redhat.

RED HAT® JBOSS®
MIDDLEWARE

Refactoring your Java EE applications using Microservices and Containers

Arun Gupta, Red Hat

Arun Gupta

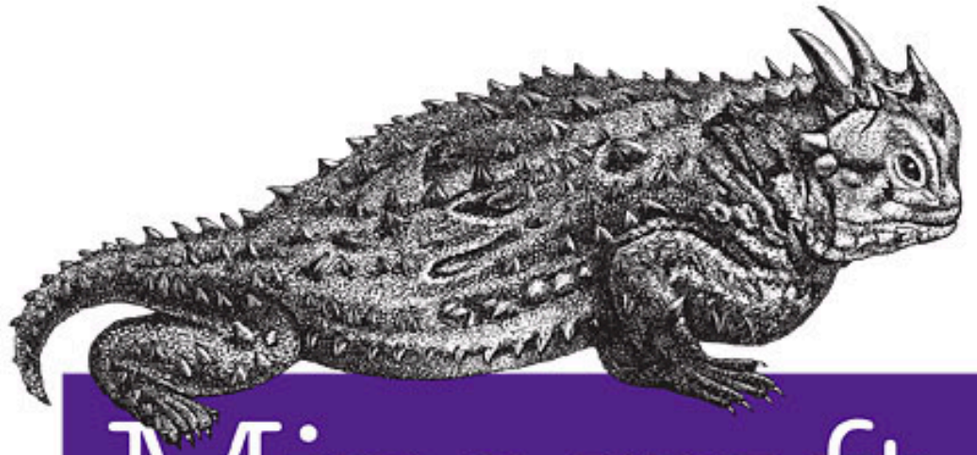
Director, Technical Marketing & Developer Advocacy

@arungupta
blog.arungupta.me
arungupta@redhat.com

redhat

DEVOXX™ 4KIDS USA

O'REILLY®

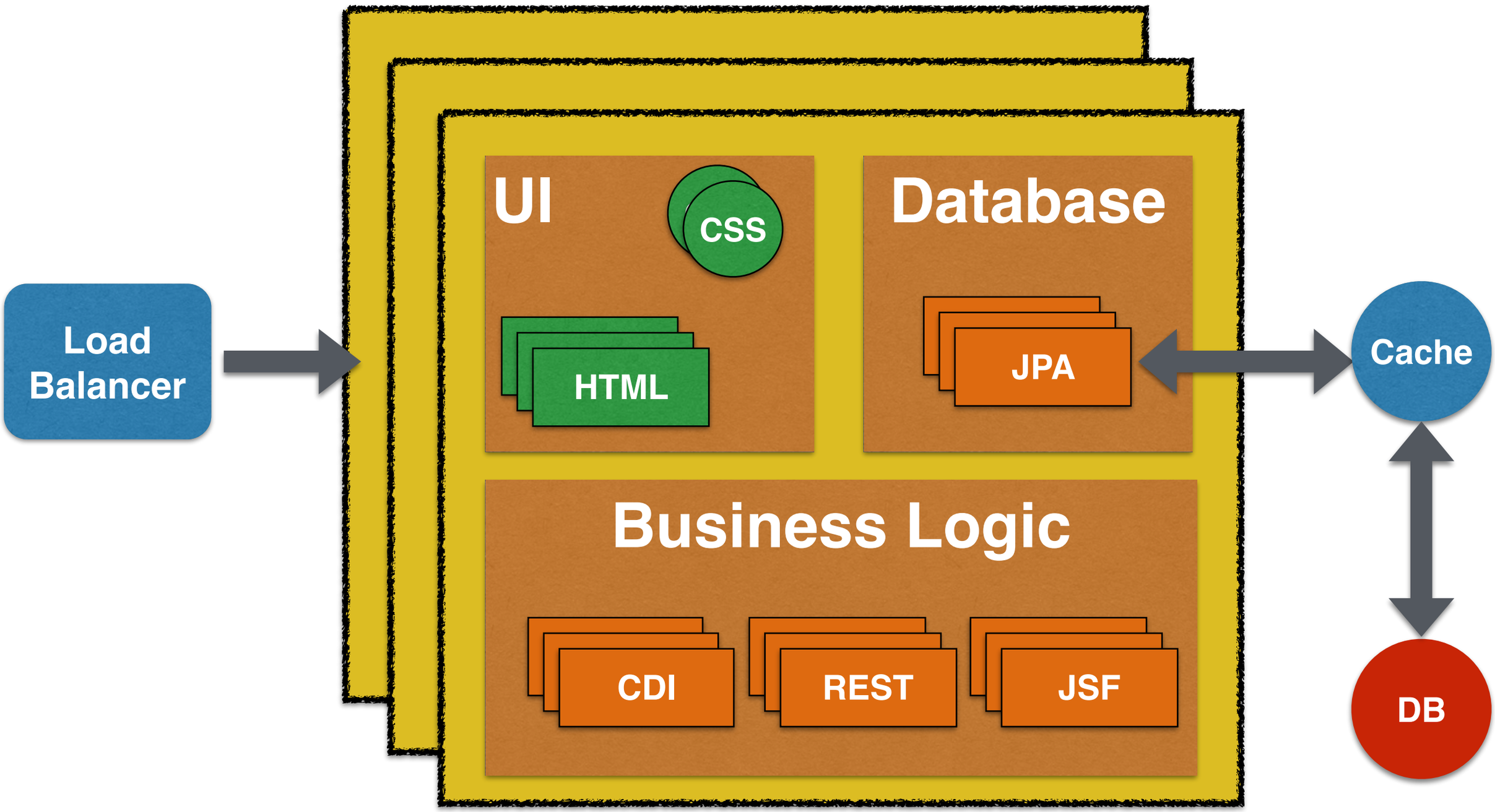


Minecraft Modding with Forge

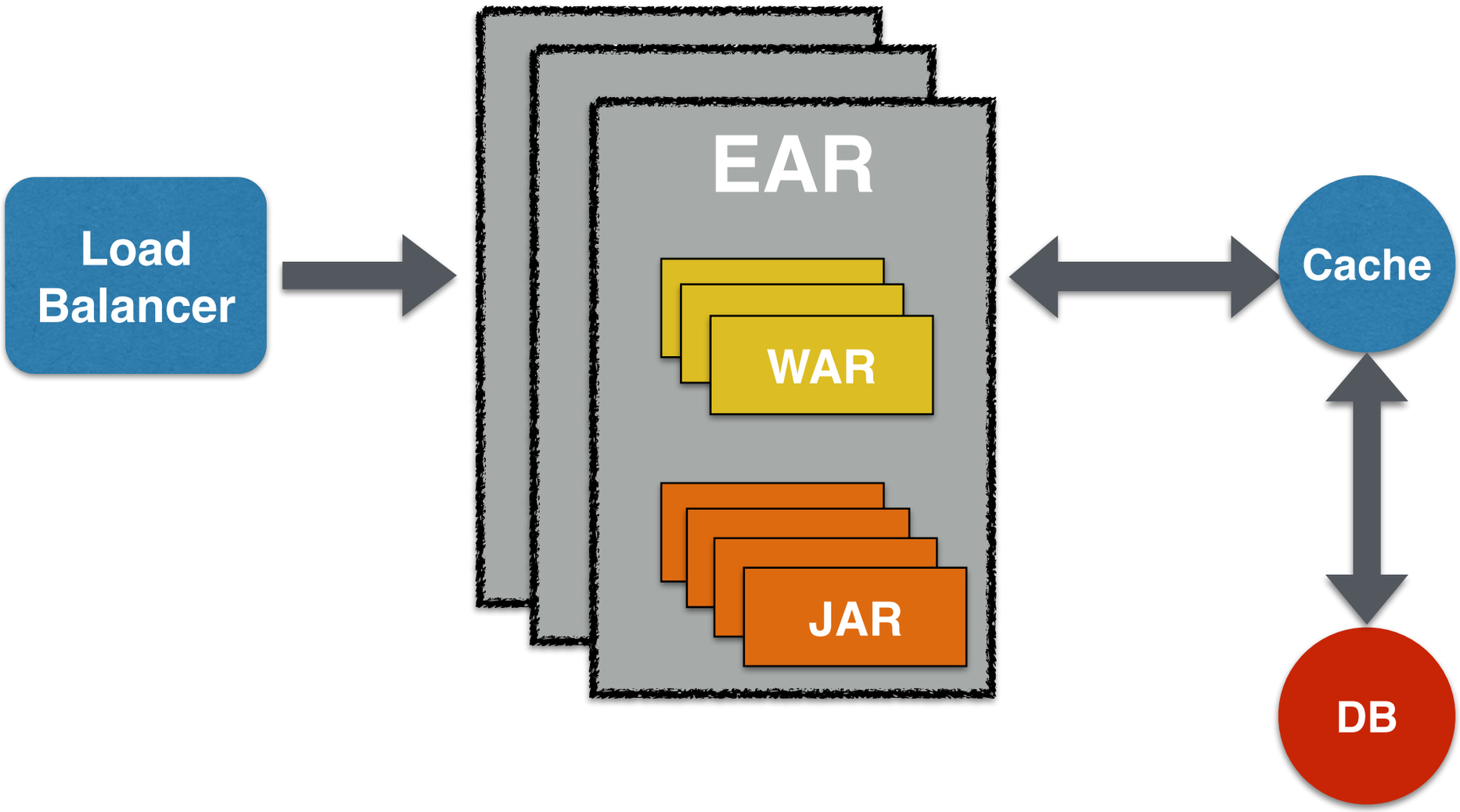
A FAMILY-FRIENDLY GUIDE TO BUILDING FUN MODS IN JAVA

Arun Gupta & Aditya Gupta

Monolith Application



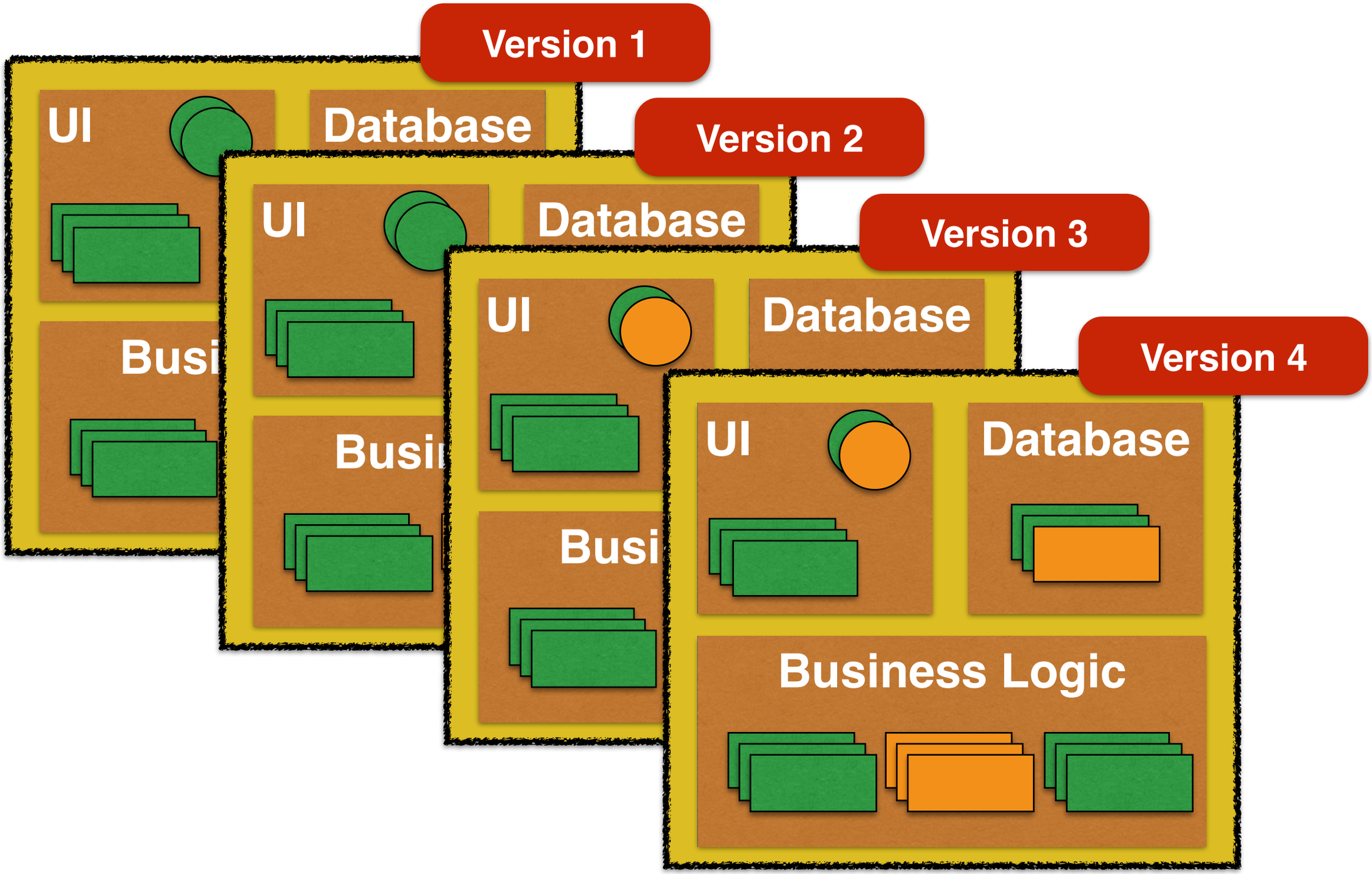
Monolith Application



Advantages of Monolith Application

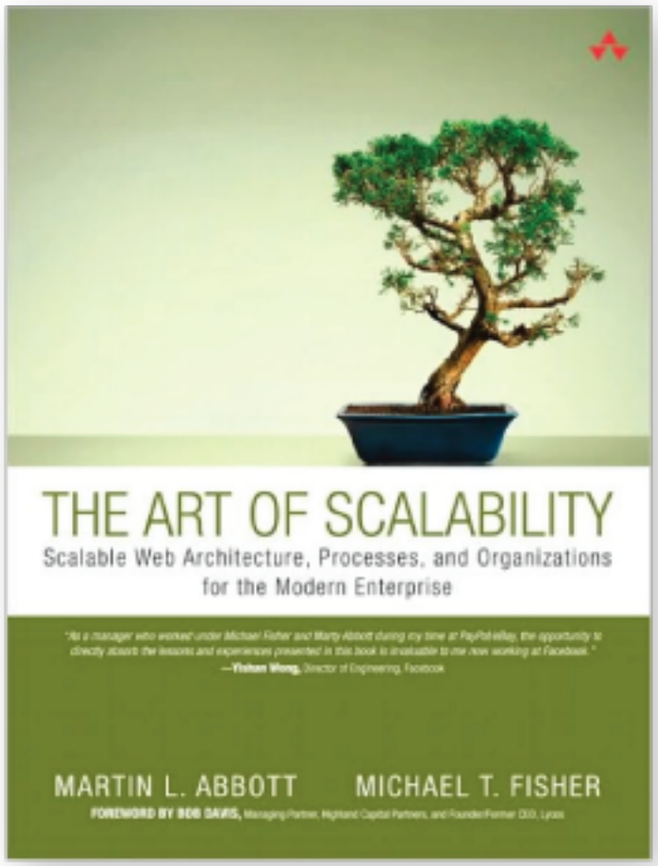
- Typically packaged in a single `.ear`
- Easy to test (all required services are up)
- Simple to develop

