Cloud Native Camel Design Patterns

(Tips for Running Apache Camel on Kubernetes)

ApacheCon Europe

November 2016

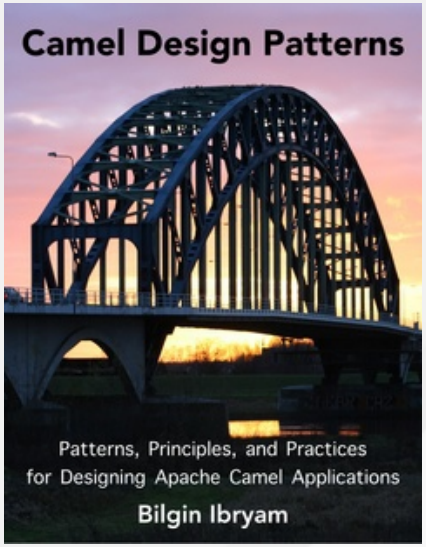
Bilgin Ibryam

redhat.

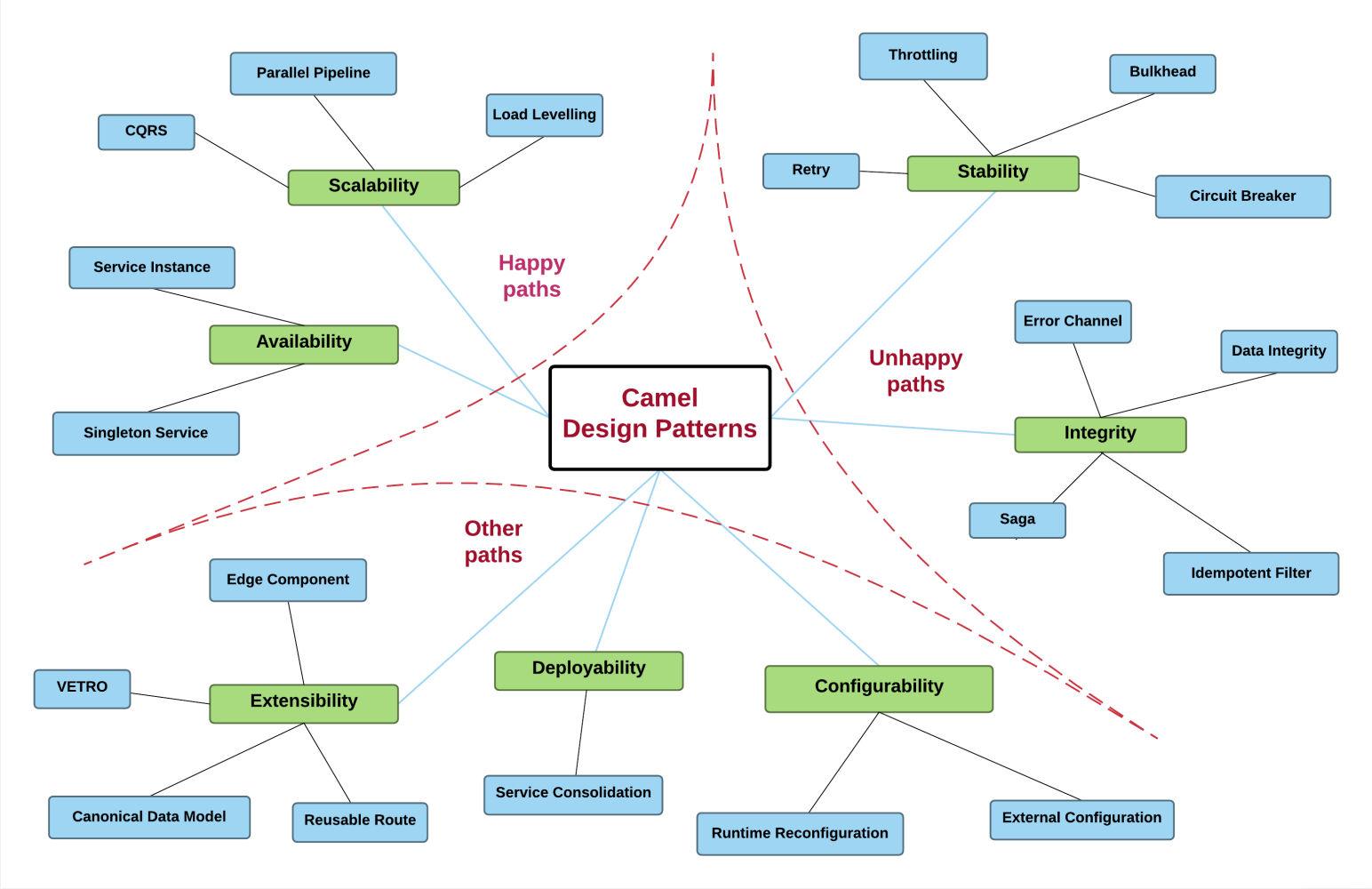
Bilgin Ibryam

- Twitter: @bibryam
- Email: bibryam@gmail.com
- Blog: http://ofbizian.com
- LinkedIn: http://www.linkedin.com/in/bibryam

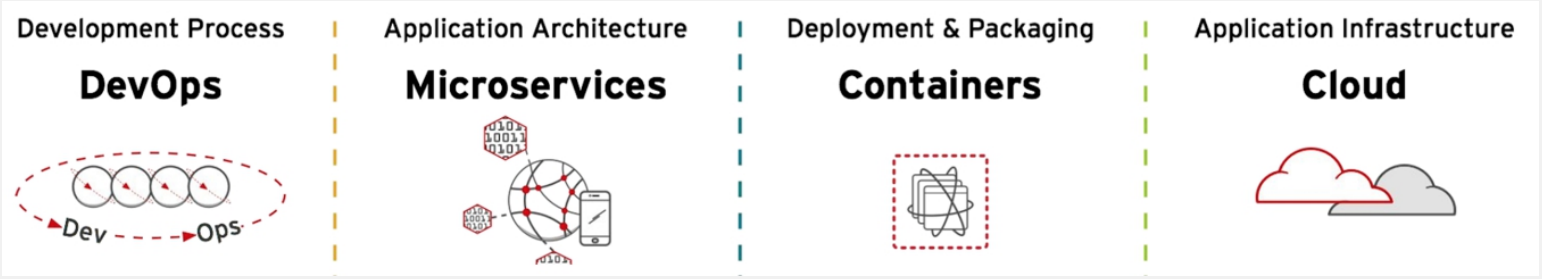
- Senior Middleware Architect at Red Hat UK
- Apache Isis Committer and PMC member
- Apache Camel Committer and PMC member
- Apache OFBiz Committer and PMC member
- Author of Camel Design Patterns (new)
- Author of Apache Camel Message Routing



Before Cloud Native



Trends in the IT Industry



- Application Infrastructure: Data Center, Hosted, **Cloud**
- Application Deployment: Physical, Virtual, **Containers**
- Application Architecture: Monolith, N-tier, **Microservices**
- Development Methodology: Waterfall, Agile, **DevOps**

And these trends do affect the way we design, develop and run Camel applications!

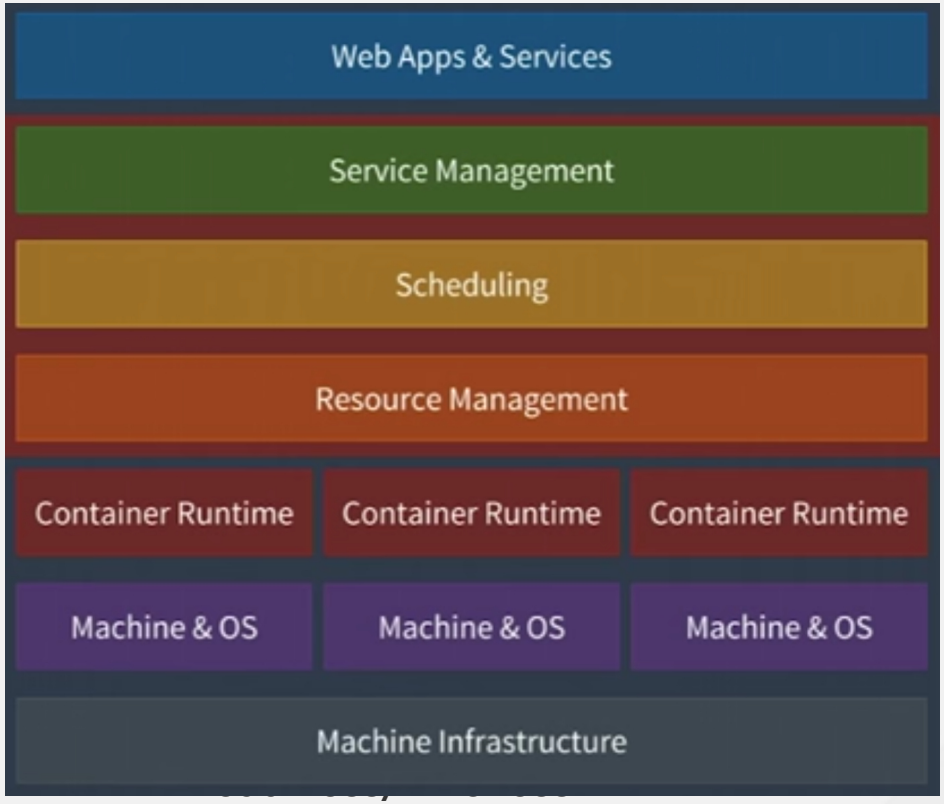
What is Cloud Native?

A cloud-native application is a distributed application that runs on a cloud infrastructure and is in its core scalable and resilient.

- **Web-scale** – originated from large cloud firms, such as Google, Amazon, Netflix, FB.
- **The twelve-factor apps** - a methodology for building apps for the Heroku platform.
- **13 factor apps, 14 factor apps** – 12 is not enough, nor is 13...
- **Cloud-aware or Cloud-ready** – cannot benefit from all characteristics of the cloud.
- **Cloud native** – a marketing term used by Cloud Foundry.
- **Cloud Native Computing Foundation** - https://cncf.io - part of the Linux Foundation.

Cloud Native Platforms

- Docker Swarm (Docker, Inc.)
- ECS (Amazon)
- Kubernetes (Google)
- OpenShift (Red Hat)
- Cloud Foundry (Pivotal/VMware)
- DC/OS (Mesosphere)
- Apache Mesos
- Nomad, Kontena, Rancher...

