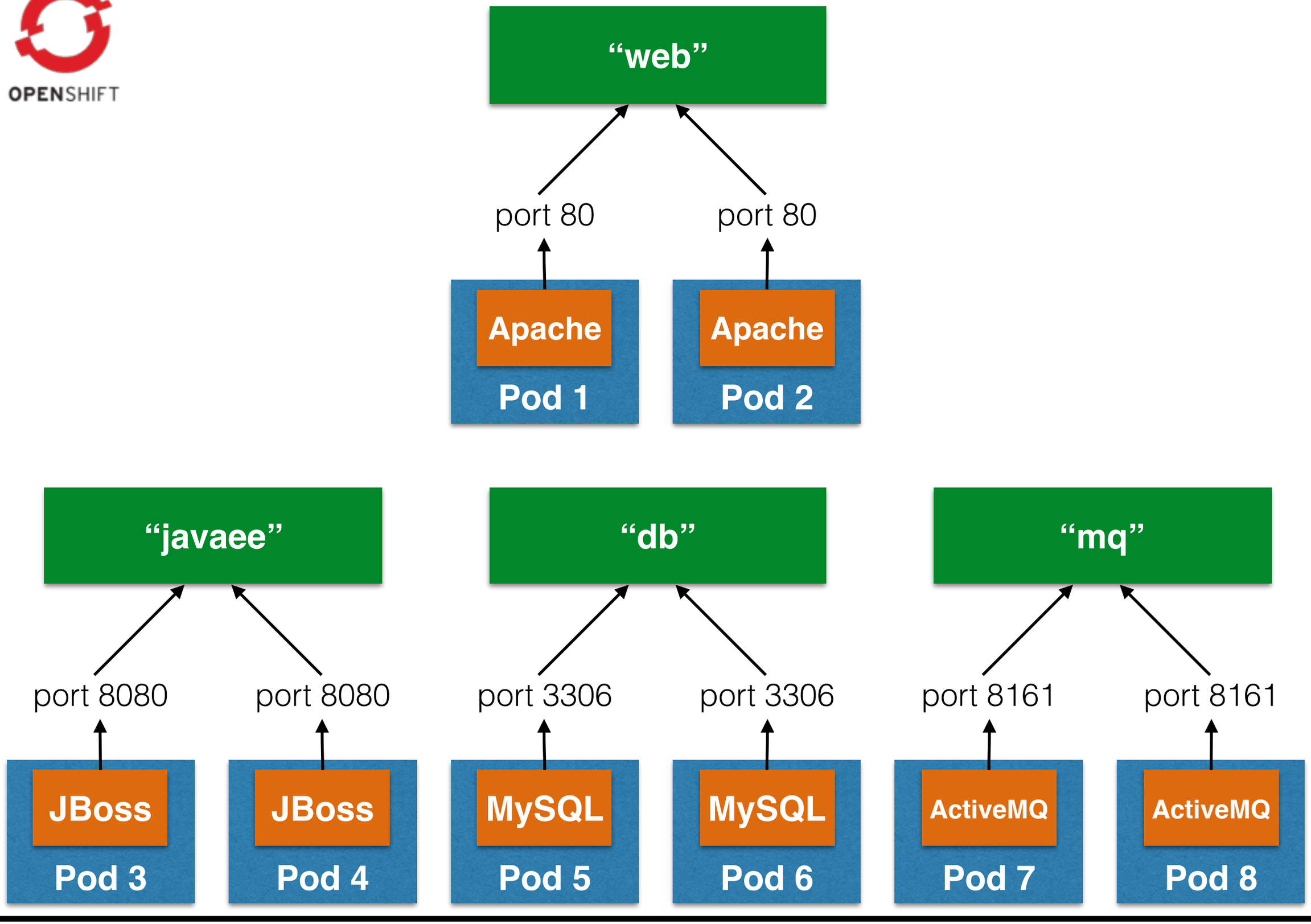
Kubernetes Config

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: wildfly-pod
  labels:
    name: wildfly
spec:
  containers:
    - image: jboss/wildfly
      name: wildfly-pod
      ports:
        - containerPort: 8080
```

```yaml
apiVersion: v1
kind: ReplicationController
metadata:
  name: wildfly-rc
  labels:
    name: wildfly
spec:
  replicas: 2
  template:
    metadata:
      labels:
        name: wildfly
    spec:
      containers:
      - name: wildfly-rc-pod
        image: jboss/wildfly
        ports:
        - containerPort: 8080
```


“web”
port 80
port 80
Apache
Pod 1
Apache
Pod 2
“javaee”
“db”
“mq”
port 8080
port 8080
port 3306
port 3306
port 8161
port 8161
JBoss
Pod 3
JBoss
Pod 4
MySQL
Pod 5
MySQL
Pod 6
ActiveMQ
Pod 7
ActiveMQ
Pod 8

OPENSHIFT



Couchbase

References

- github.com/arun-gupta/microservices

- github.com/javaee-samples/docker-java