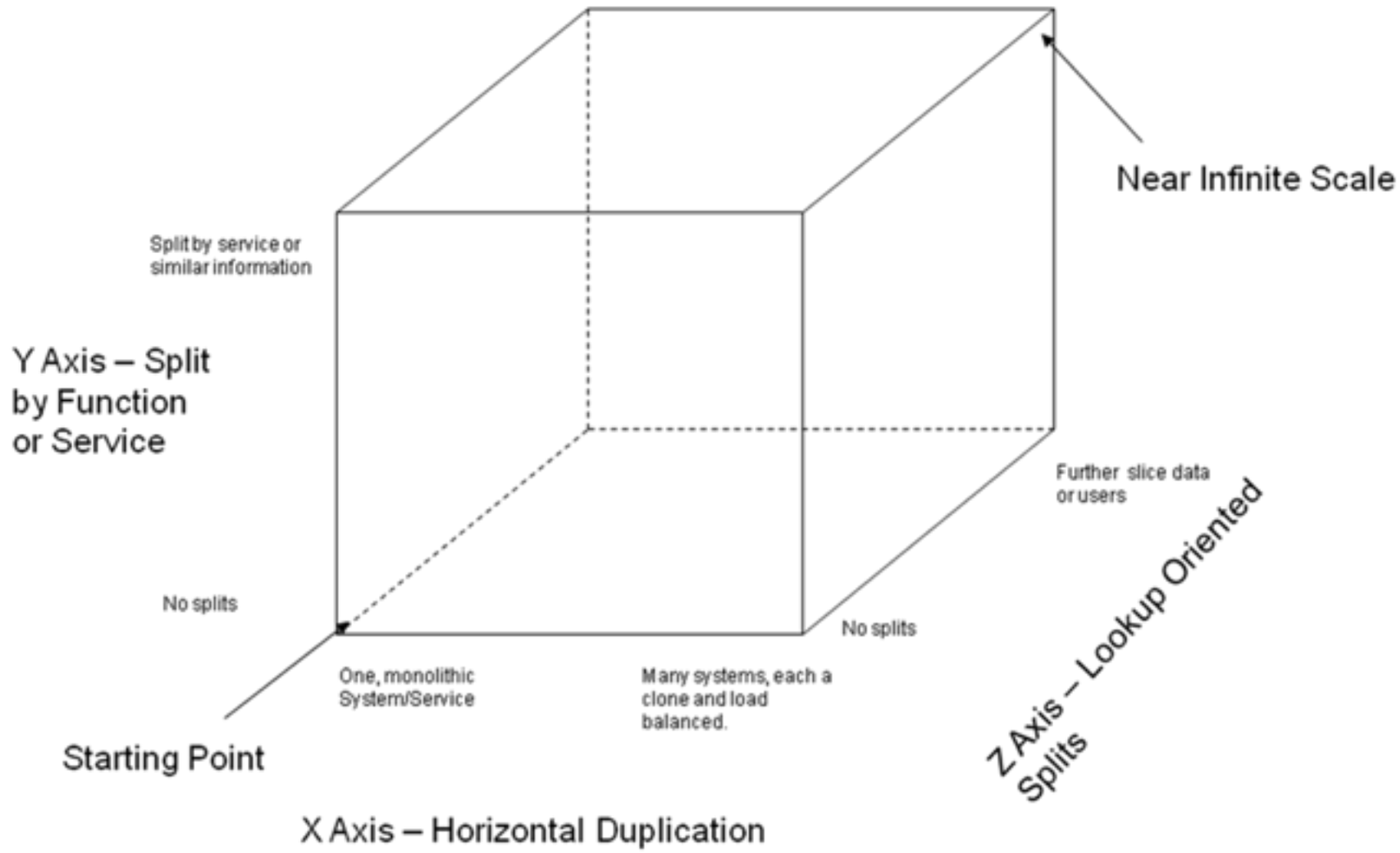
Monolith Application



Disadvantages of Monolith Application

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





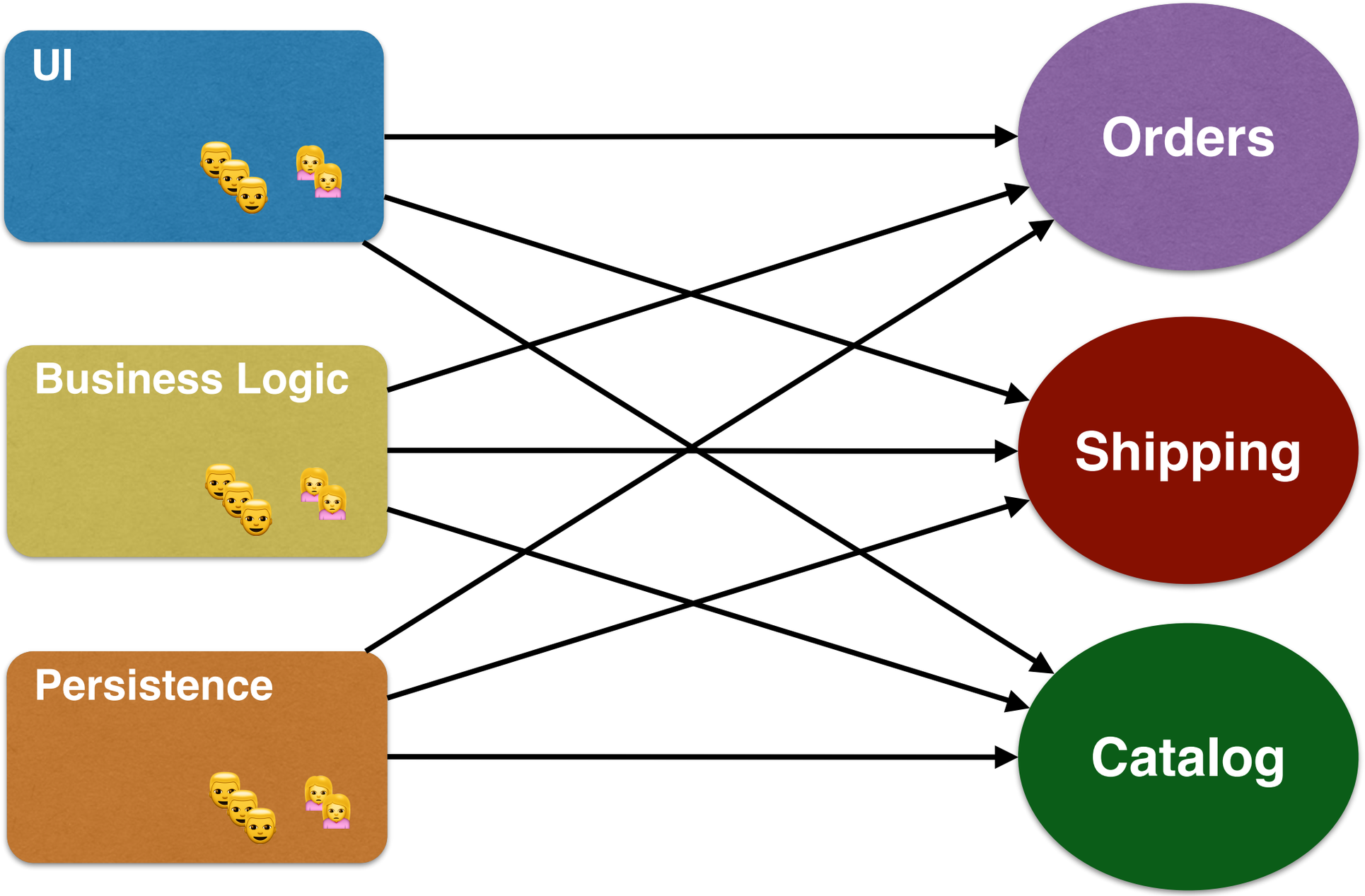
http://akfpartners.com/techblog/2008/05/08/splitting-applications-or-services-for-scale/

redhat

*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**.*

redhat


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

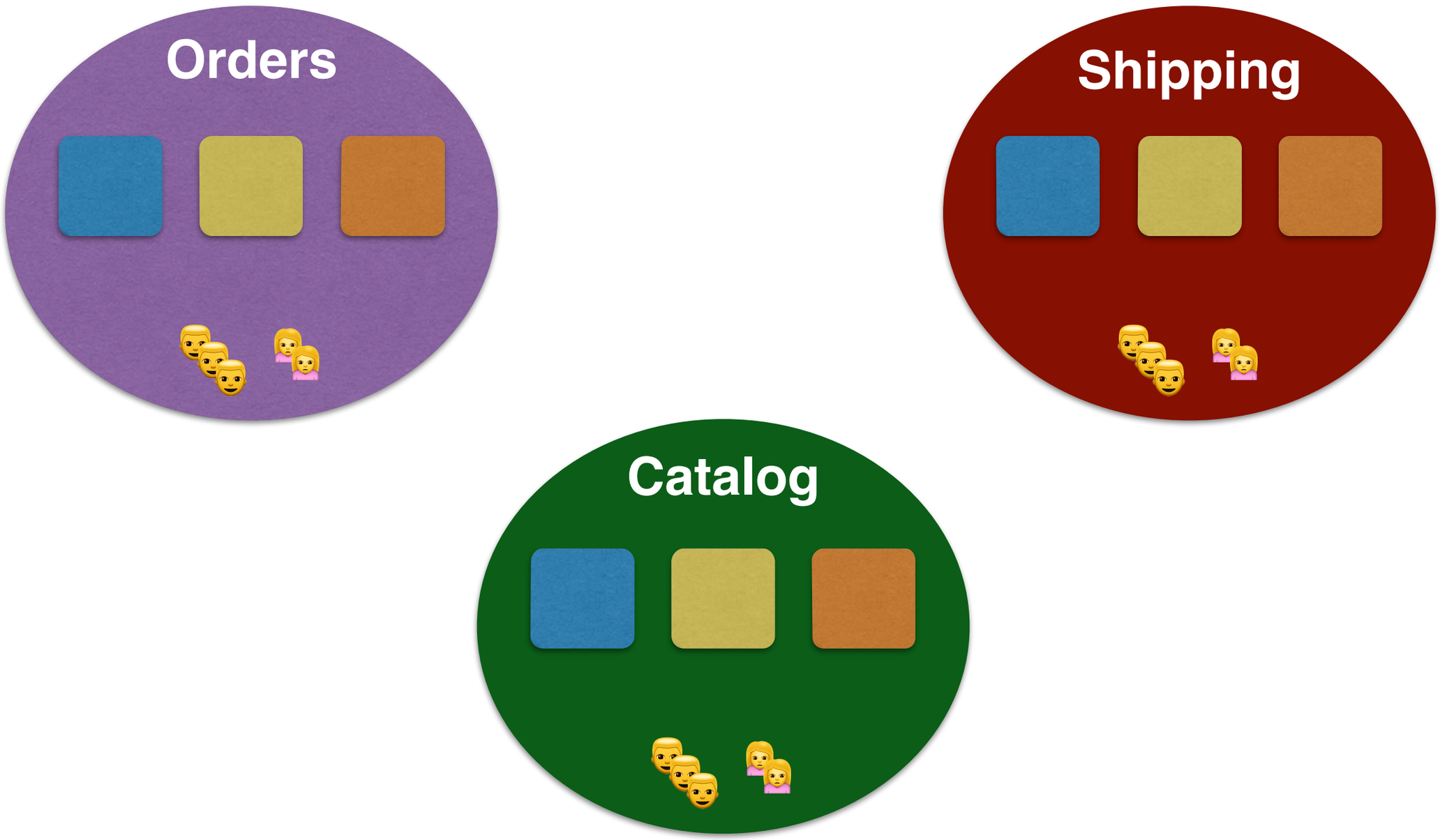


http://www.melconway.com/Home/Committees_Paper.html

redhat

Characteristics

Teams around business capability



Characteristics

Single Responsibility Principle

DO

1

THING

Characteristics

Explicitly Published interface

Characteristics

Independently replaceable and upgradeable

redhat

Characteristics

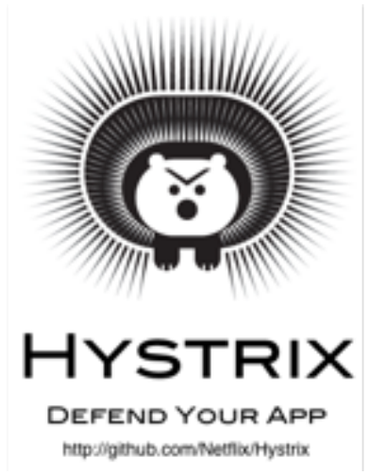
With great power, comes great responsibility

“you build it, you run it!”