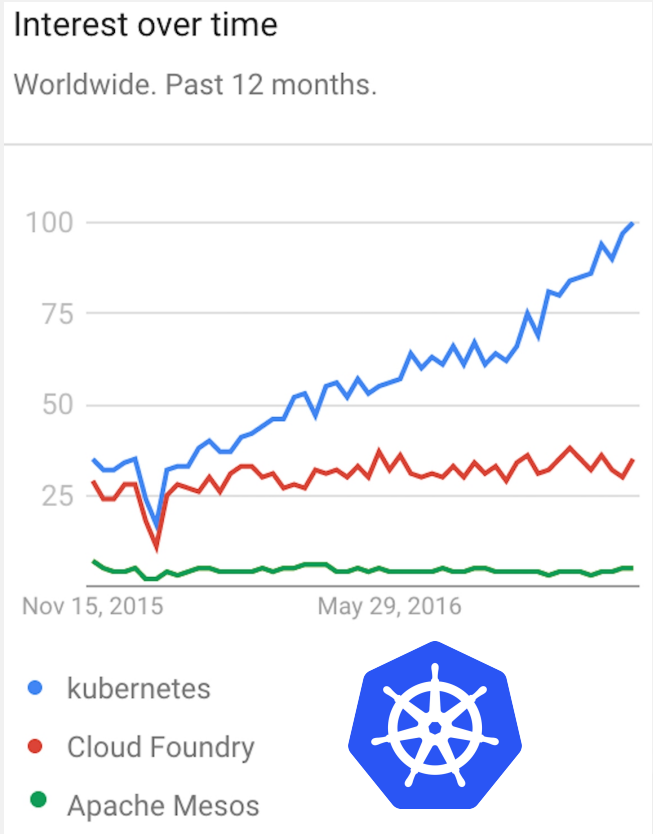


Container Orchestration Wars - by Karl Isenberg http://bit.ly/kube-war

Why Kubernetes?

Portable, extensible, self-healing, platform for automating deployment, scaling, and operations of containers.

- Technology - based on Google's Borg project
- Community - 1K contributors and 34K commits
- Open Source, Open Standards, part of CNCF
- Backed by large organizations



Growing Kubernetes Ecosystem

- Cloud providers: Azure, VMware, Openstack, Rackspace, CenturyLink
- Distros: CoreOS Tectonic, Mirantis Murano (OpenStack), RedHat Atomic, Hyper.sh, VMTurbo
- PaaS: RedHat OpenShift, Deis, Rancher, WSO2, Gondor/Kel, Apcera
- CD: Fabric8, Shippable, CloudBees, Solano
- Deployment: Kumoru, Redspread, Spinnaker
- Package managers: Helm, KPM
- Monitoring: Prometheus, Sysdig, Datadog
- Networking: Weaveworks, Tigera, OpenContrail
- Storage: NetApp, ClusterHQ
- Appliances: Redapt, Diamante



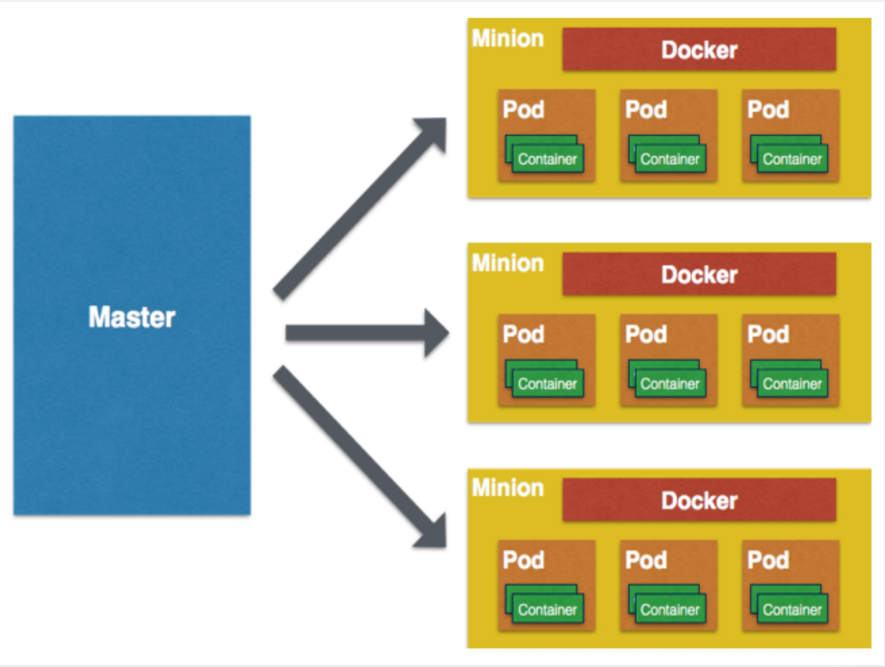



How to run Kubernetes

- **Installing from scratch** – kubeadm with some manual configuration
- **Kops sub project** – production grade Kubernetes on AWS
- **Kubernetes Anywhere** – GCE/AWS/Azure installation
- **Google Container Engine** – self service Kubernetes clusters
- **OpenShift by Red Hat** – managed Kubernetes & PAYG options
- **3rd party service providers** - @StackPointCloud, @AppsCodeHQ, @kclusterio
- **Local machine** – VM, Vagrant, minikube start
- **Java devs** - mvn fabric8:cluster-start (No Docker, VirtualBox or Vagrant required!)

Kubernetes Primitives for Developers

- How to run Kubernetes locally? → **Minikube**
- How to package apps? → **Docker/Rkt**
- What is the deployment unit? → **Pod**
- How to group artifacts? → **Labels**
- How to isolate resources? → **Namespaces**
- How to mange configs? → **ConfigMap/Secret**
- How to get storage? → **PersistentVolume**
- How to do service discovery & load balancing? → **Service & Route**
- How to update/rollback services? → **Deployment**



Deployment Patterns

Camel Runtime and Packaging

Service only packaging

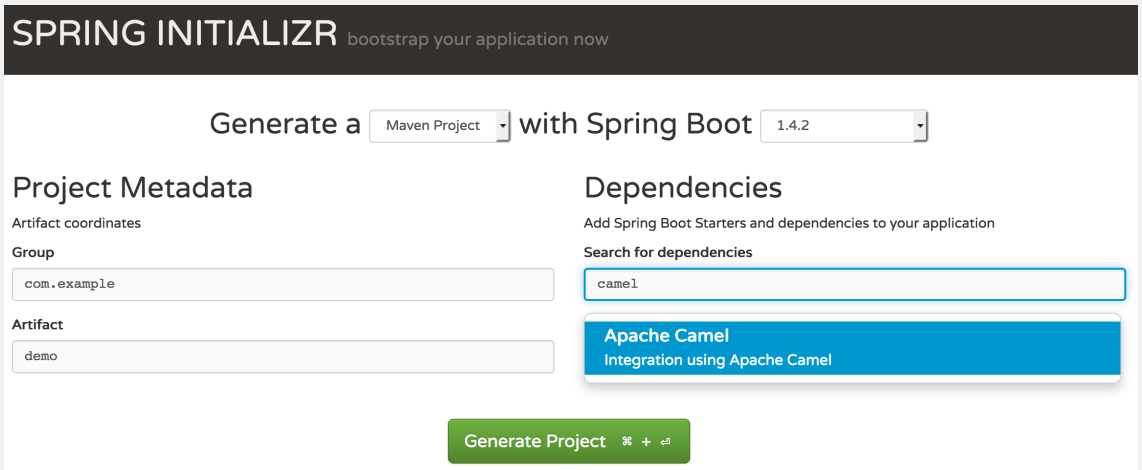
- Servlet container (Apache Tomcat) – **.war**
- Application server (WildFly) - **.ear**
- OSGI container (Karaf) - **.fab, .kar, feature.xml, Fuse Fabric profile**

Service and Runtime packaging - **.zip**

- Immutable Karaf distribution – for OSGI fans mainly
- WildFly-Swarm – for JEE shops (through WildFly-Swarm Camel)
- Standalone Java application
- Camel Boot - Spring Boot for Apache Camel

Spring Boot and Apache Camel

- Spring Boot integration reimplemented
- Spring Boot Starters
- Spring Boot Auto Configuration
- Spring Boot Health Check - leverage the Spring-Boot actuator module
- A new BOM (Bill of Material) – camel-spring-boot-dependencies
- Unified programing model with Spring annotations and Camel Java DSL



Dockerizing Camel Applications

Different workflows:

- Centralized platform based build
- Manual scripting with Dockerfile
- Multiple Maven plugins:
  - Alexec (92*), Wouterd(77*), spotify(735*)
  - Fabric8 maven docker plugin (472*)

Fabric8 docker-maven-plugin goals

Goal	Description
docker:start	Create and start containers
docker:stop	Stop and destroy containers
docker:build	Build images
docker:watch	Watch for doing rebuilds and restarts
docker:push	Push images to a registry
docker:remove	Remove images from local docker host
docker:logs	Show container logs
docker:source	Attach docker build archive to Maven project

Start Kubernetes: **mvn fabric8:cluster-start**

Build (artifacts and docker image): **mvn clean install docker:build**

Deploy a service to Kubernetes: **mvn fabric8:json fabric8:apply**

Talking to Kubernetes

How to tell Kubernetes to:

- Keep 3 instances of my-service up
- Use the command "/bin/echo", "hello", "world" to start
- Allocate 512MiB memory and one core for it
- Make port 80 accessible
- Set foo environment variable with bar value
- Mount configs with name my-service to location /my-service
- Mount a 5GB R/W persistent volume
- And for updates do rolling update by bringing only 30% of containers down
- And do canary release...

Application Descriptor Pattern

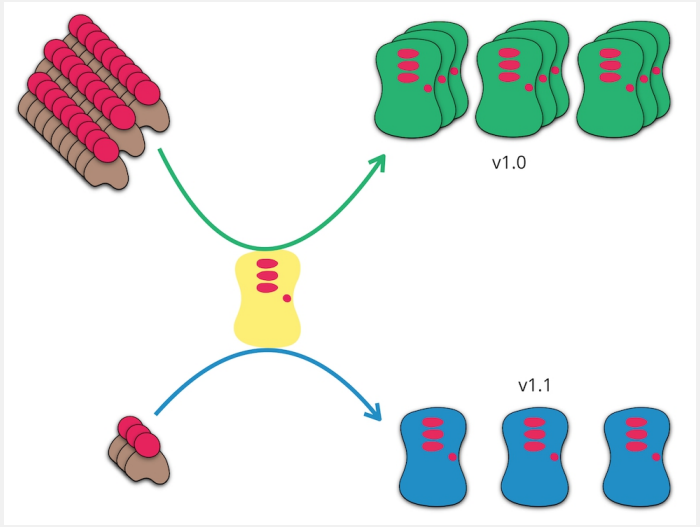
Every service requires a manifest/recipe/contract describing its prerequisites from the platform and runtime model.