redhat

Characteristics

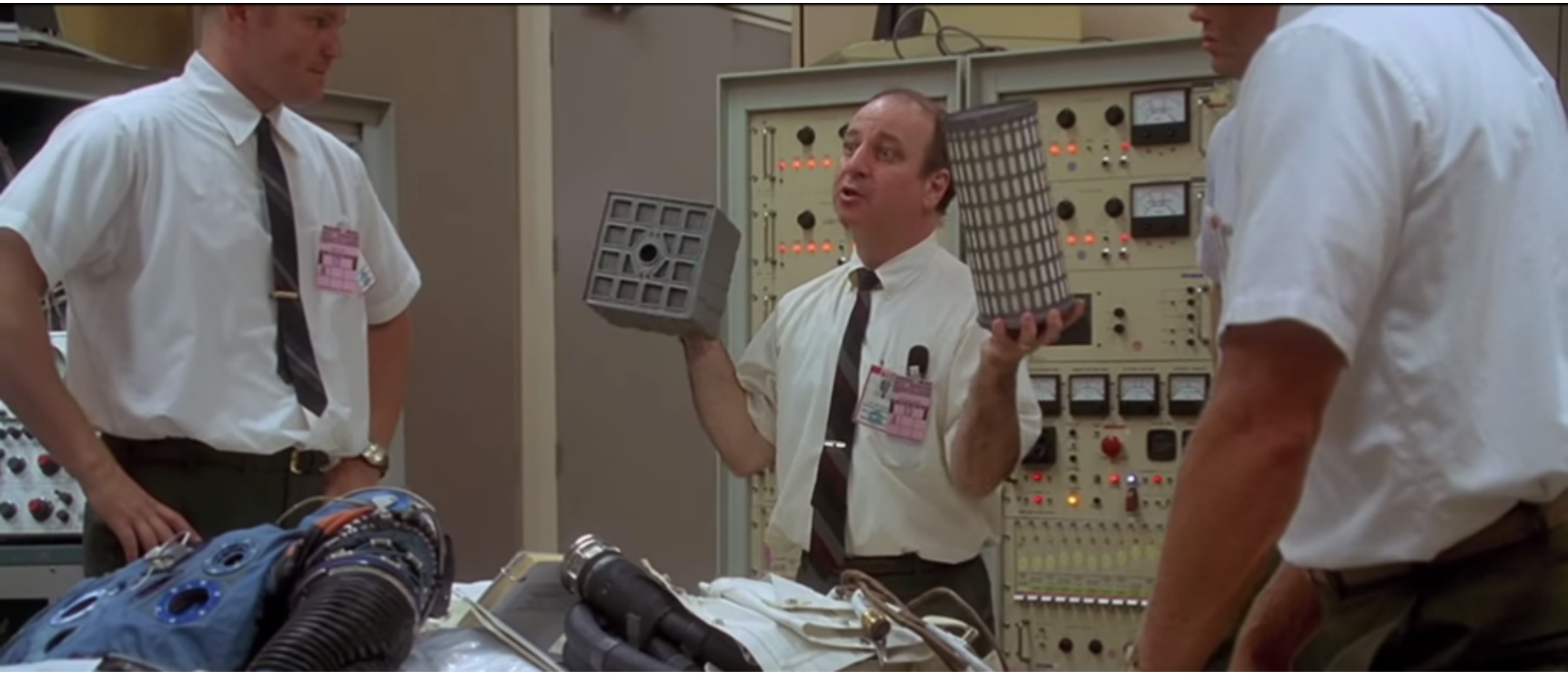
Designed for failure



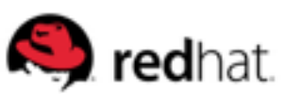
Fault tolerance is a requirement, not a feature

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

redhat

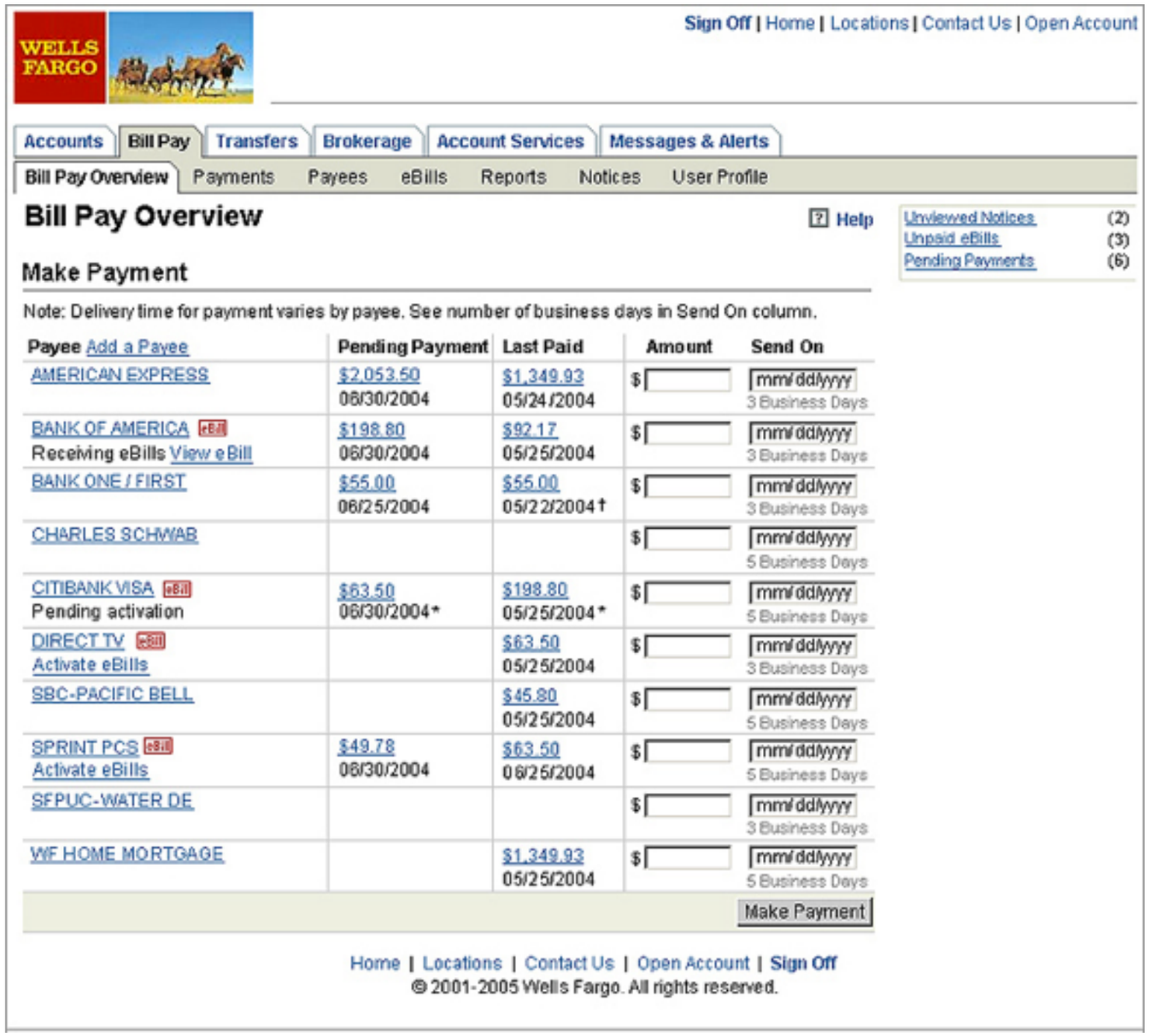


Characteristics



Characteristics

100% automated



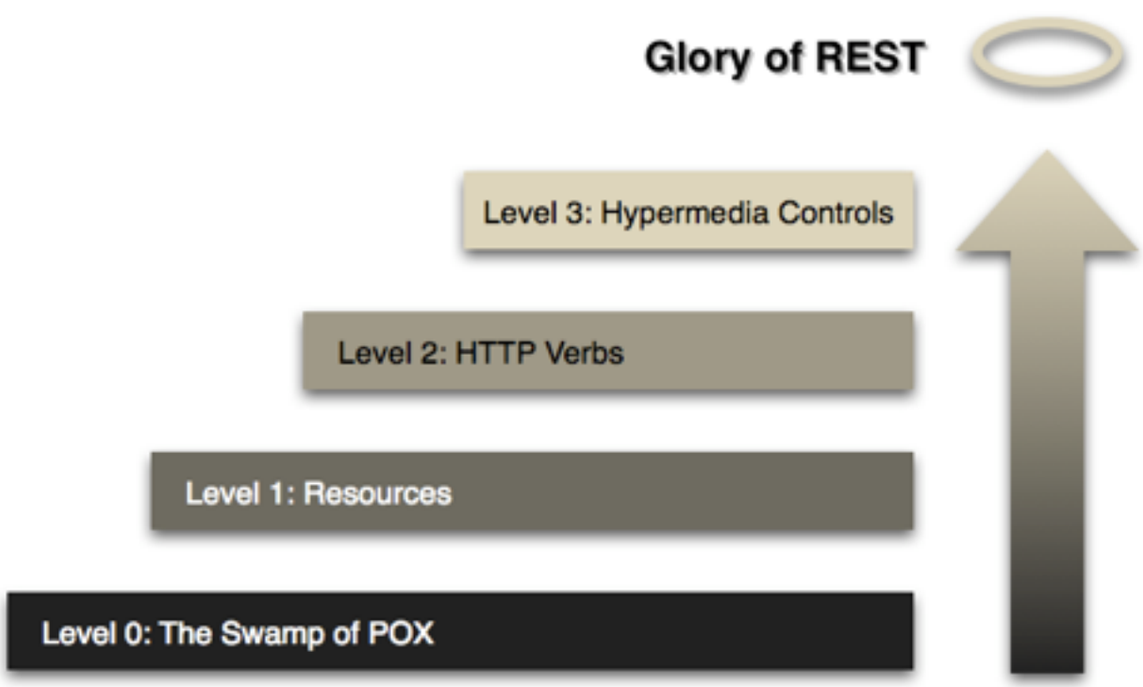
redhat

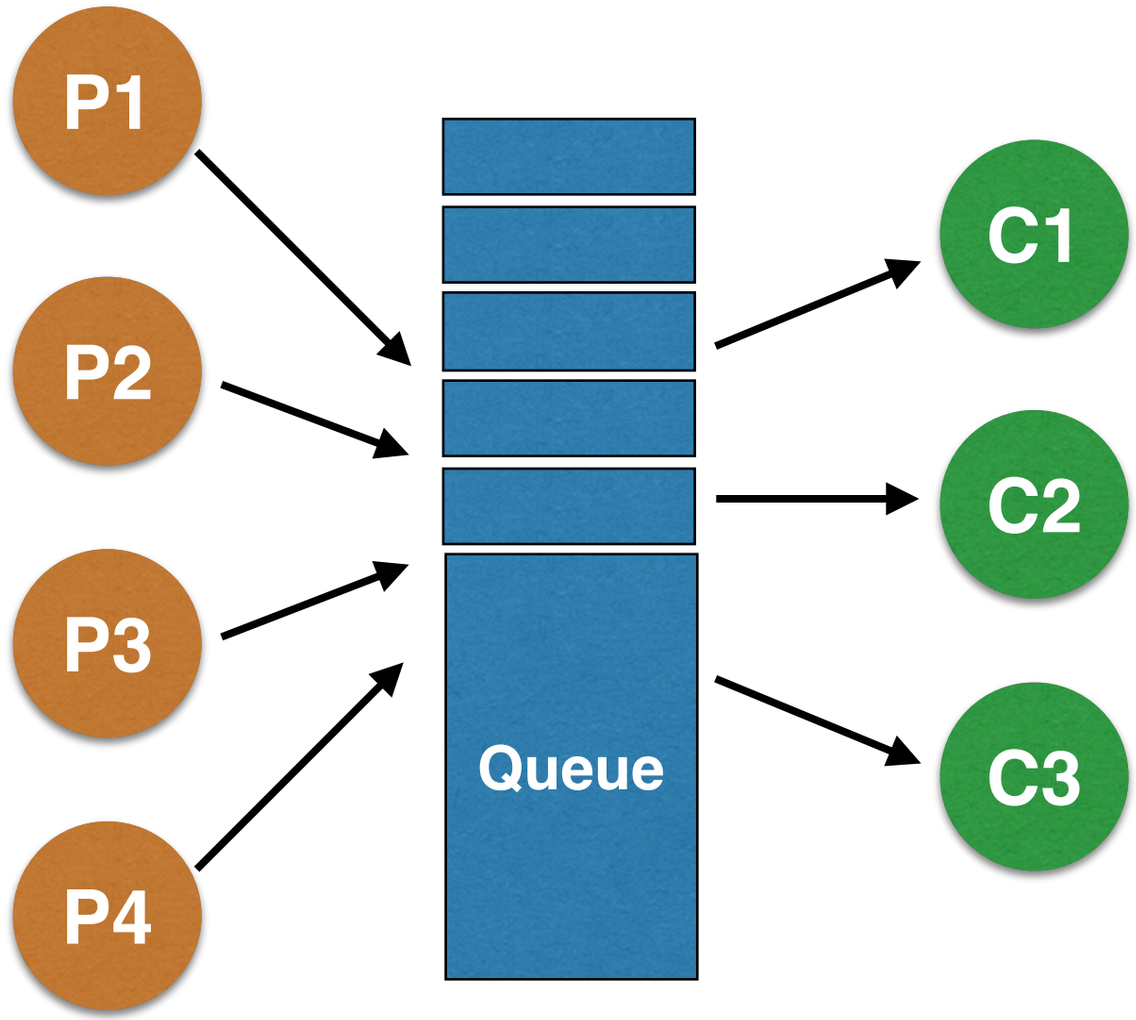
Characteristics

Sync or Async Messaging

Characteristics

REST vs Pub/Sub





redhat

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

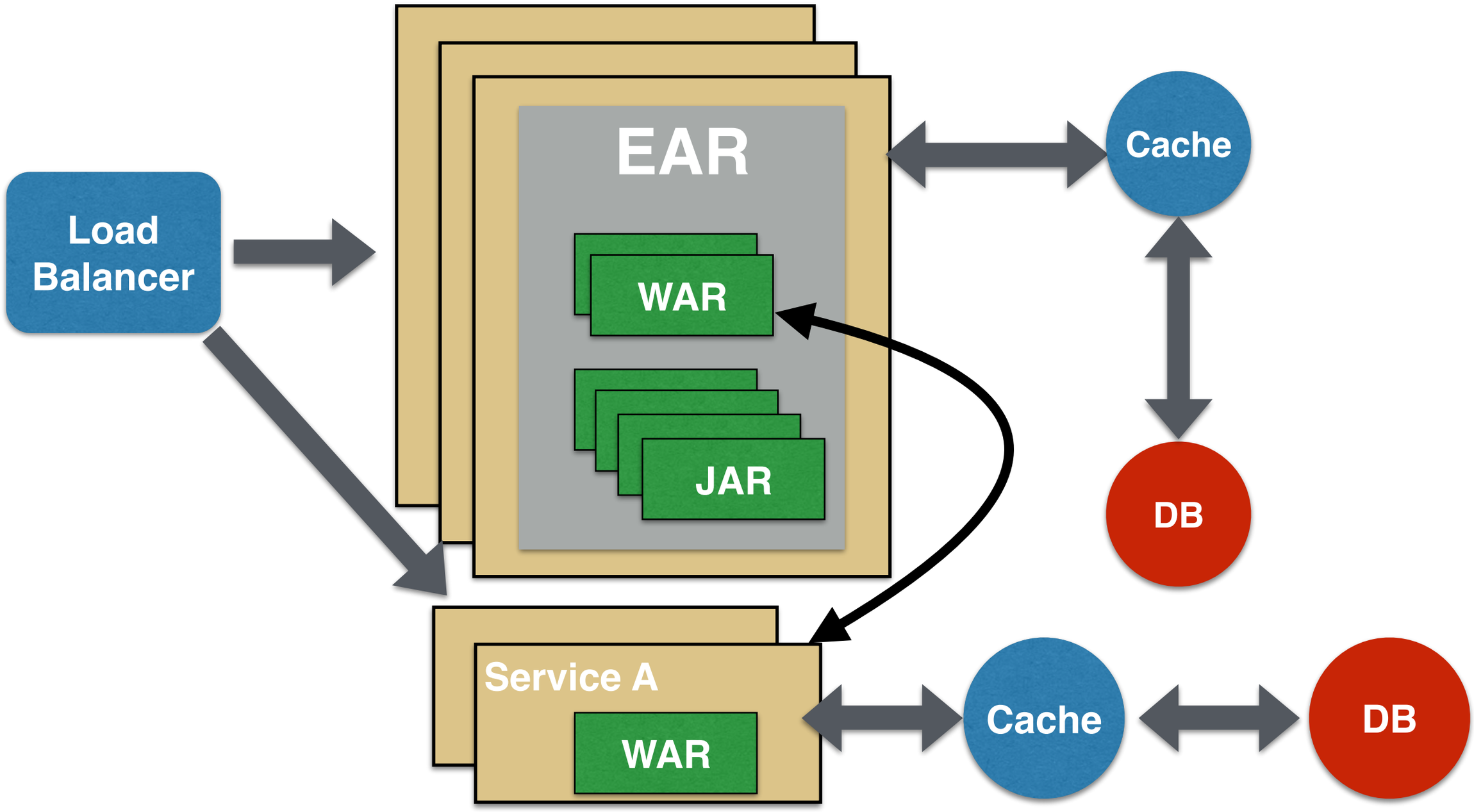
5:07 PM - 27 May 2015

Strategies for decomposing

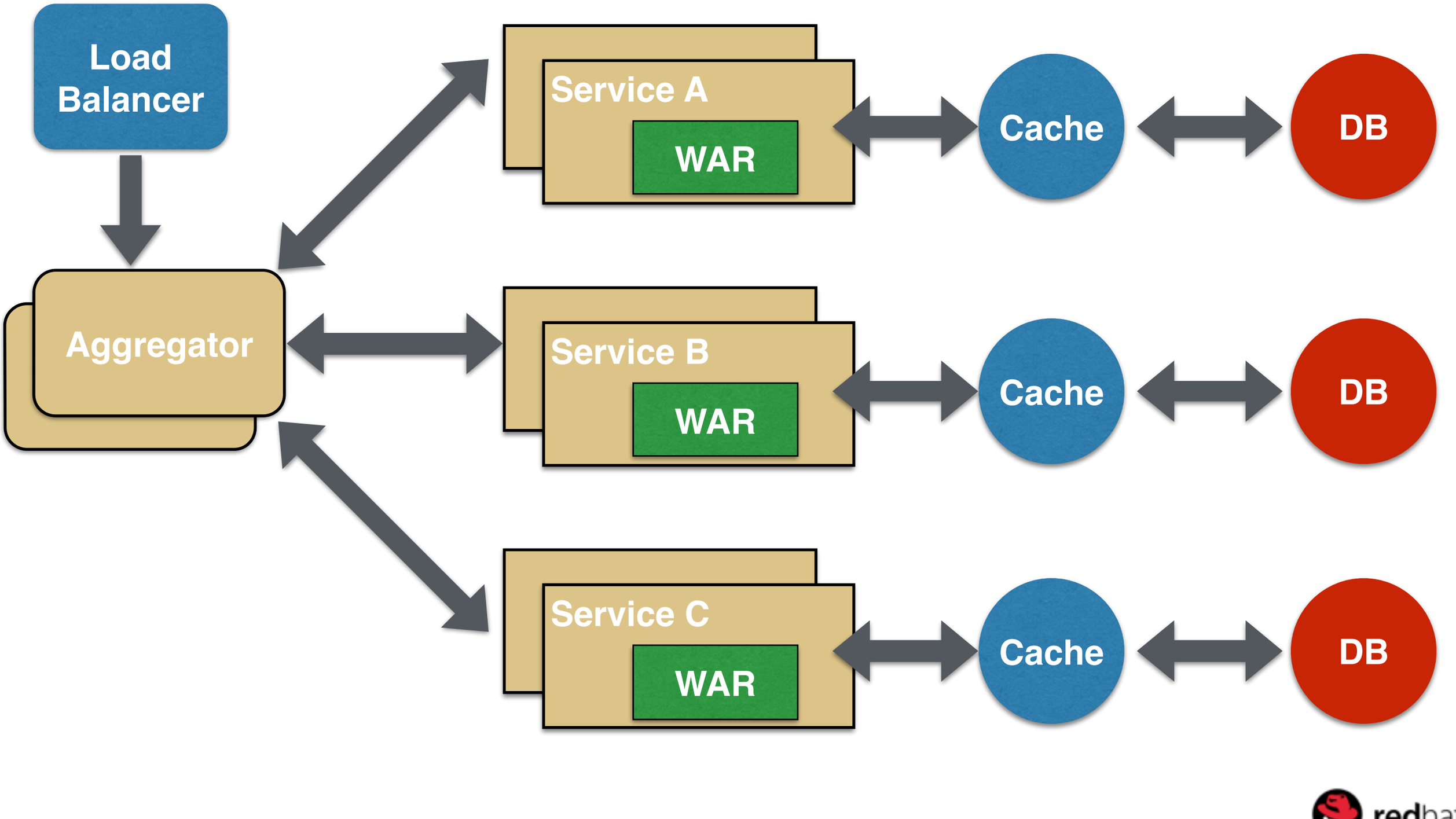
Strategies for decomposing

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

Towards microservices

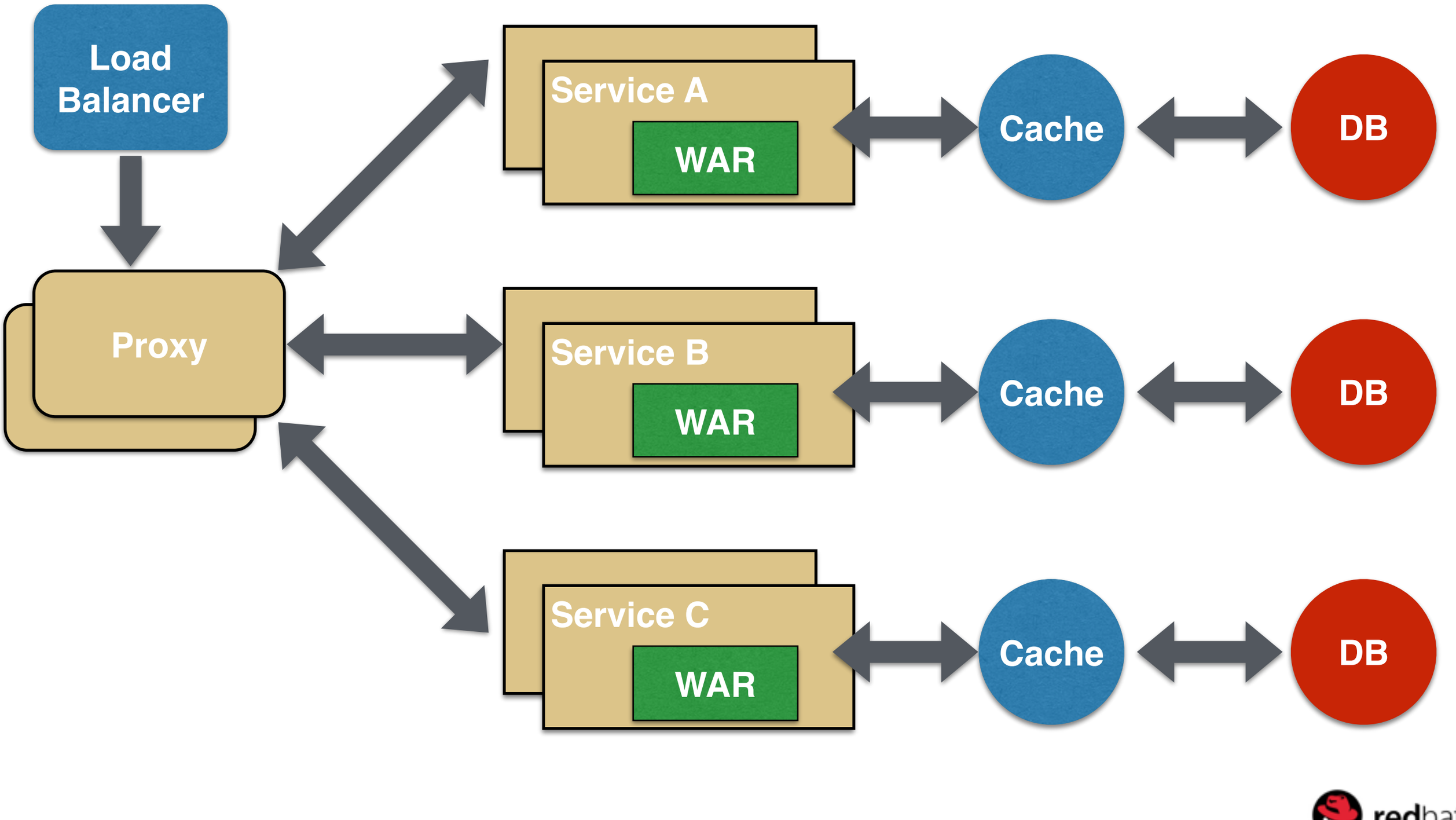


Aggregator Pattern #1



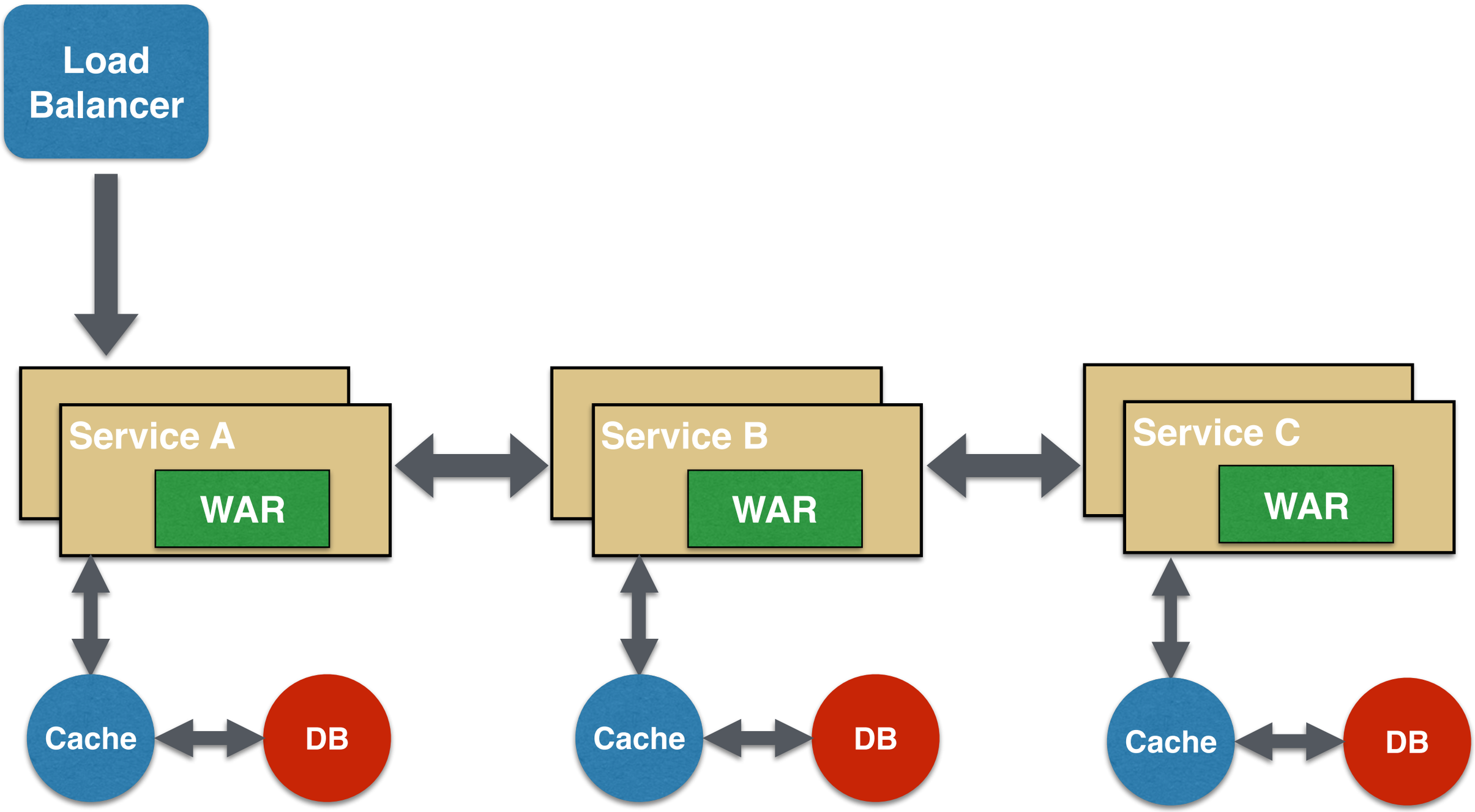
redhat

Proxy Pattern #2

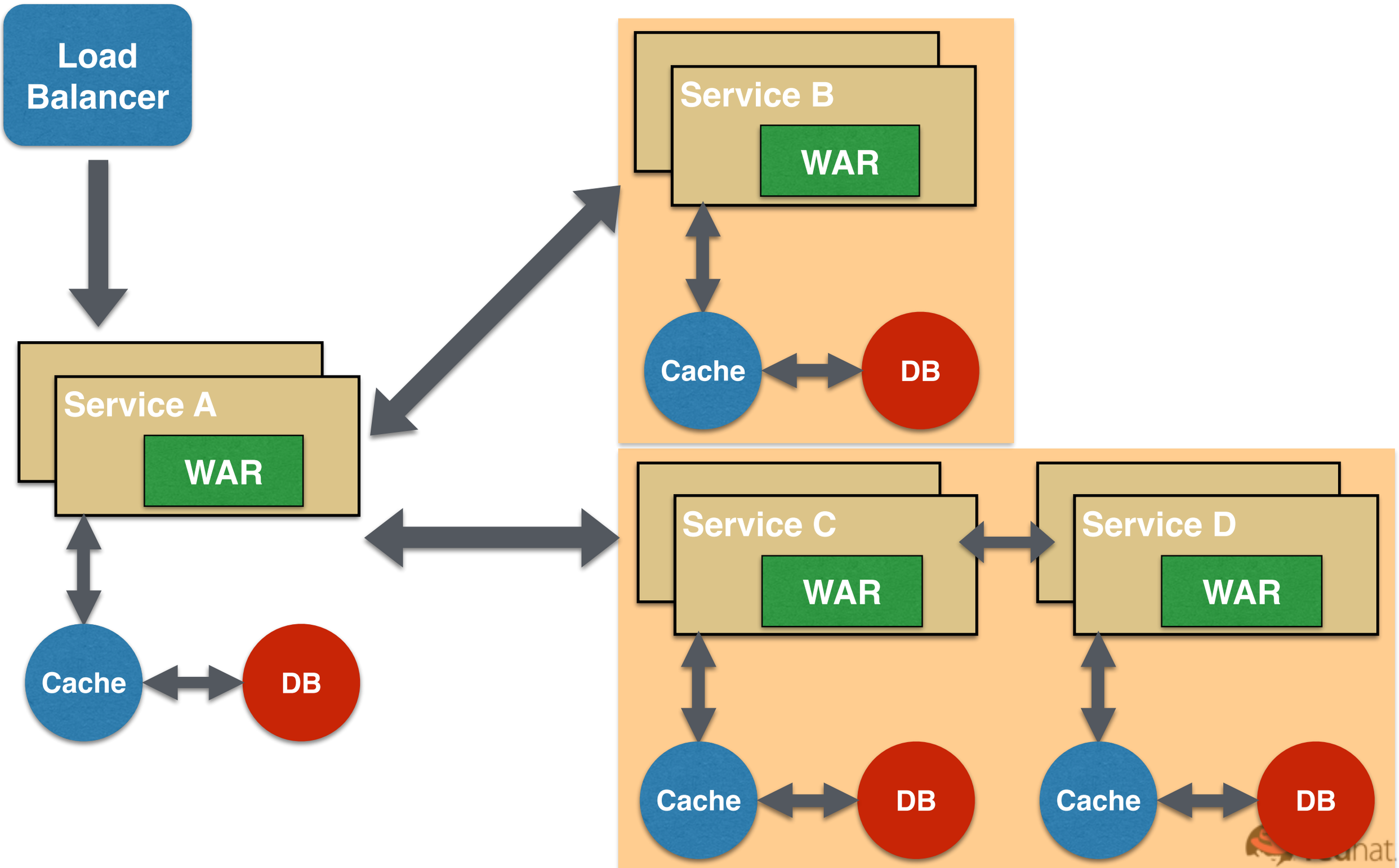