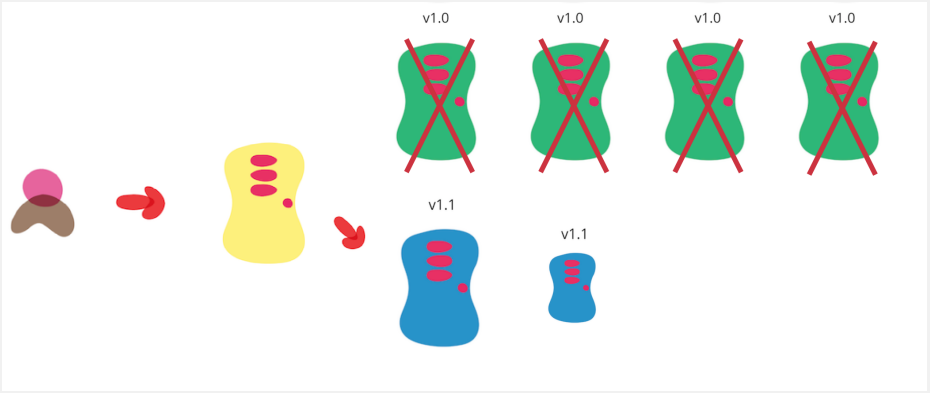
```
@KubernetesProvider
public KubernetesList create() {
  return new KubernetesListBuilder()
    .addNewReplicationControllerItem()
      .withNewMetadata()
        .withName("Hello-Controller")
      .endMetadata()
      .withNewSpec()
        .withReplicas(1)
        .addToSelector("component", "my-compo
        .withNewTemplate()
          .withNewSpec()
            .addNewContainer()
              .withName("my-container")
              .withImage("my/image")
            .endContainer()
          .endSpec()
        .endTemplate()
```

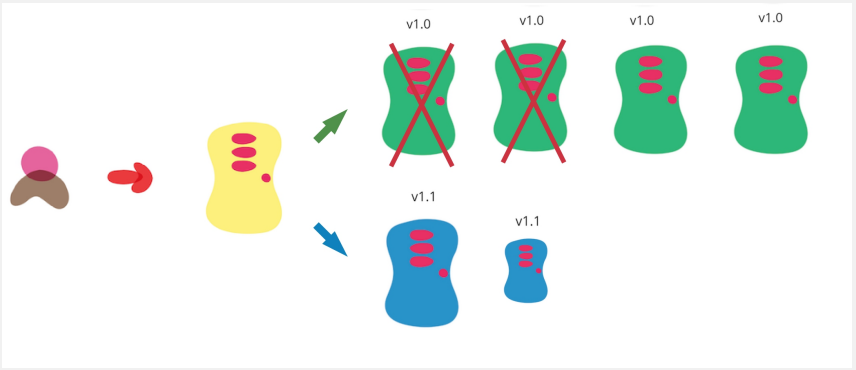
```
"apiVersion" : "v1",
"kind" : "ReplicationController",
"metadata" : {
  "annotations" : { },
  "labels" : {
    "container" : "java",
    "component" : "typesafe-kubernetes-dsl",
    "provider" : "fabric8",
    "project" : "typesafe-kubernetes-dsl",
    "version" : "1.0-SNAPSHOT",
    "group" : "quickstarts"
  },
  "name" : "typesafe-kubernetes-dsl"
},
"spec" : {
  "replicas" : 1,
  "selector" : {
    "container" : "java",
```

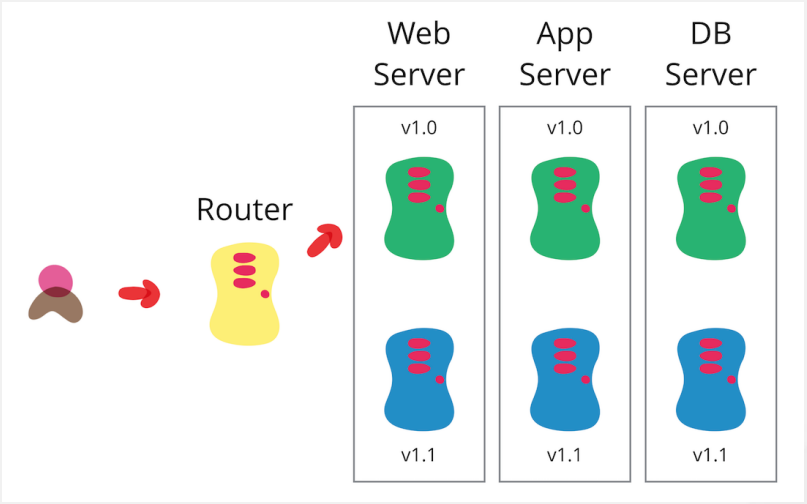
mvn fabric8:json fabric8:apply

Kubernetes Deployments









Health Check Pattern

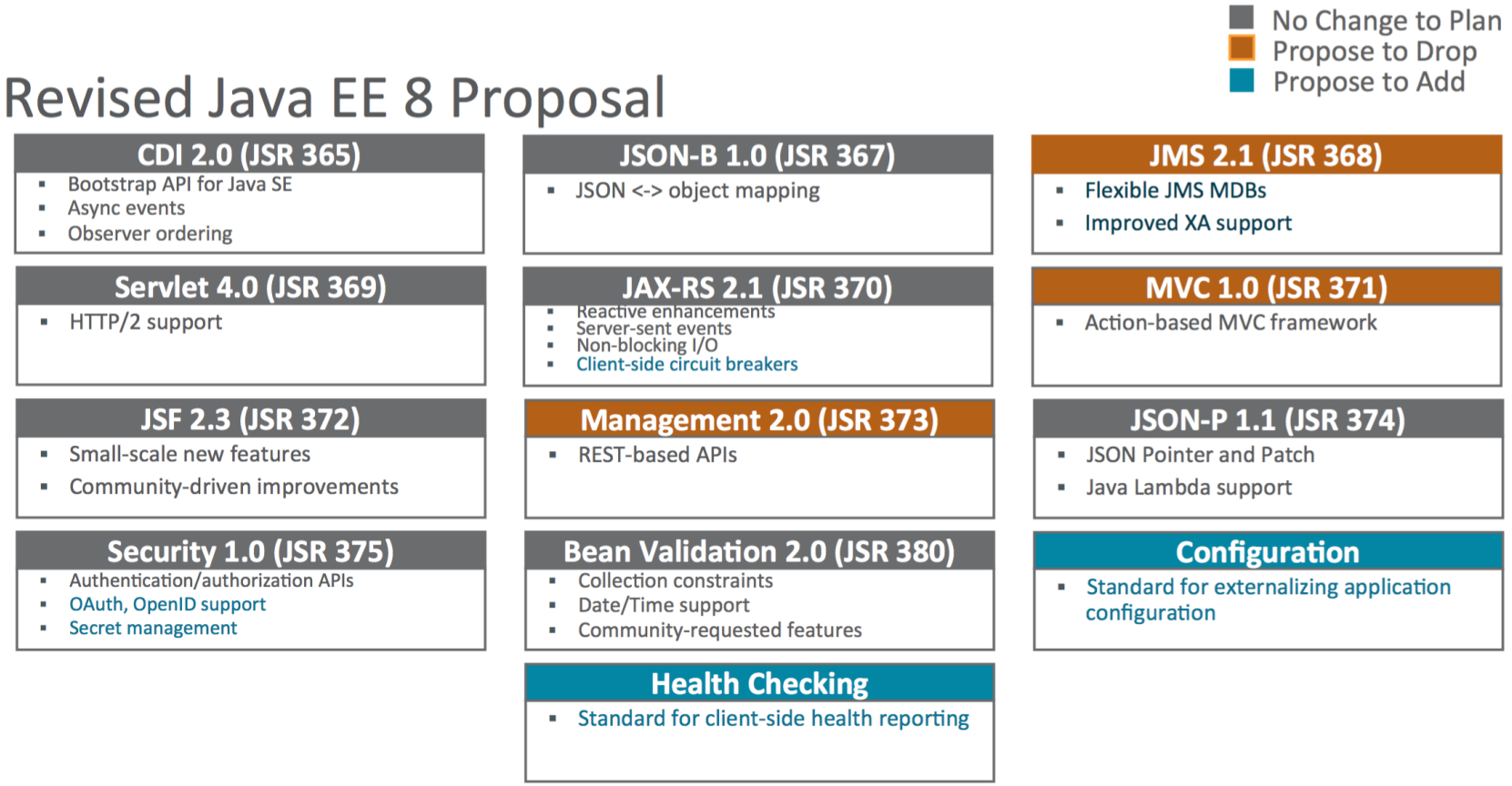
In order to be a good cloud native citizen, every app should be able to report its health status.

- Process Health Check – checks for the process to be running
- Application Readiness Health Checking
- Application Liveness Health Checking
  - HTTP Health Checks – expects return code between 200-399
  - Container Exec – expects return code 0
  - TCP Socket – expects to open socket connection

```
livenessProbe:
  exec:
    command:
    - cat
    - /tmp/health
  initialDelaySeconds: 15
  timeoutSeconds: 1
```

```
livenessProbe:
  httpGet:
    path: /healthz
    port: 8080
    httpHeaders:
      - name: X-Custom-Header
        value: Awesome
  initialDelaySeconds: 15
  timeoutSeconds: 1
```

Health Check Pattern



Lifecycle Hooks

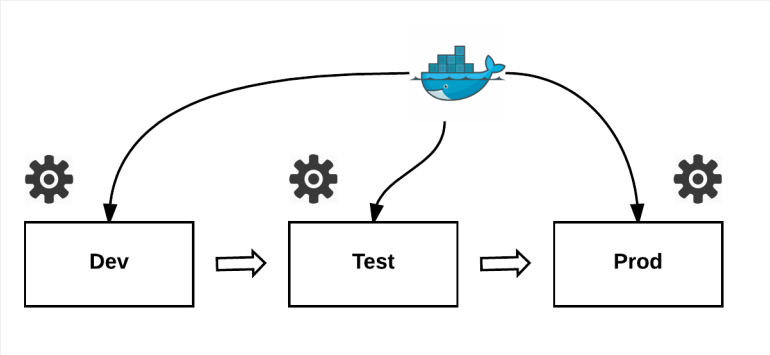
Allows applications to do graceful shutdown and startup.