redhat

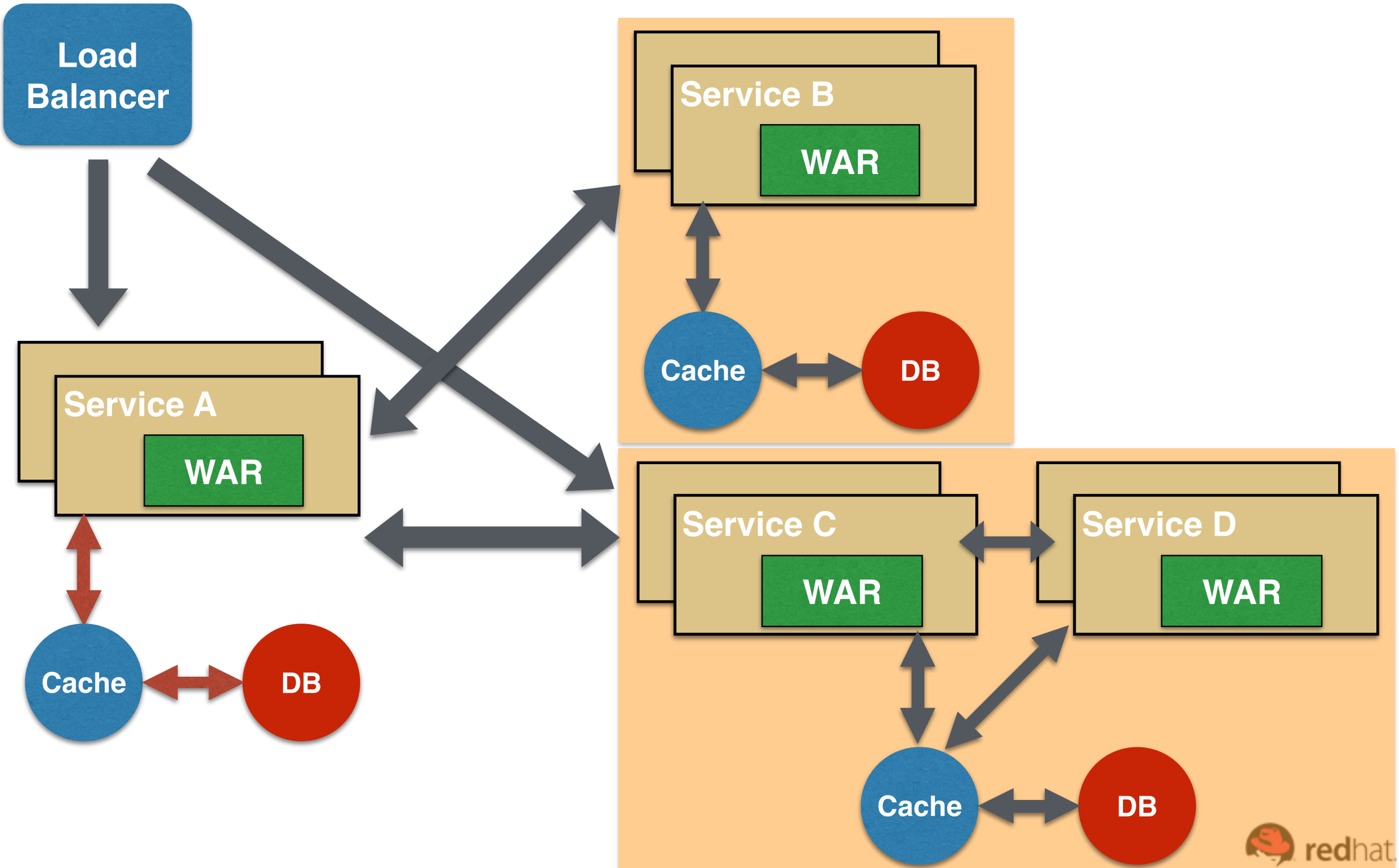
Chained Pattern #3



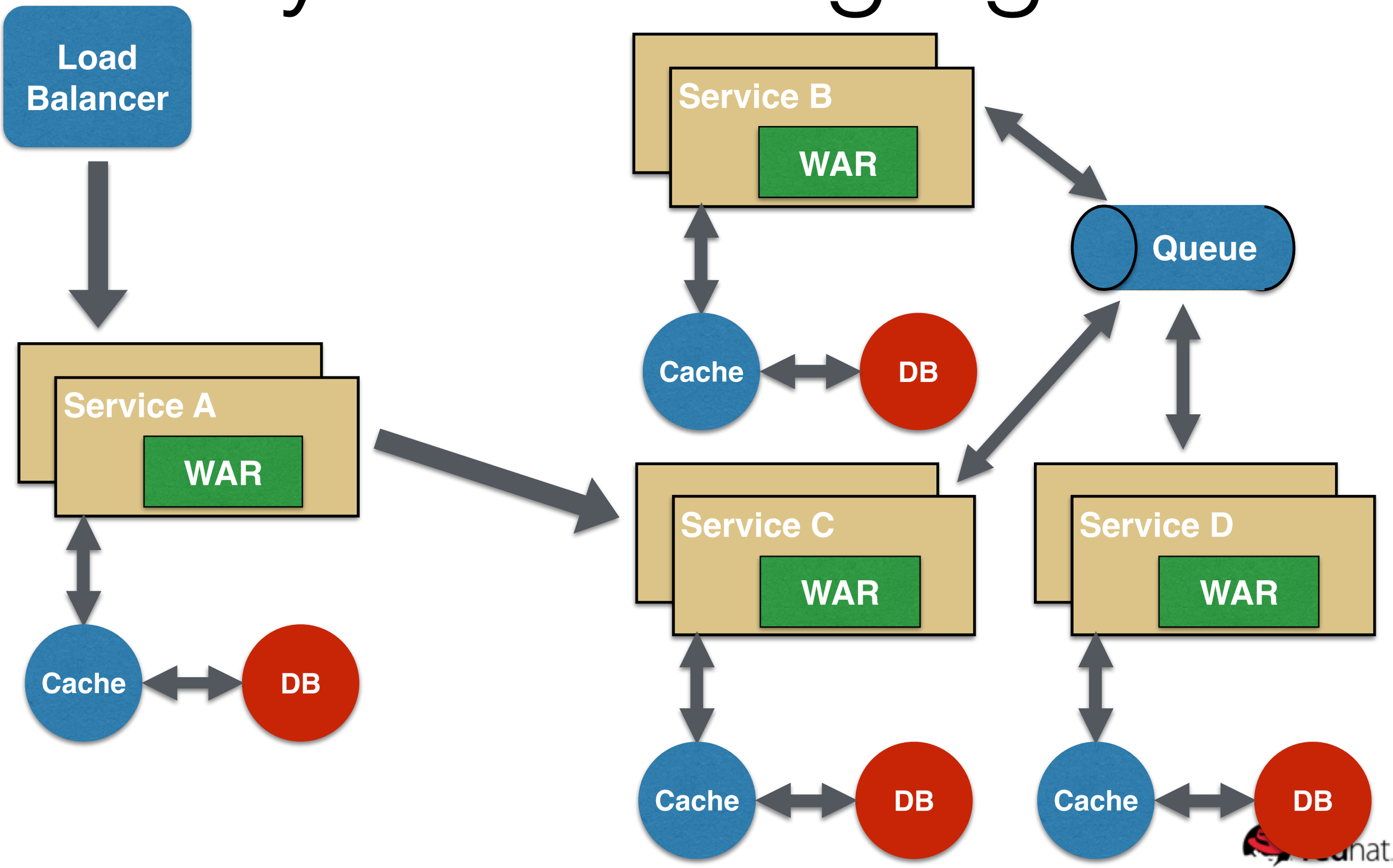
Branch Pattern #4



Shared Resources #5



Async Messaging #5




SAY MICROSERVICE
ONE MORE TIME
memegenerator.net

Advantages of microservices

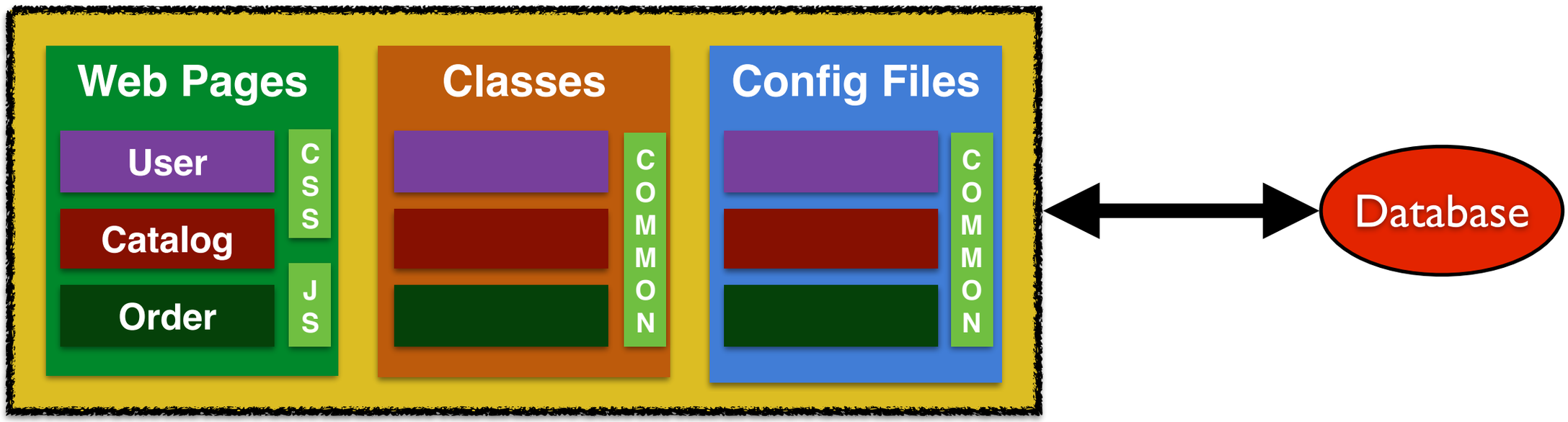
- Easier to develop, understand, maintain
- Starts faster than a monolith, speeds up deployments
- Local change can be easily deployed, great enabler of CD
- Each service can scale on X- and Z-axis
- Improves fault isolation
- Eliminates any long-term commitment to a technology stack
- Freedom of choice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

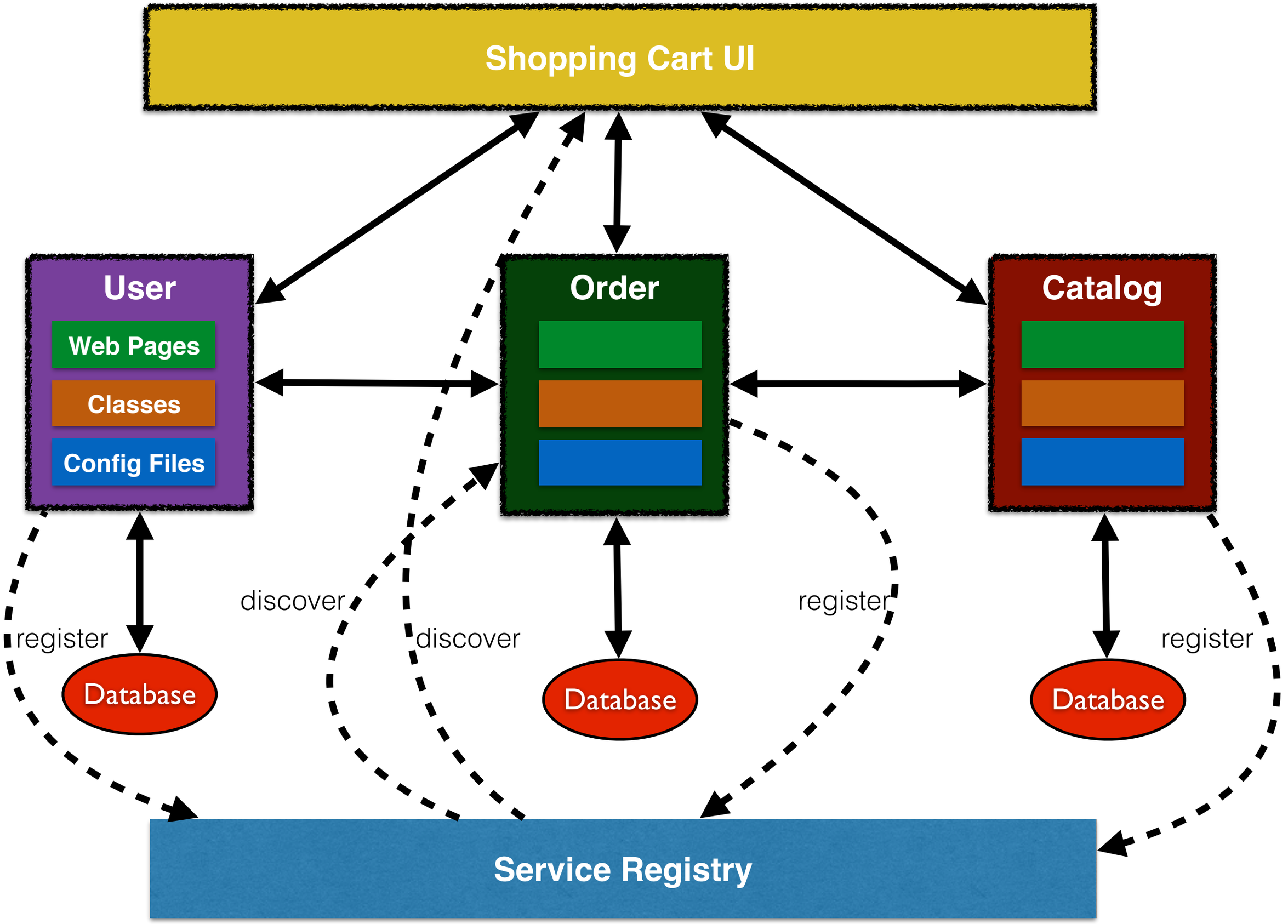
“If you can't build a [well-structured] monolith, what makes you think microservices are the answer?”

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

Arun Gupta

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


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

- 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, …

Design Time

App Developers

configure

Service Providers

configure

API Manager

Runtime

App Users

apply

Service Users

apply

API Gateway

Services

Services

Services

Services

redhat

API Manager

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

API Gateway

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

Design Time

App Developers

configure

Service Providers

configure

API Manager

User Interface

REST API

LDAP

Services

Services

Services

Services

Runtime

App Users

apply

Service Users

apply

API Gateway

REST API

Policy Engine

LDAP

JDBC

REST

Design Time

App Developers

configure

Service Providers

configure

API Manager

User Interface

REST API

LDAP

Runtime

apply

apply

B2B

API Gateway

REST API

Policy Engine

LDAP

JDBC

REST

Java EE

node.js

VERT.X

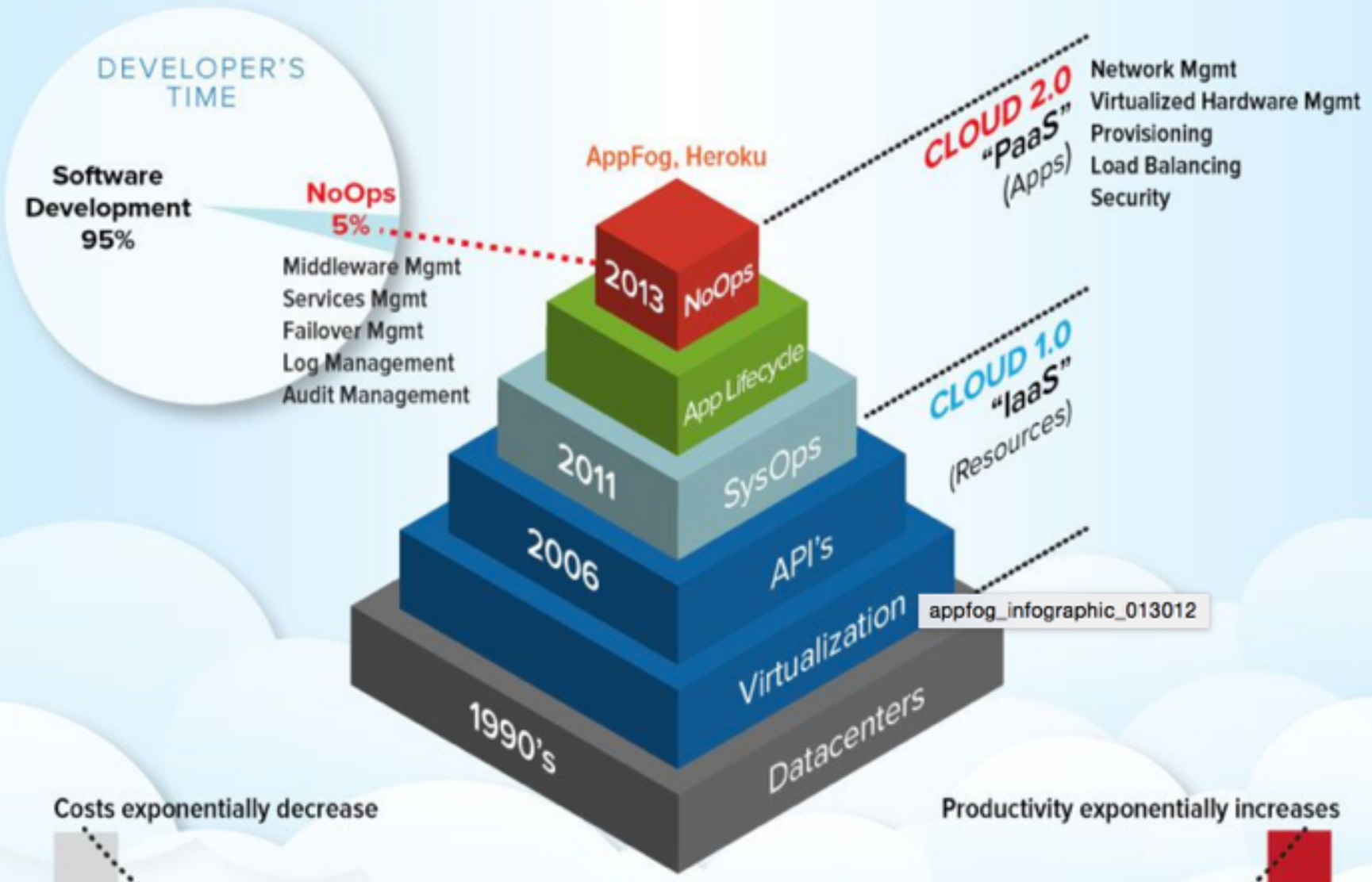
redhat

NoOps

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

2013: A bright NoOps future

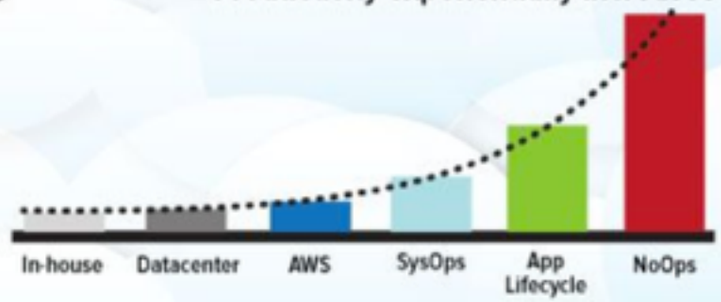
So where does this all lead? The end-game is NoOps. Where building and running an app is purely a developer process — and where developers are not having to spend time doing Ops work.



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.

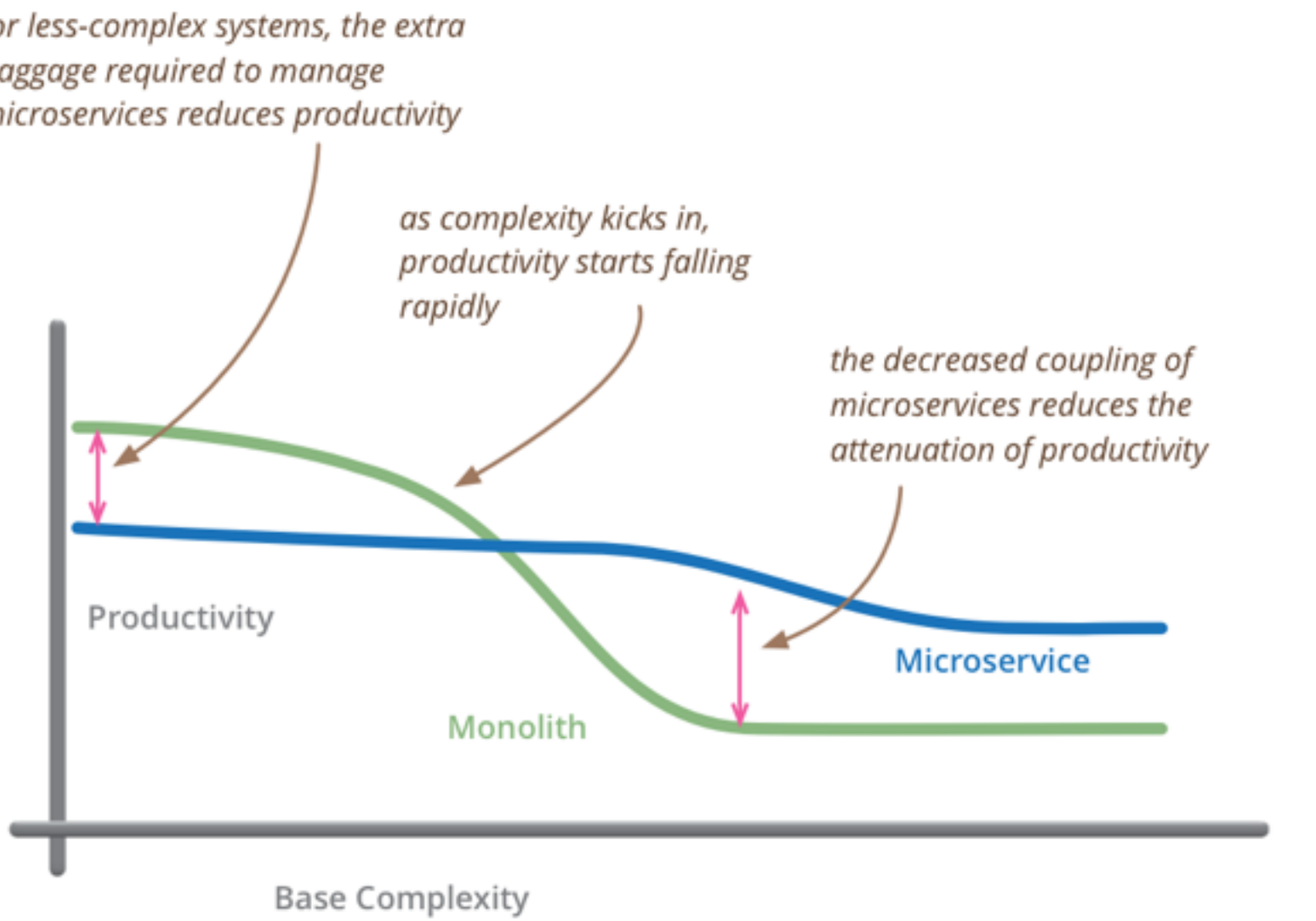
By freeing developers to focus on what they want to do — and on what they do best — we are enabling them to even more easily start companies and build products. And that is why AppFog is a *Start-up Enabler.*

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

Drawbacks of microservices

- 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



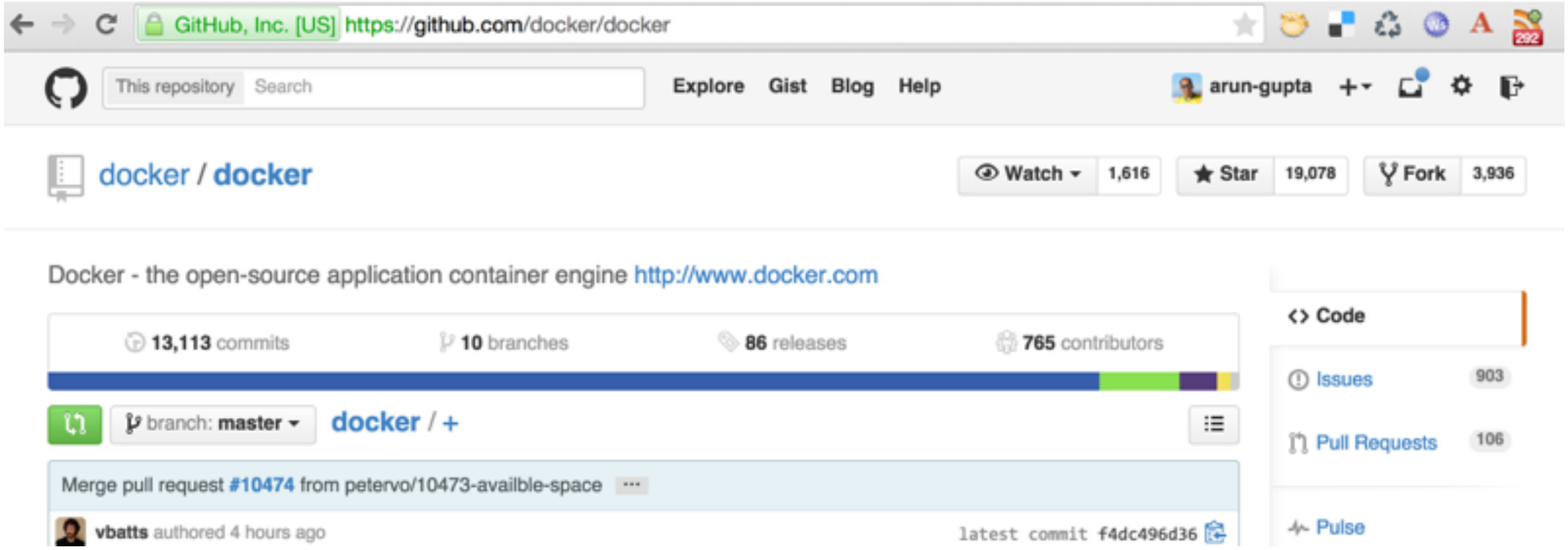
"don't even consider microservices unless you have a system that's too complex to manage as a monolith"

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

redhat

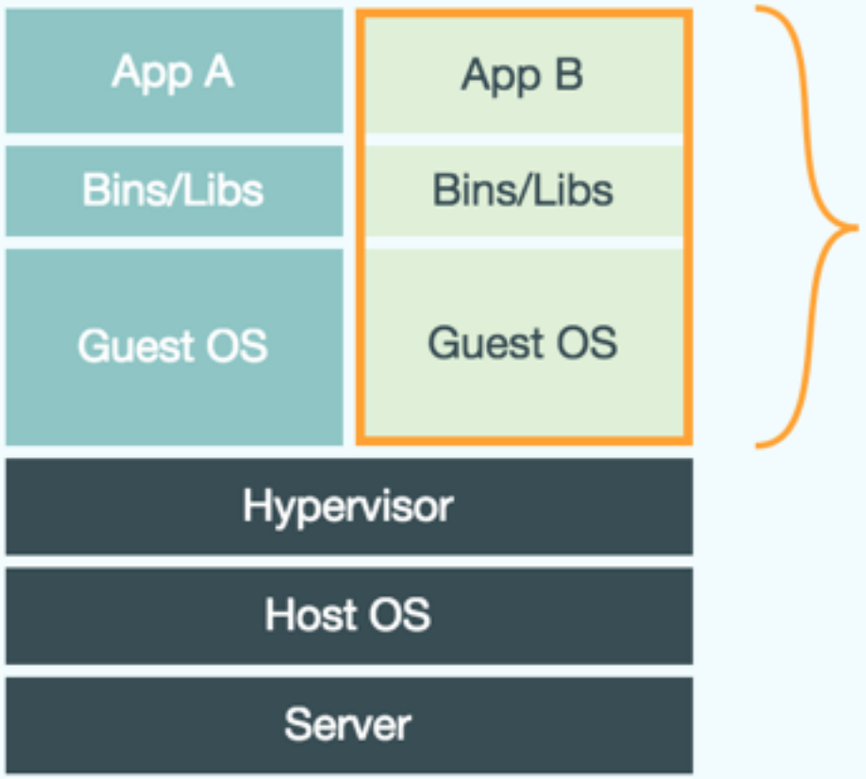
What is Docker?