- To stop a pod, Kubernetes will send **SIGTERM** 30 seconds before **SIGKILL**
- **PreStop** lifecycle hook executed prior to sending SIGTERM.
- **PostStart** sent immediately after a container is created.
  - Has the same formats as livenessProbe/readinessProbe and has “at least once” guarantee.
- **Termination message** - /dev/termination-log

```
lifecycle:
  postStart:
    exec:
      command:
        - "touch"
        - "/var/log/lifecycle/post-start"
  preStop:
    httpGet:
      path: "/abort"
      port: 8080
```

Application Configuration Pattern

The bad news: you have externalize everything that is environment specific.



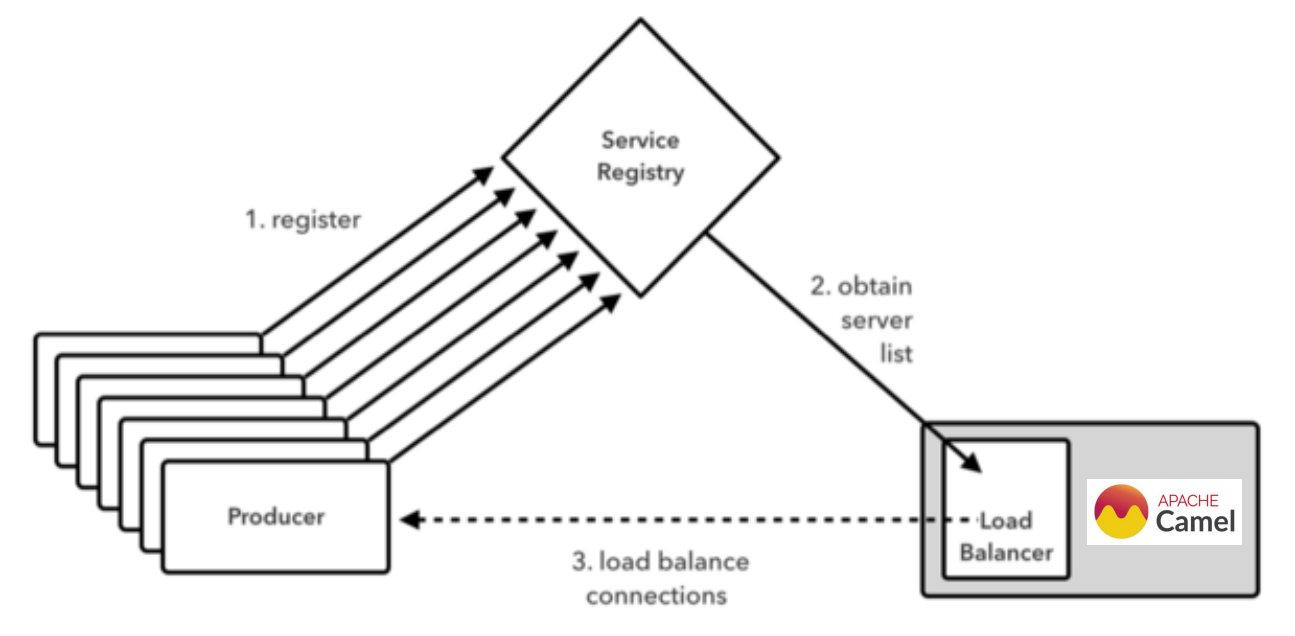
The good news: there is less things to externalize.

```
<route>
  <from uri="file:/usr/share/orders"/>
  <to uri="bean:magicProcessor"/>
  <to uri="https4://${CUSTOMER_SERVICE_HOST}:${CUSTOMER_SERVICE_PORT}"/>
</route>

<route>
  <from uri="sftp://{{HOST}}:1022/root?username={{USER}}&password={{PASS}}"/>
  <to uri="bean:processFile"/>
</route>
```

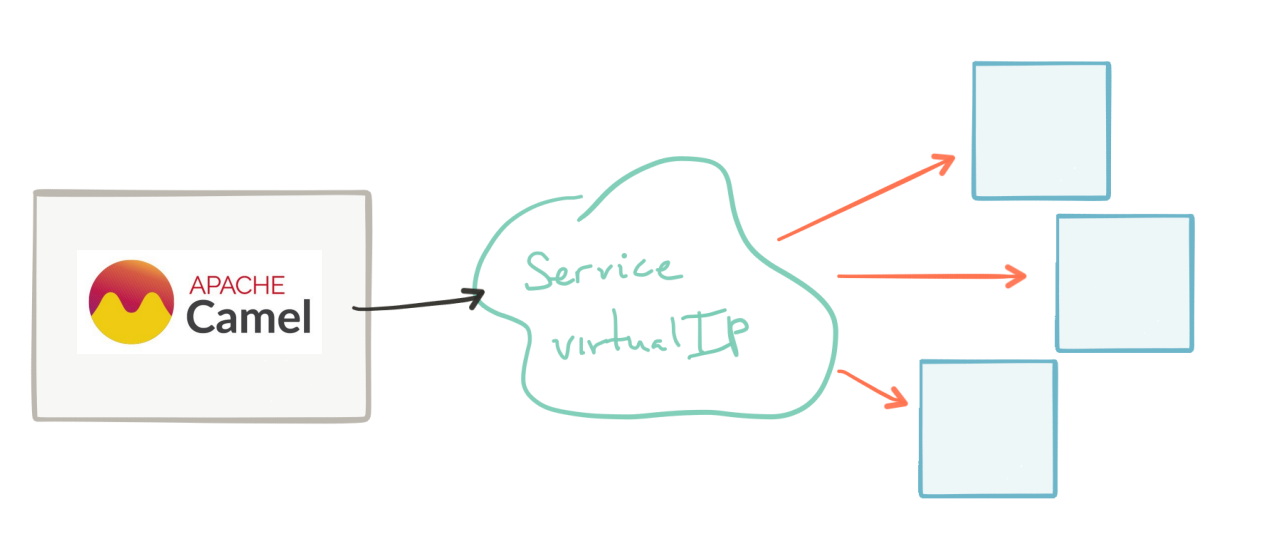
Service Discovery & Load Balancing

Client side – on the JVM



Service Discovery & Load Balancing

Provided by the platform

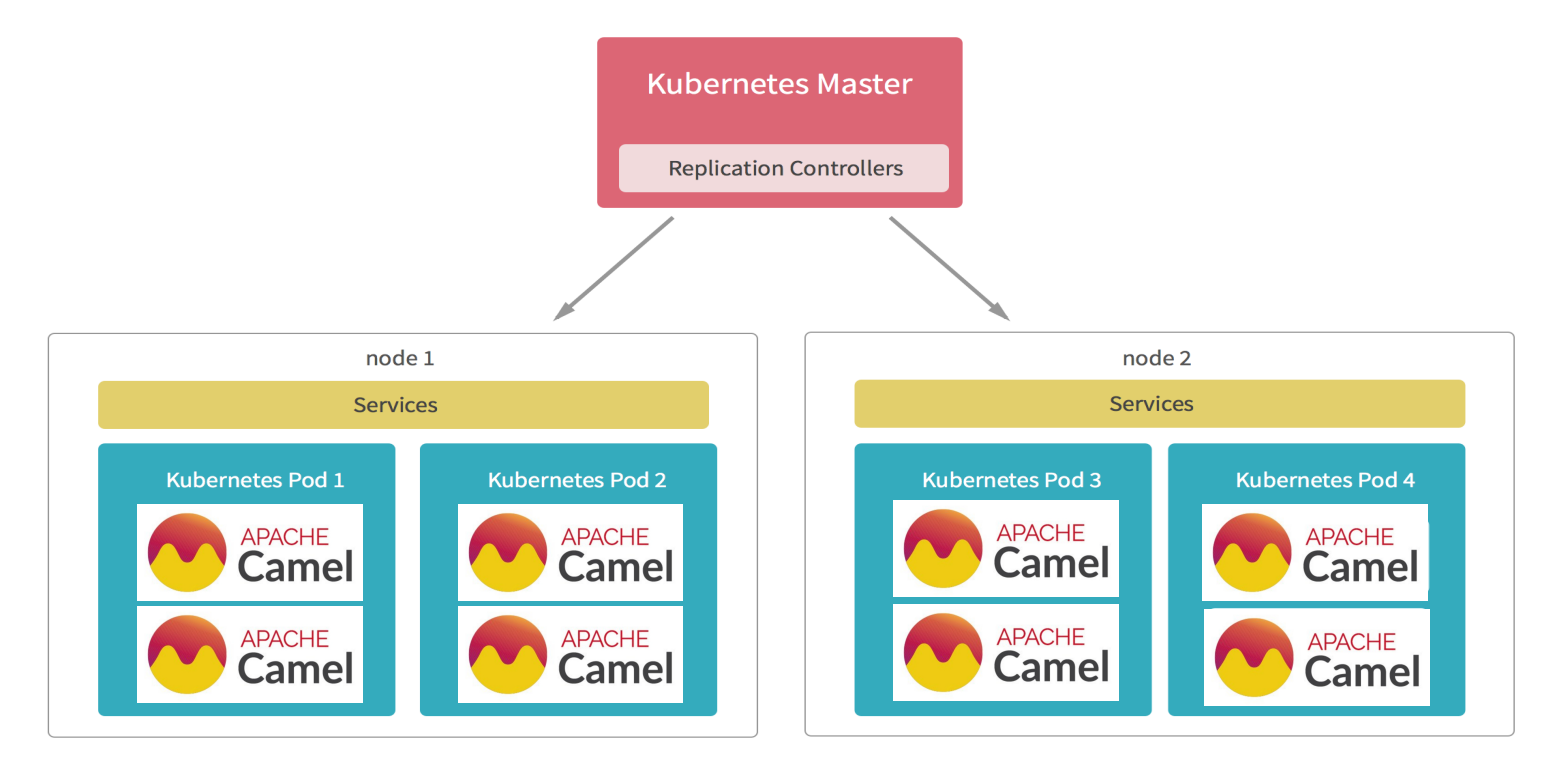



```
<route>
  <from uri="file:/usr/share/orders"/>
  <to uri="https4://${CUSTOMER_SERVICE_HOST}:${CUSTOMER_SERVICE_PORT}"/>
</route>

<route>
  <from uri="file:/usr/share/orders"/>
  <to uri="https4://CUSTOMER/"/>
</route>
```

Service Instance

How to accommodate increasing workloads?