- Open source project and company



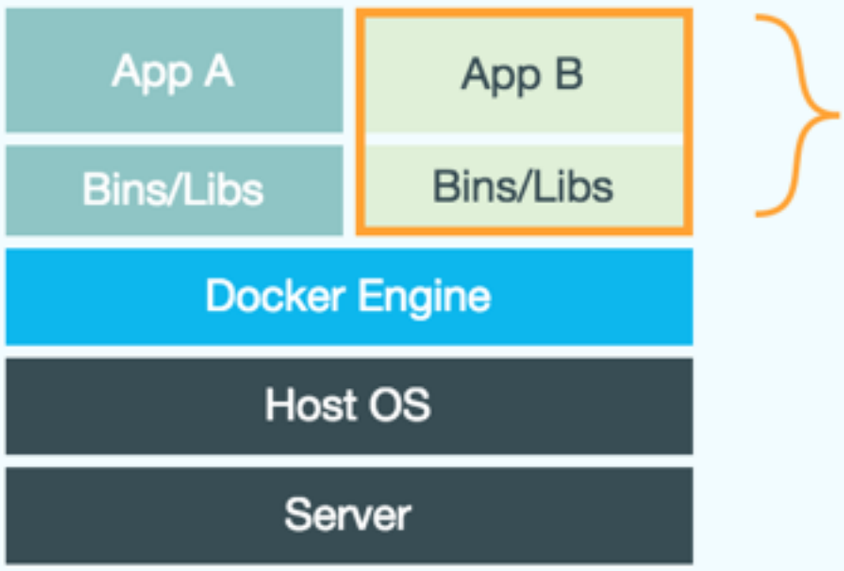
- Used to create containers for software applications

- Package Once Deploy Anywhere (PODA)



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

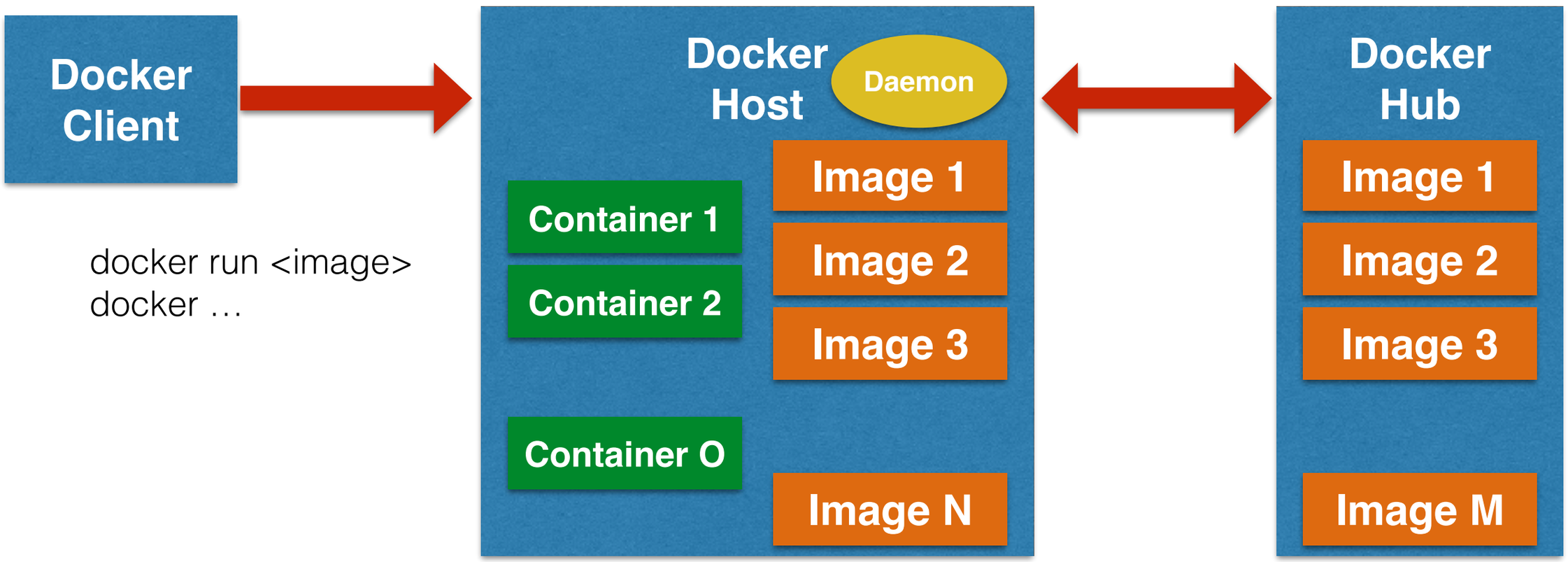
The Docker Engine container comprises just the application and its dependencies. It runs as an isolated process in userspace on the host operating system, sharing the kernel with other containers. Thus, it enjoys the resource isolation and allocation benefits of VMs but is much more portable and efficient.

Underlying Technology

- 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 run <image>
docker …

- 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
```

Advantages of Containers

- Faster deployments
- Isolation
- Portability - “it works on my machine”
- Snapshotting
- Security sandbox
- Limit resource usage
- Simplified dependency
- Sharing

Docker: Pros and Cons

- PROS
  - Extreme application portability
  - Very easy to create and work with derivative
  - Fast boot on containers
- CONS
  - Machine, Swarm, Compose not ready for production
  - No usage tracking/reporting

Kubernetes

- Open source orchestration system for Docker containers

- Provide declarative primitives for the “desired state”

- Self-healing

- Auto-restarting

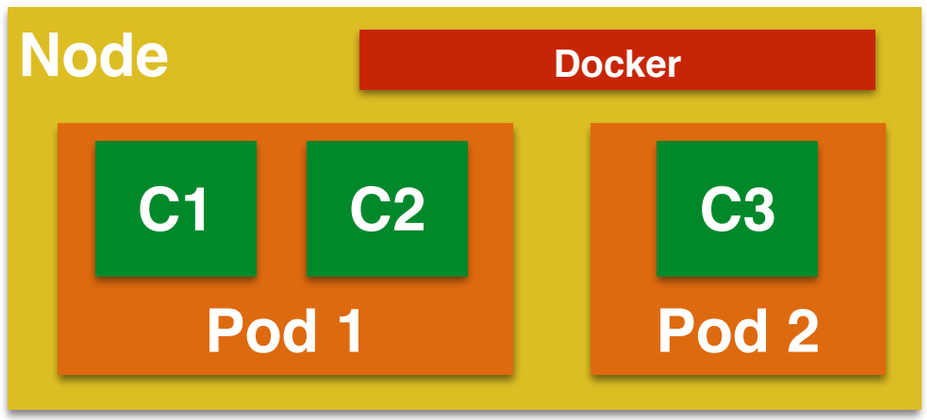
- Schedule across hosts

- Replicating

redhat

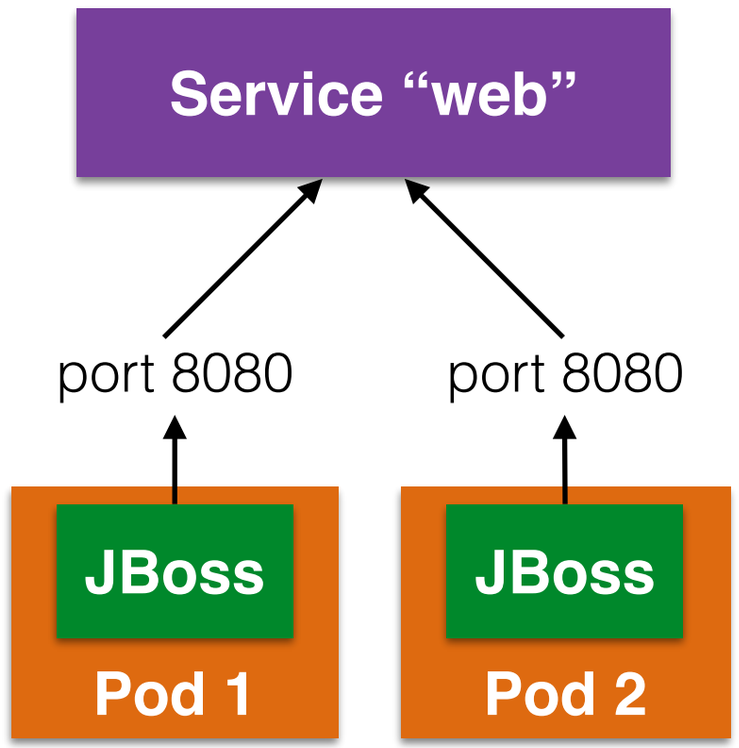
Concepts

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

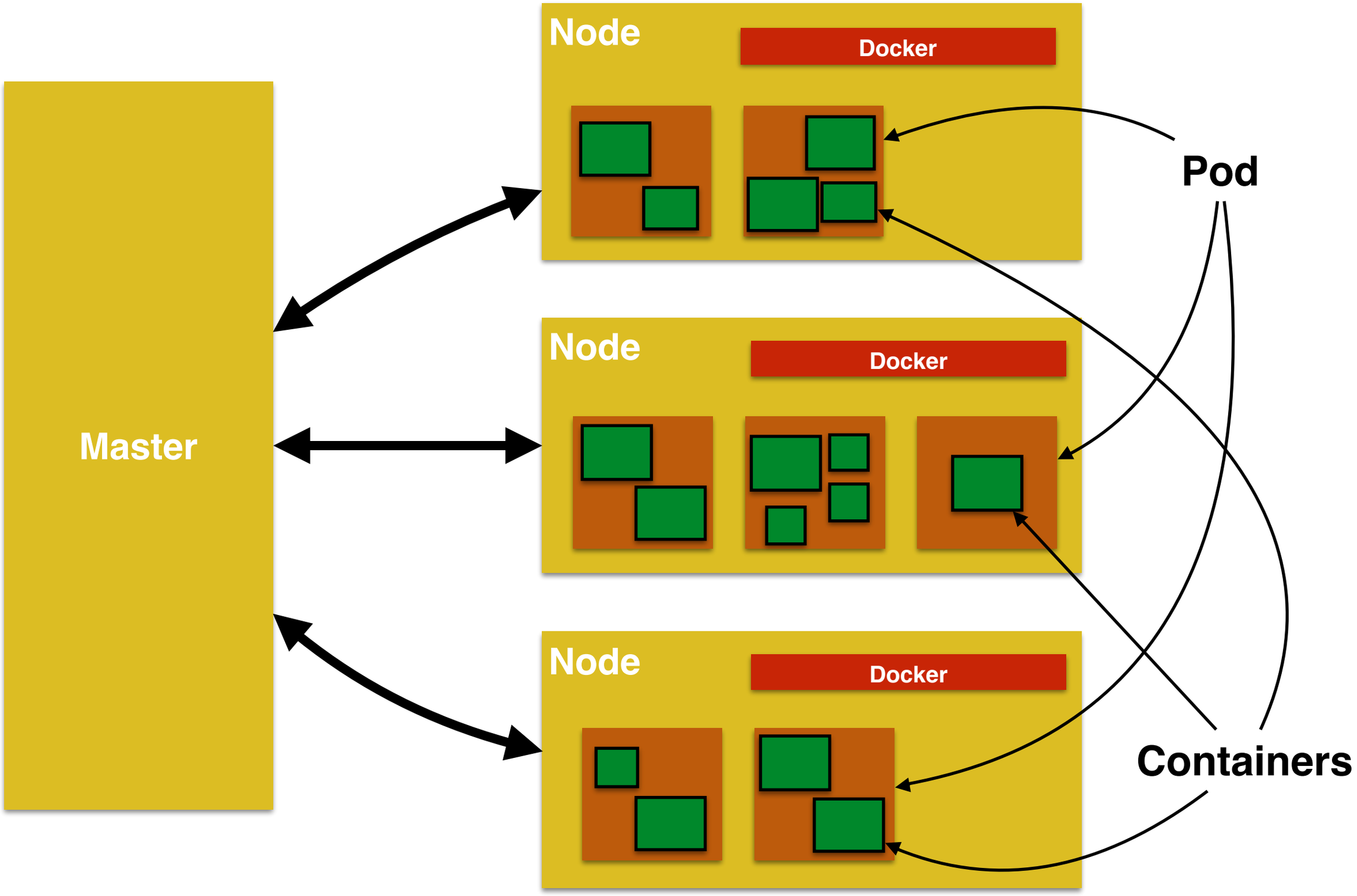


- **Service**: Single, stable name for a set of pods, also acts as LB

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



Kubernetes Architecture



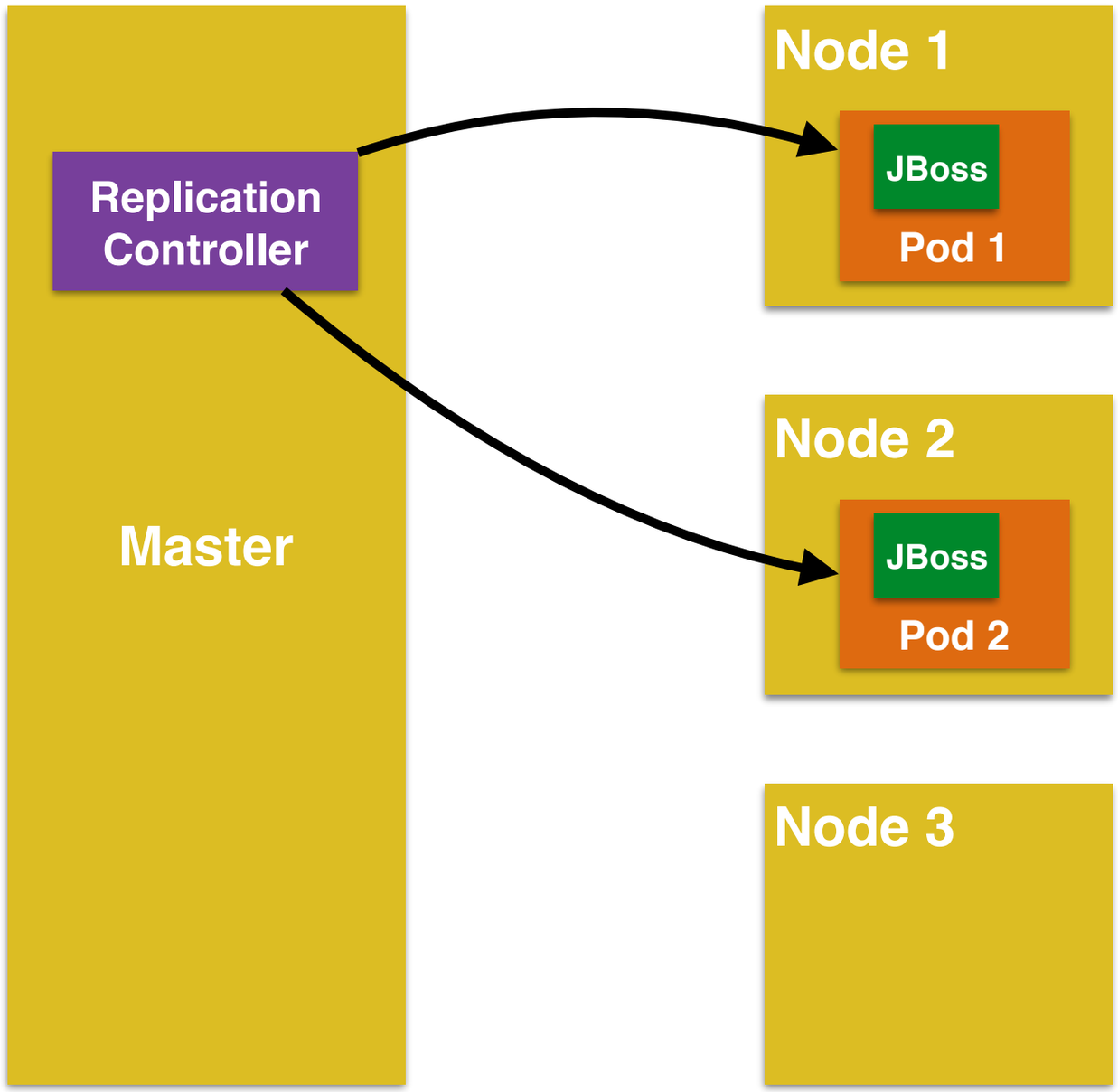
Replication Controller

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

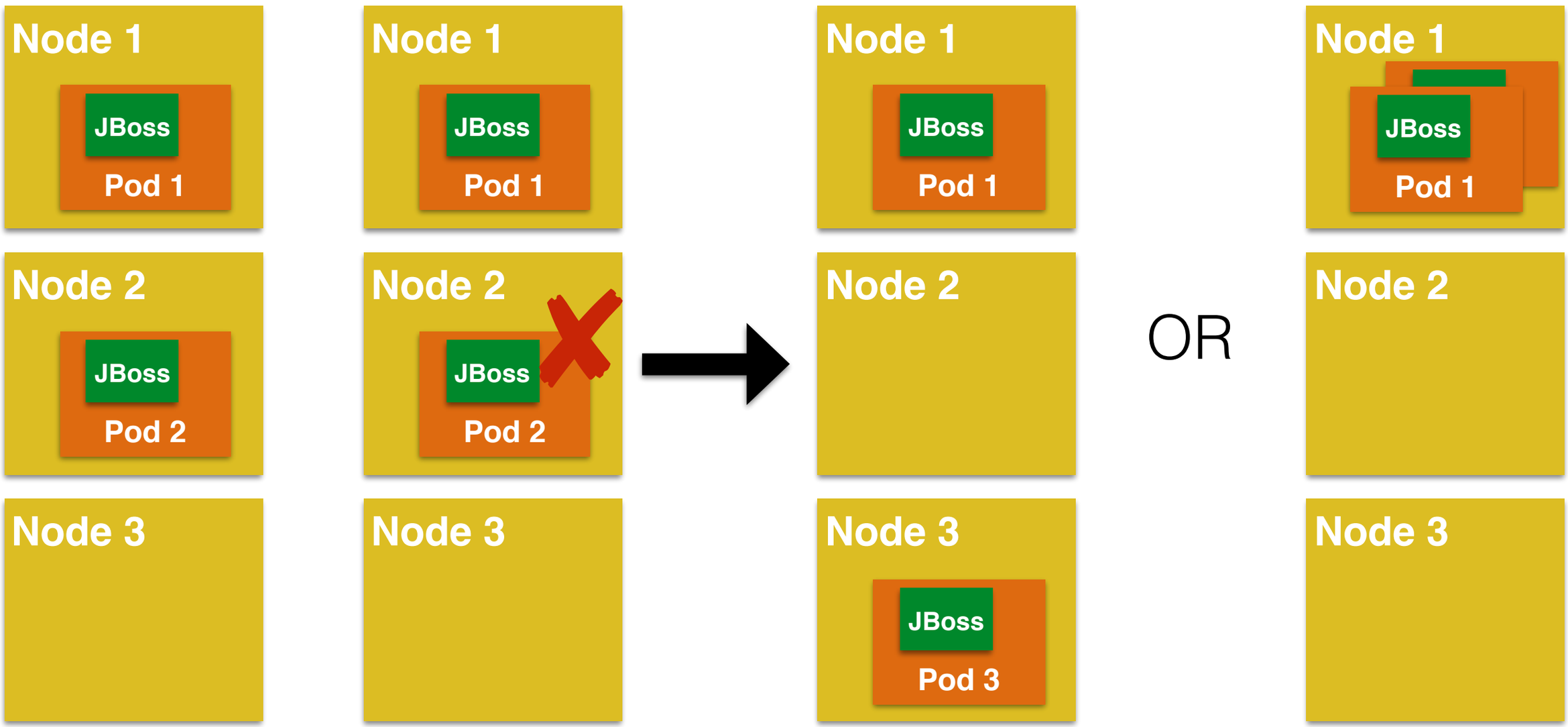
- Recommended to wrap a Pod in a RC

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

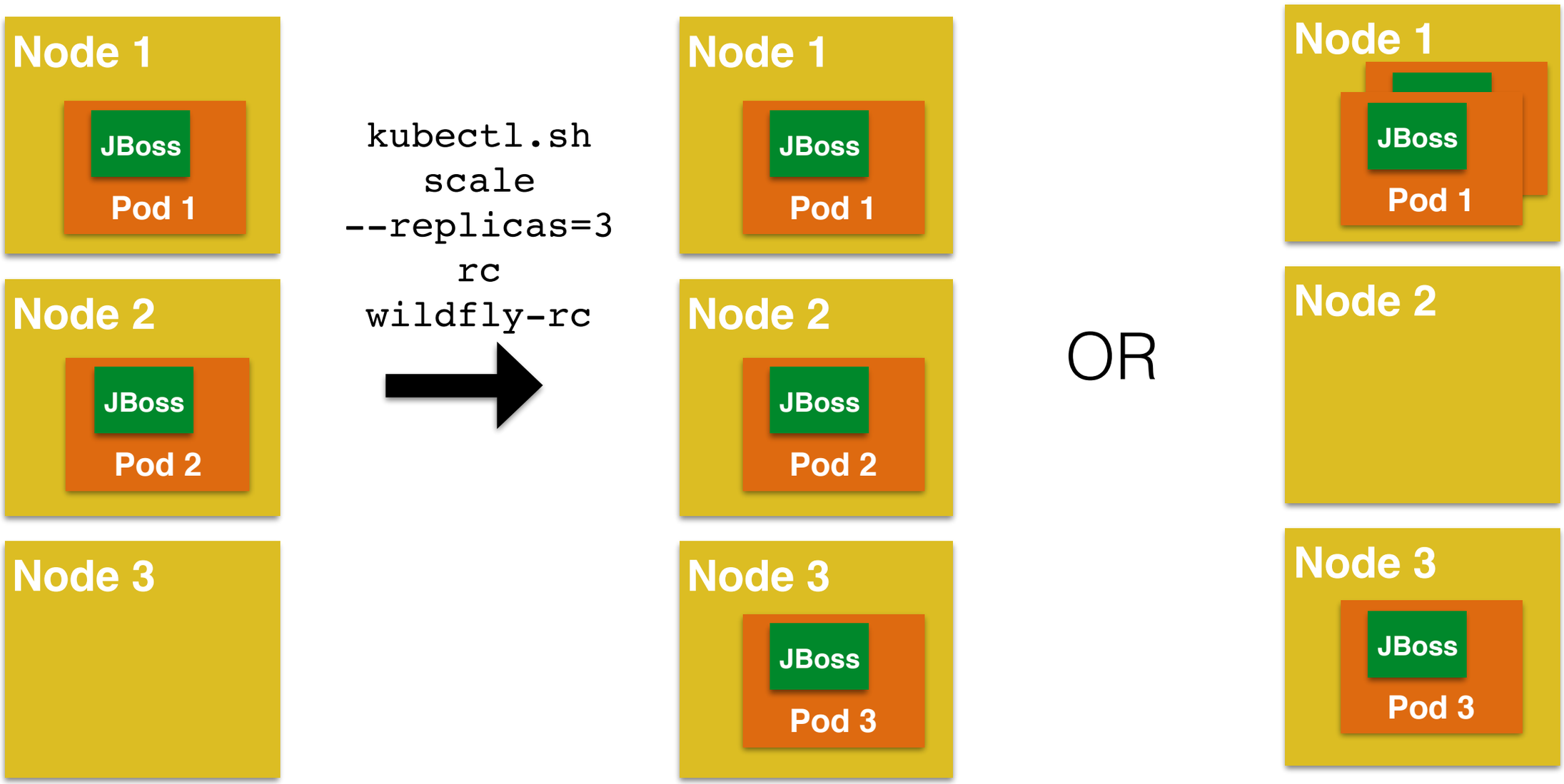
Replication Controller



Replication Controller: Automatic Rescheduling



Replication Controller: 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
```

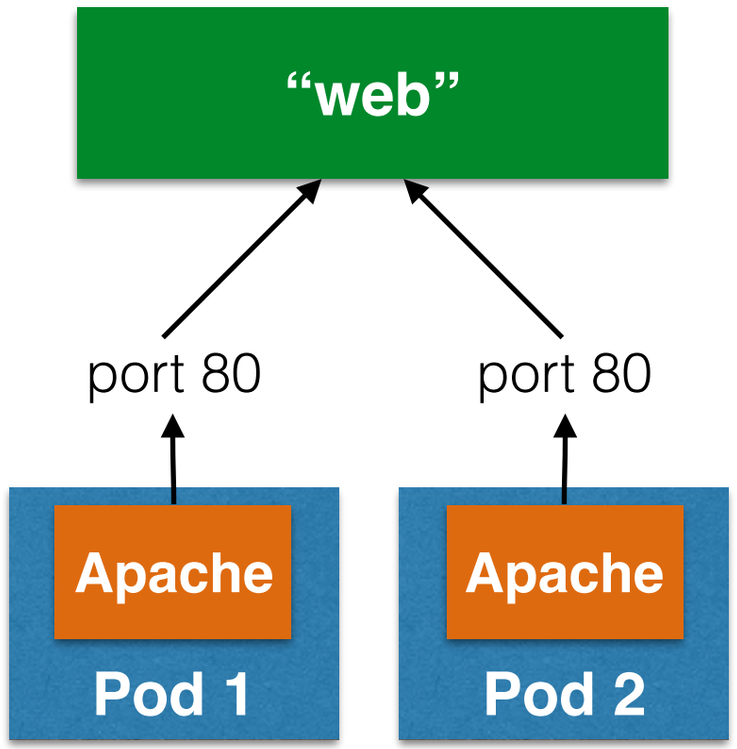
Kubernetes: Pros and Cons

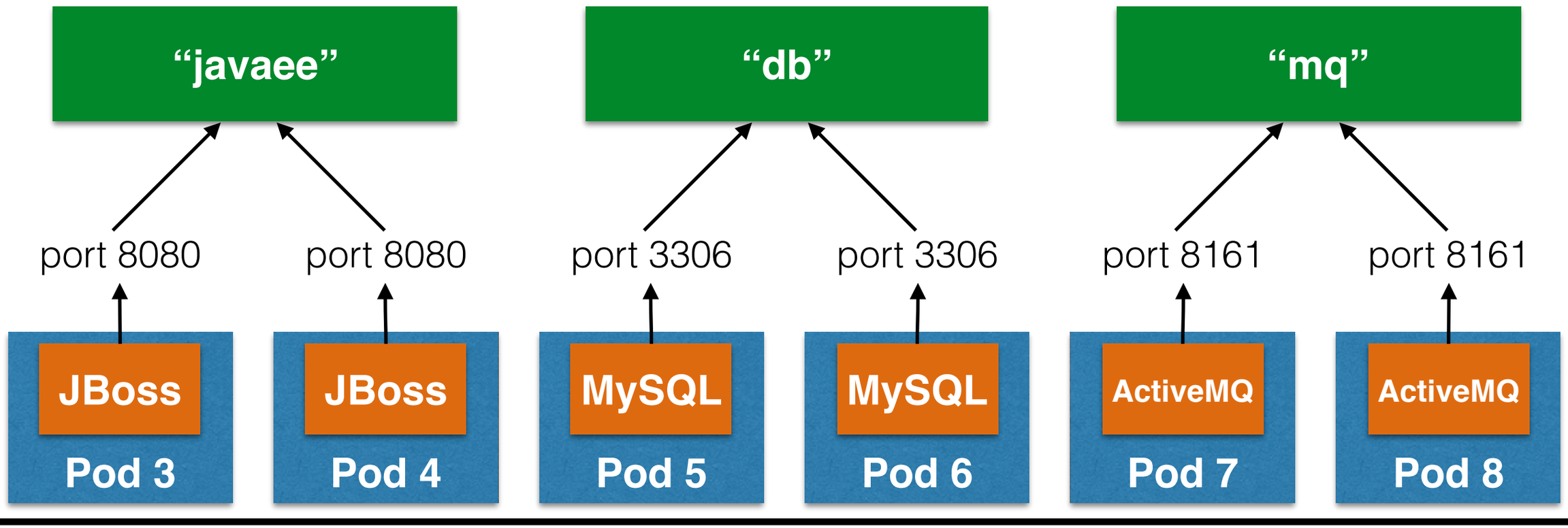
- PROS
  - Manage related Docker containers as a unit
  - Container communication across hosts
  - Availability and scalability through automated deployment and monitoring of pods and their replicas, across hosts

Kubernetes: Pros and Cons

- CONS
  - Lifecycle of applications - build, deploy, manage, promote
  - Port existing source code to run in Kubernetes
  - DevOps: Dev -> Test -> Production
  - No multi-tenancy
  - On-premise (available on GCE)
    - Assumes inter-pod networking as part of infrastructure
    - Requires explicit load balancer

OpenShift Application





redhat

redhat.

Applications

PaaS

Containers & Orchestration

Container Host

IaaS

xPaaS

RED HAT JBOSS MIDDLEWARE

VERT.X

node

OPENSHIFT

kubernetes

docker

RED HAT ENTERPRISE LINUX ATOMIC HOST

openstack

redhat.

jboss.org

JBoss Tools

RED HAT JBOSS DEVELOPER STUDIO

Arquillian

RESTEasy

WildFly

HIBERNATE

VERT.X

FEEDHENRY™

Infinispan

AeroGear

Apache Camel

Qhawtio

fabric8

redhat