Service Instance

Areas to consider before horizontally scaling a Camel application.

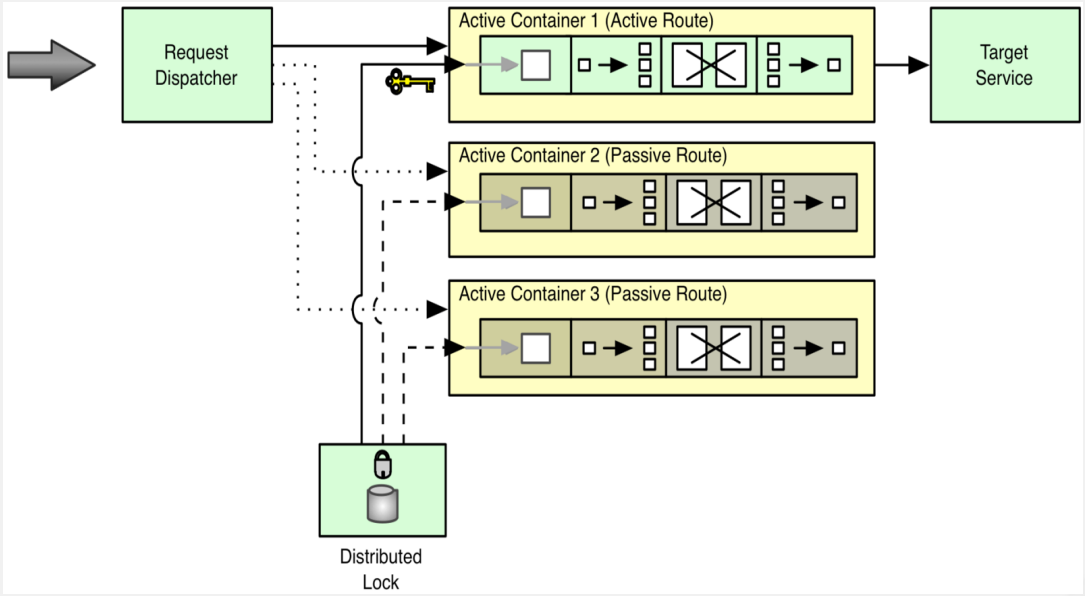
- **Service state:** load balancer, circuit breaker, resequencer, sampler, throttler, idempotent consumer and aggregator are stateful EIPs!
- **Request dispatcher:** Messaging, HTTP, file consumption (what about locking?)
- **Message ordering:** exclusive consumer, message groups, consumer priority, message priority, virtual topics
- **Singleton service requirements:** for batch jobs, and concurrent polling
- **Other** resource contention and coupling considerations

Singleton Service Pattern

How to ensure that only a single instance of a service is running?

JVM based:

- Karaf
- ActiveMQ
- JBoss HA



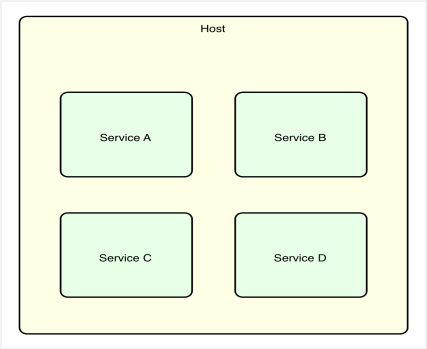
Camel based:

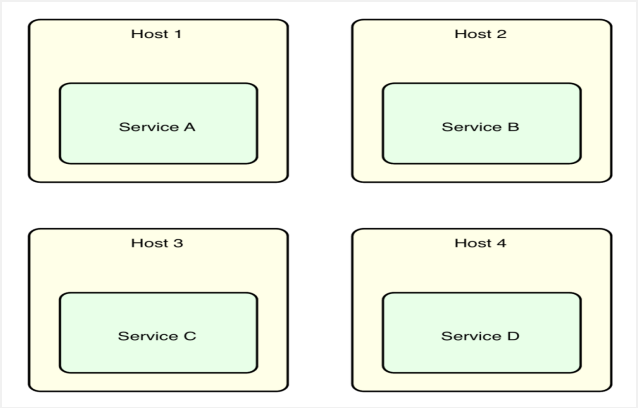
- Quartz, ZooKeeper, JGroups
- JBoss Fuse Master Component
- Use the database as a lock
- Exclusive consumers in ActiveMQ

Forget about all of these options, and create a Kubernetes pod with 1 replica.

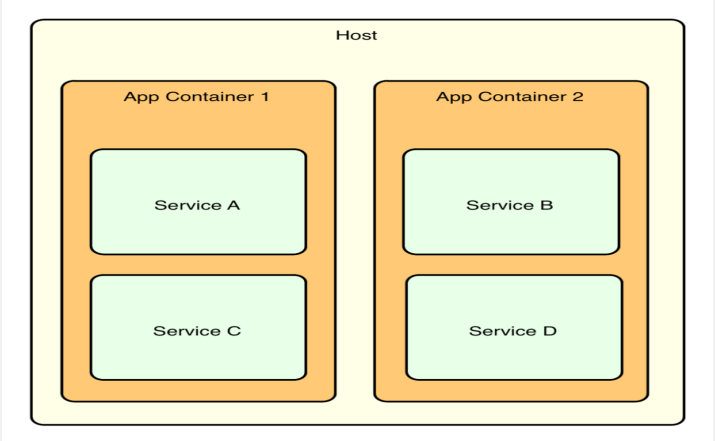
Service Consolidation Pattern

Forget about all these service placement principles...





- Single Service per Host
- Multiple Services per Host
- Shared Application Container(s)



Service Consolidation Pattern

...and trust Kubernetes Scheduler

- Policies driven
- Predicates and Priorities
- Topology-aware
- Extensible
- ServiceAffinity Predicate
- ServiceAntiAffinity Priority

```
{
 "kind" : "Policy",
 "version" : "v1",
 "predicates" : [
        {"name" : "PodFitsPorts"},
        {"name" : "PodFitsResources"},
        {"name" : "NoDiskConflict"},
        {"name" : "MatchNodeSelector"},
        {"name" : "HostName"}
 ],
 "priorities" : [
        {"name" : "LeastRequestedPriority", "weight" : 1},
        {"name" : "BalancedResourceAllocation", "weight" : 1},
        {"name" : "ServiceSpreadingPriority", "weight" : 1}
 ]
}
```

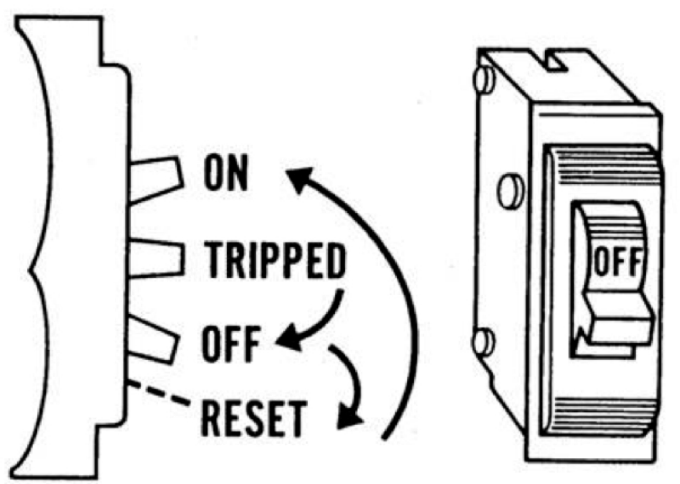
Sample topological levels:

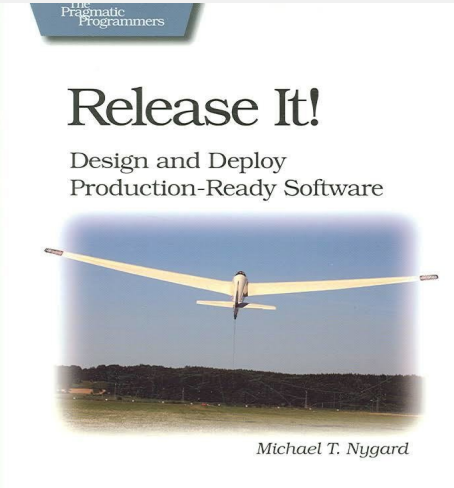
region (affinity) → zone (affinity) → rack (anti-affinity)

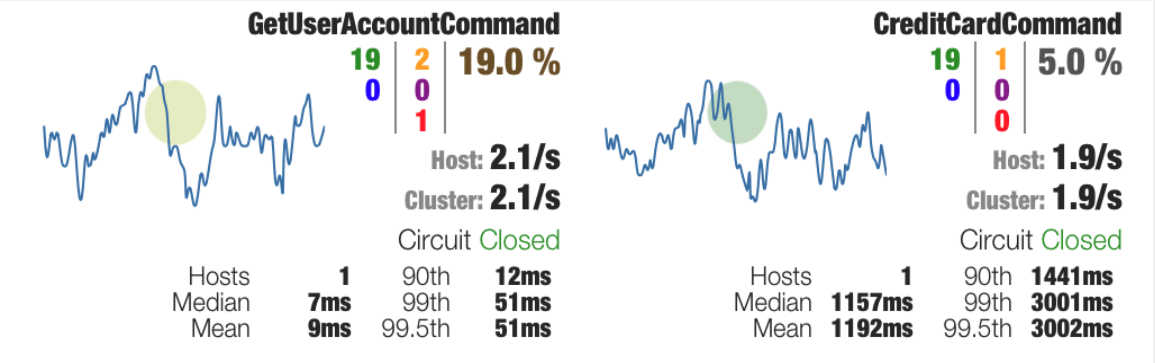
Error Handling Patterns

Circuit Breaker Pattern

Improves the stability and the resilience of a system by guarding integration points from cascading failures and slow responses.







Circuit Breaker Pattern

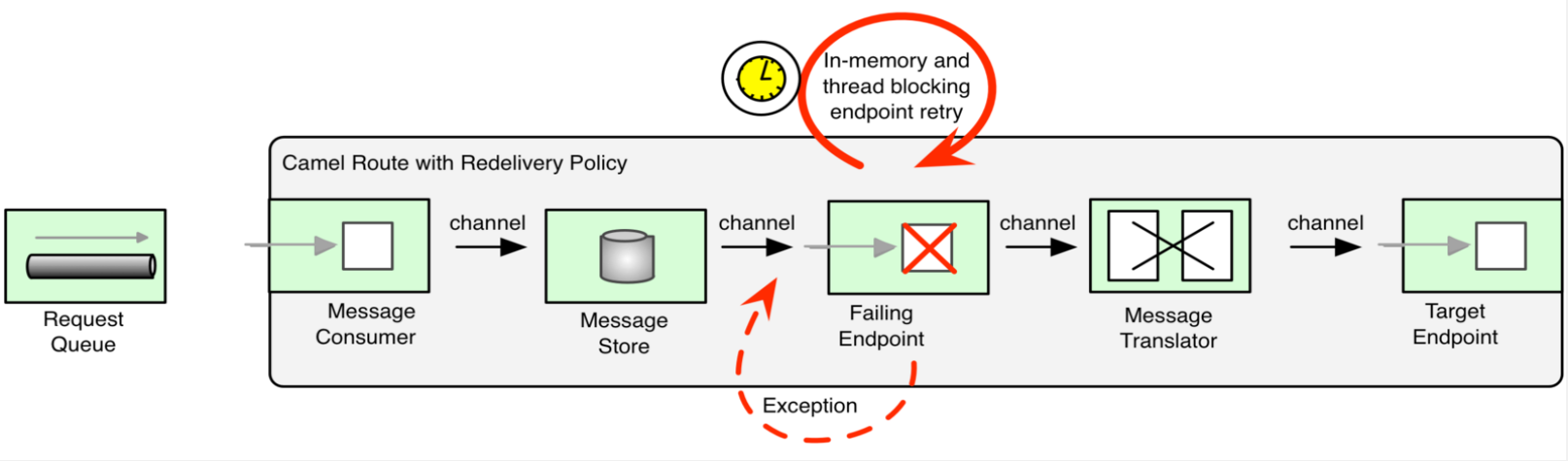
Two Circuit Breaker Implementations in Camel 2.18

```
<route>
  <from uri="direct:start"/>
  <loadBalance>
      <circuitBreaker threshold="2" halfOpenAfter="1000">
          <exception>MyCustomException</exception>
      </circuitBreaker>
      <to uri="mock:result"/>
  </loadBalance>
</route>
```

```
@Override
public void configure() {
    from("timer:trigger?period=1s")
        .log(" Client request: ${body}")
        .hystrix()
            .to("http://localhost:9090/service1")
        // use onFallback() to provide a repsonse message immediately
        // use onFallbackViaNetwork() when there is a 2nd service call
        .onFallbackViaNetwork()
            .to("http://localhost:7070/service2")
        .end()
        .log("Client response: ${body}");
}
```

Retry Pattern

Camel RedeliveryPolicy



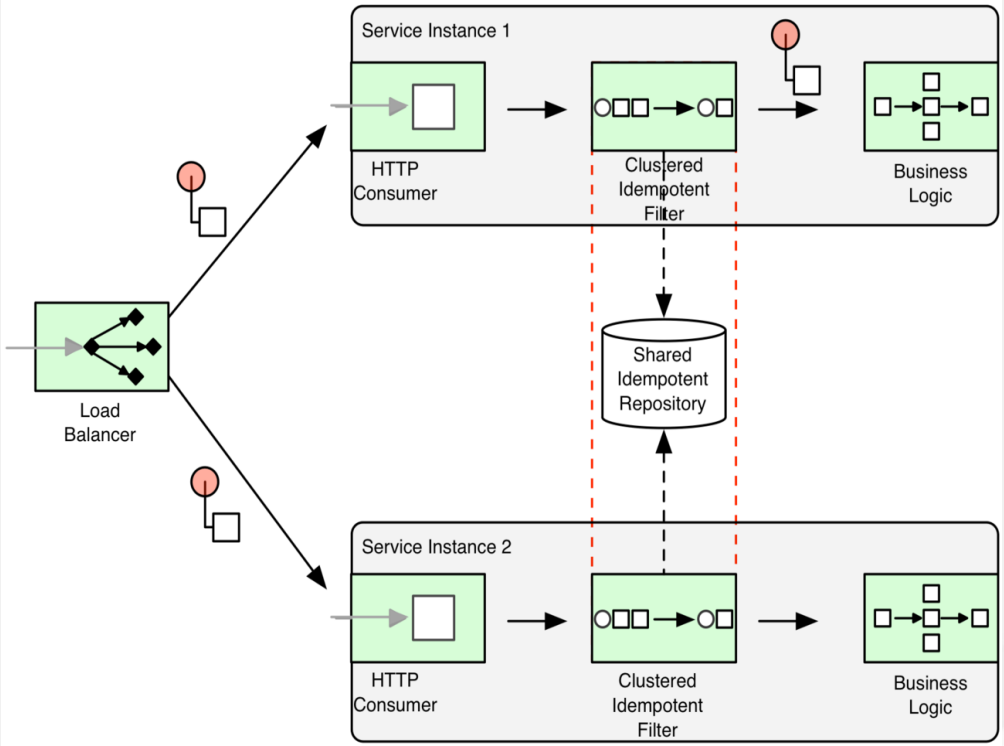
- The most well known retry mechanism in Camel
- Retries only the failing endpoint
- Fully in-memory
- Thread blocking behavior by default
- Can be asynchronous
- Good for small number of quick retries (in milliseconds)

Idempotent Filter Pattern

How to filter out duplicate messages and ensure only unique messages are passed through?

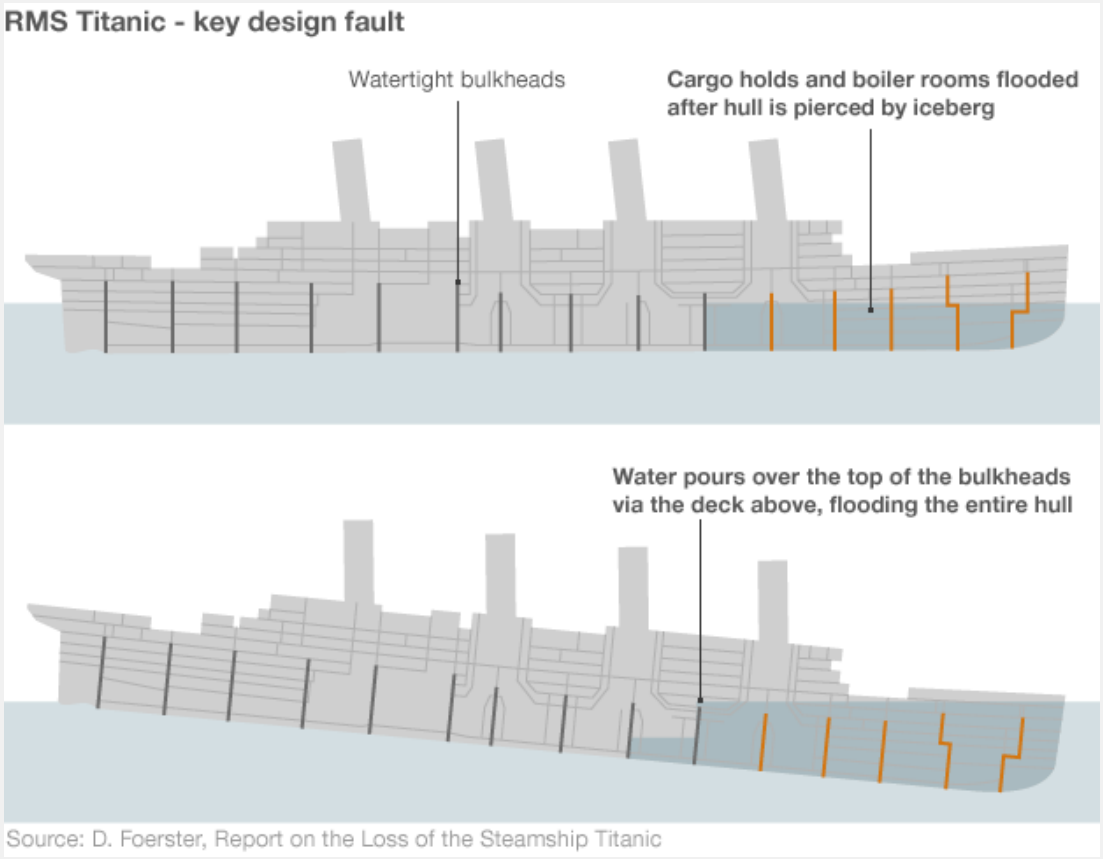
Distributed Idempotent Filters:

- Infinispan
- Hazelcast
- Redis
- RDBS

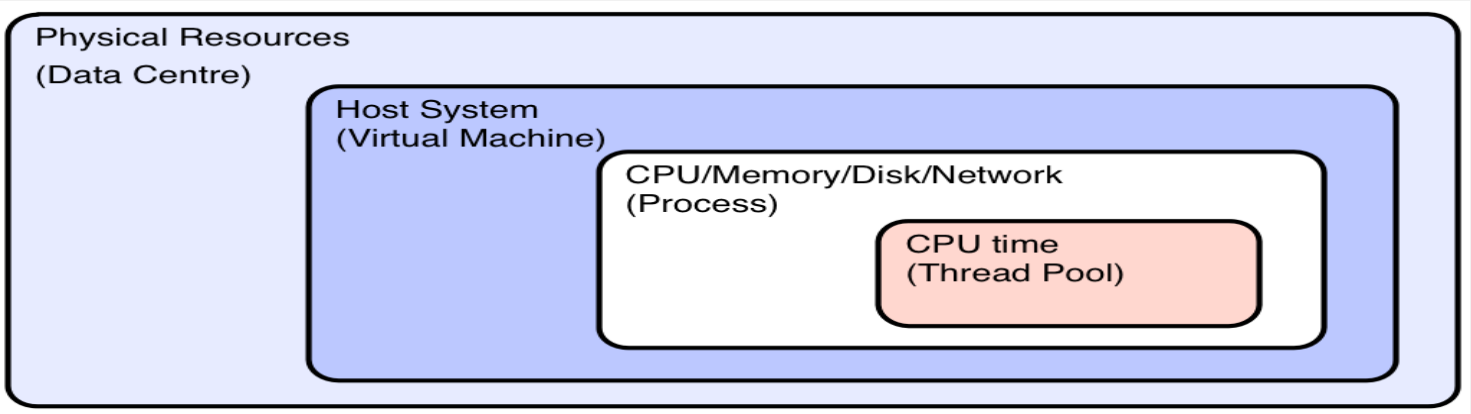


Bulkhead Pattern

Enforces resource partitioning and damage containment in order to preserve partial functionality in the case of a failure.



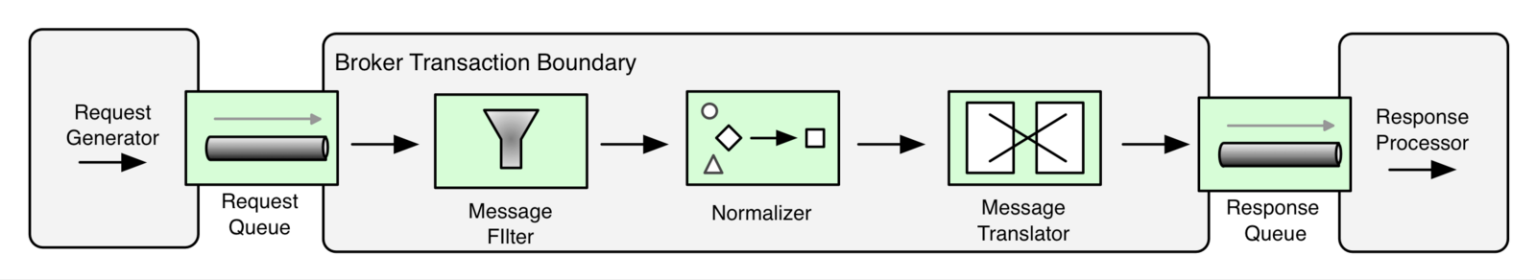
Bulkhead Pattern



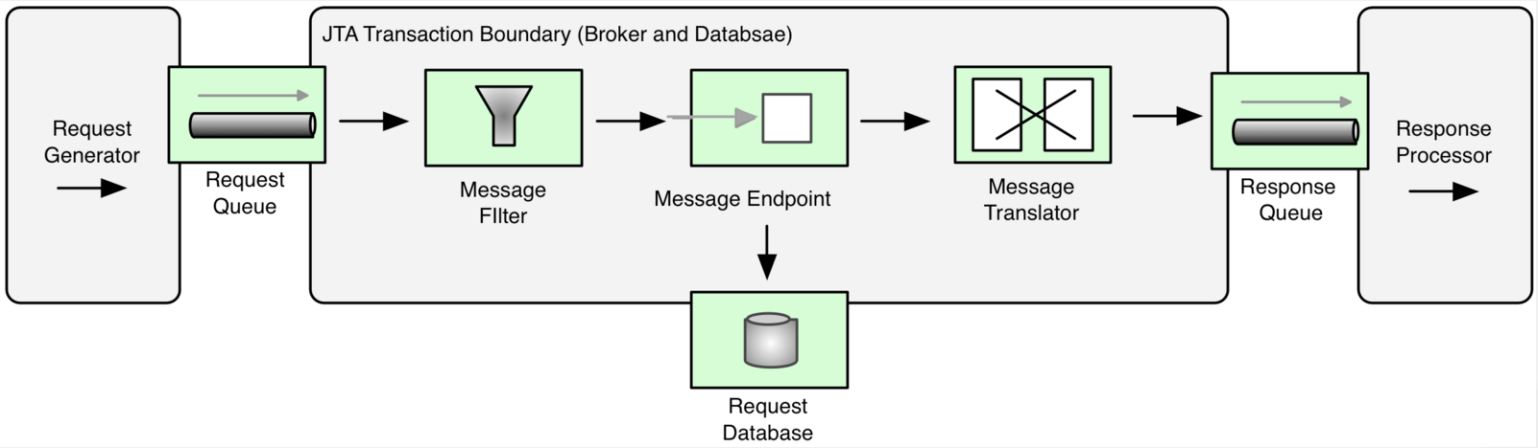
- **Level 1**: Kubernetes scheduler will ensure cross DC spreading of services.
- **Level 2**: Kubernetes scheduler will ensure cross VM spreading of services.
- **Level 3**: Use MSA and Bounded Context principles to identify services.
- **Level 4**: Configure Camel multi-threaded elements, such as: Delayer, Multicast, Recipient List, Splitter, Threads, Throttler, Wire Tap, Polling Consumer, ProducerTemplate, and OnCompletion, Circuit Breaker, Async Error Handler.

Transactions

Local transactions

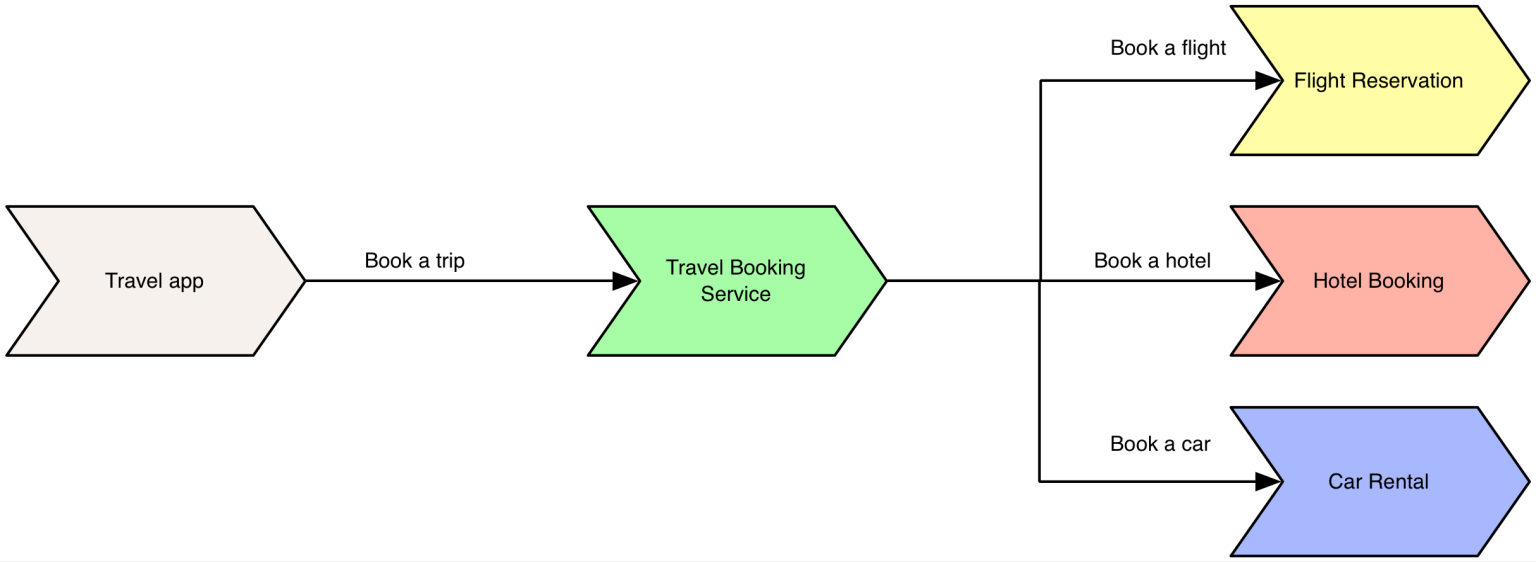


Global transactions



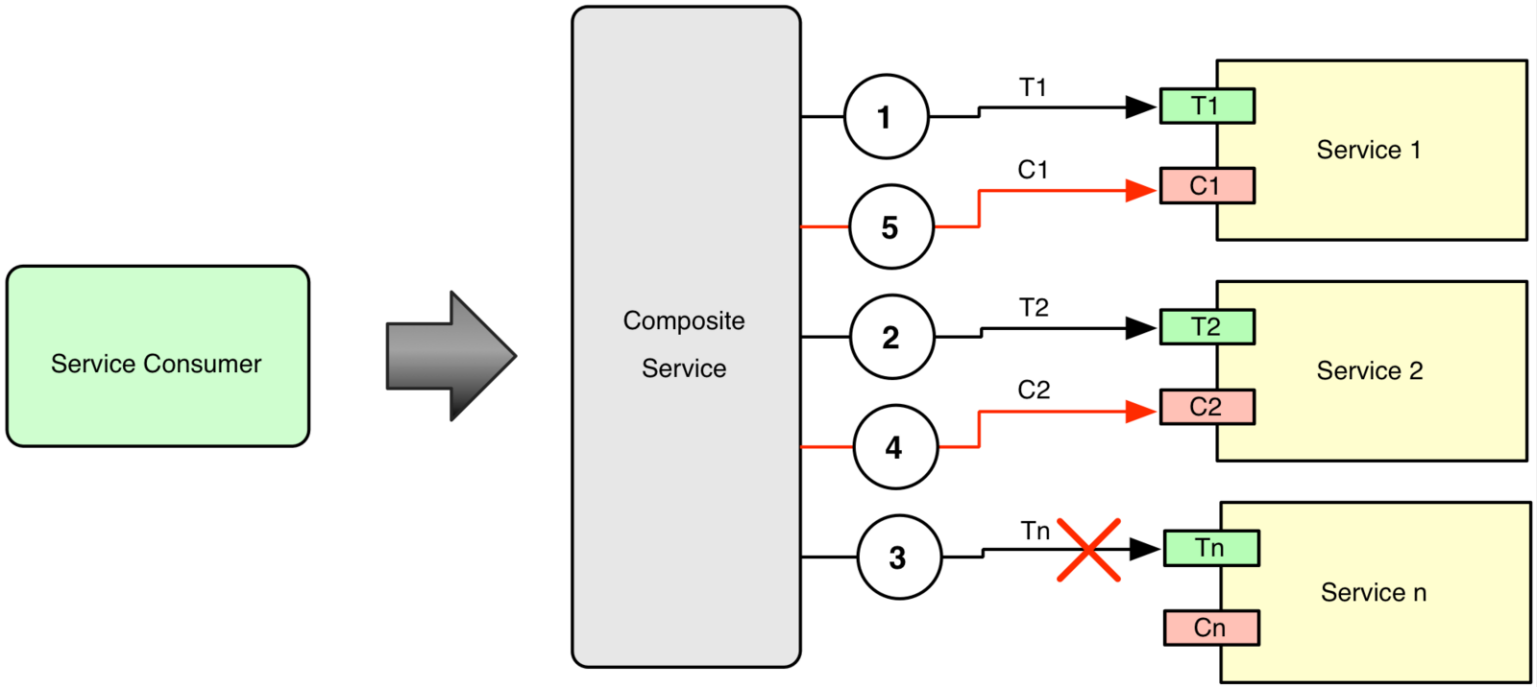
Saga Pattern

How to avoid distributed transactions and ensure data consistency?



Saga Pattern

Ensures that each step of the business process has a compensating action to undo the work completed in the case of partial failures.



New Patterns

Batch Jobs on the JVM

Camel batch support :

- Camel timer component – based JDK Timer
- Camel Scheduler component – based on JDK ScheduledExecutorService
- Camel Quartz component – based on Quartz Scheduler 1.x
- Camel Quartz2 component – based on Quartz Scheduler 2.x
- Polling consumers

Limitations:

- **Resource consumption** – small HA cluster example: 2x VMs, 2x JVMs, with monitoring, metrics, logs aggregation agents and clustering solution.
- **Clustering and HA** – 3 ZooKeeper servers, or a shared relational database for state and locking.
- **Fixed topology** – cannot move or scale jobs dynamically.

Kubernetes Scheduled Job

A scheduled job creates one or more pods, once or repeatedly, and ensures that a specified number of them successfully terminate.

Features

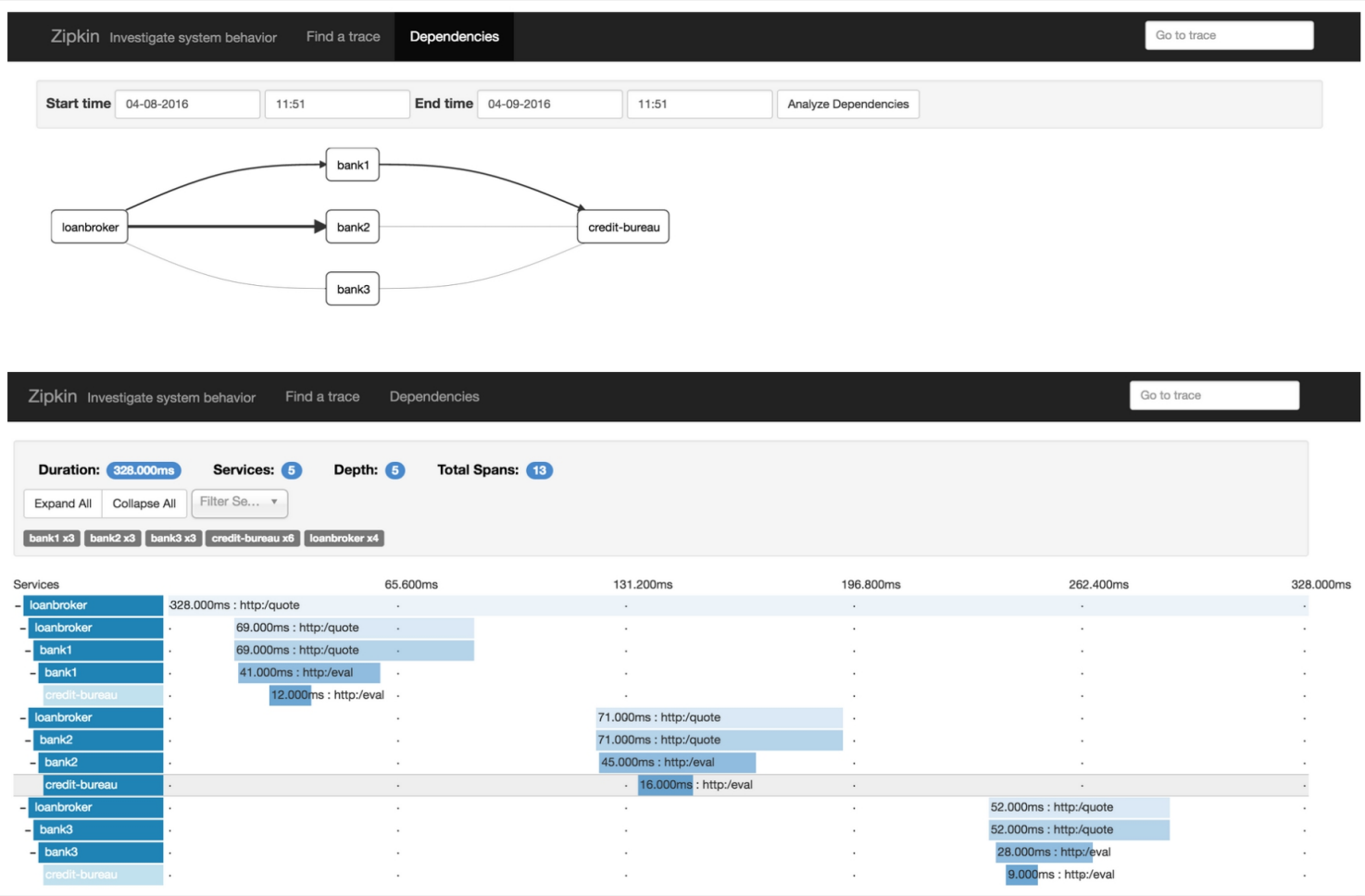
- Non-parallel Jobs
- Parallel Jobs with a fixed completion
- Parallel Jobs with a work queue
- Concurrency Policy

Prerequisites

- Idempotent jobs
- Meaningful exit code
- Clean Job start

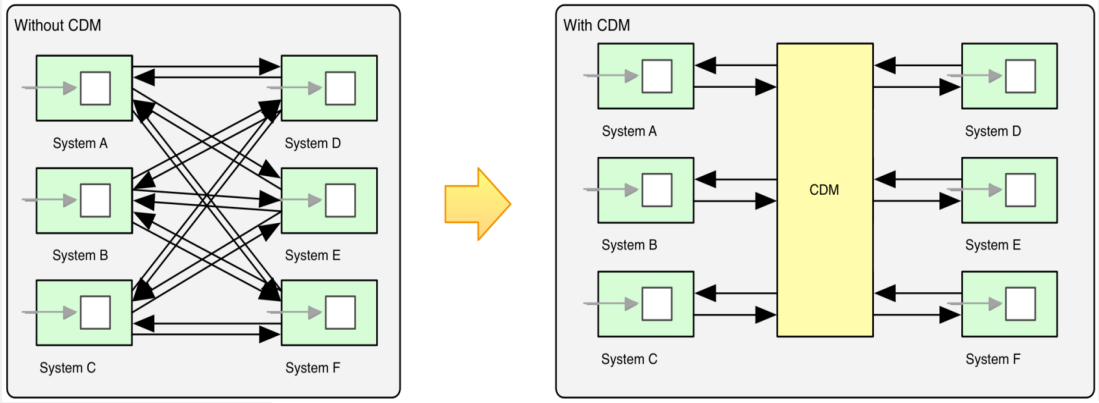
```
apiVersion: batch/v2alpha1
kind: ScheduledJob
metadata:
  name: hello
spec:
  schedule: 0/1 * * * ?
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: hello
            image: busybox
            args:
            - /bin/sh
            - -c
            - date
          restartPolicy: OnFailure
          concurrencyPolicy: Forbid
```

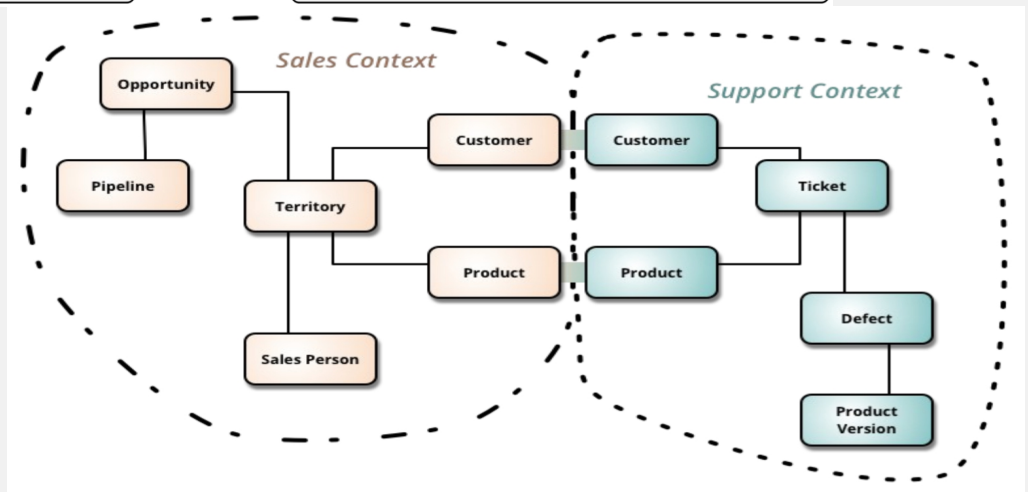
Tracing Pattern



Canonical Data Model

CDM is daed. Long live Bounded Context





How are Integration Patterns Evolving?

Less Relevant (or provided by the platform)	Not Changed (still relevant)	More Important (or a new concern)
Canonical Data Model	Load Leveling	Bounded Context
Distributed/Global Transactions	Bulkhead, Error Channel	Saga, Compensating Transactions
Batch Job on JVM, Singleton Service on JVM	Parallel Pipeline	Circuit Breaker, CQRS
Load Balancing on JVM	Runtime Reconfiguration, External Configuration	Service Instance, External Configuration
Service Discovery on JVM	VETRO	Tracing, Health Check
Service Consolidation	Data Integrity	Policy Driven Scheduling
Reusable Route	Monitoring	Retry, Idempotent Filter

5 Takeaways from this Session

- ✔ Kubernetes is awesome.
- ✔ Kubernetes is the best place to run Apache Camel applications.
- ✔ With Kubernetes there are less concerns for developers to worry about.
- ✔ You need to write even more resilient and scalable services now.
- ✔ Don't reinvent the wheel, use Fabric8 Cloud Native tooling.

Q & A

Kubernetes http://kubernetes.io/

Fabric8 https://fabric8.io/

Camel Design Patterns http://bit.ly/camel-patterns/

@bibryam

redhat.