Cloud Native Apps with Spring Cloud

Spencer Gibb
twitter: @spencerbgibb
email: sgibb@pivotal.io

Dave Syer
twitter: @david_syer
email: dsyer@pivotal.io


Pivotal


CLOUD FOUNDRY


spring io


RabbitMQ
Messaging that just works


GEMFIRE®


Greenplum


PIVOTAL LABS


redis


BIGTOP

Cloud Native

- Distributed
- Automated
- Organizational
- Anti-fragile
- Replaceable

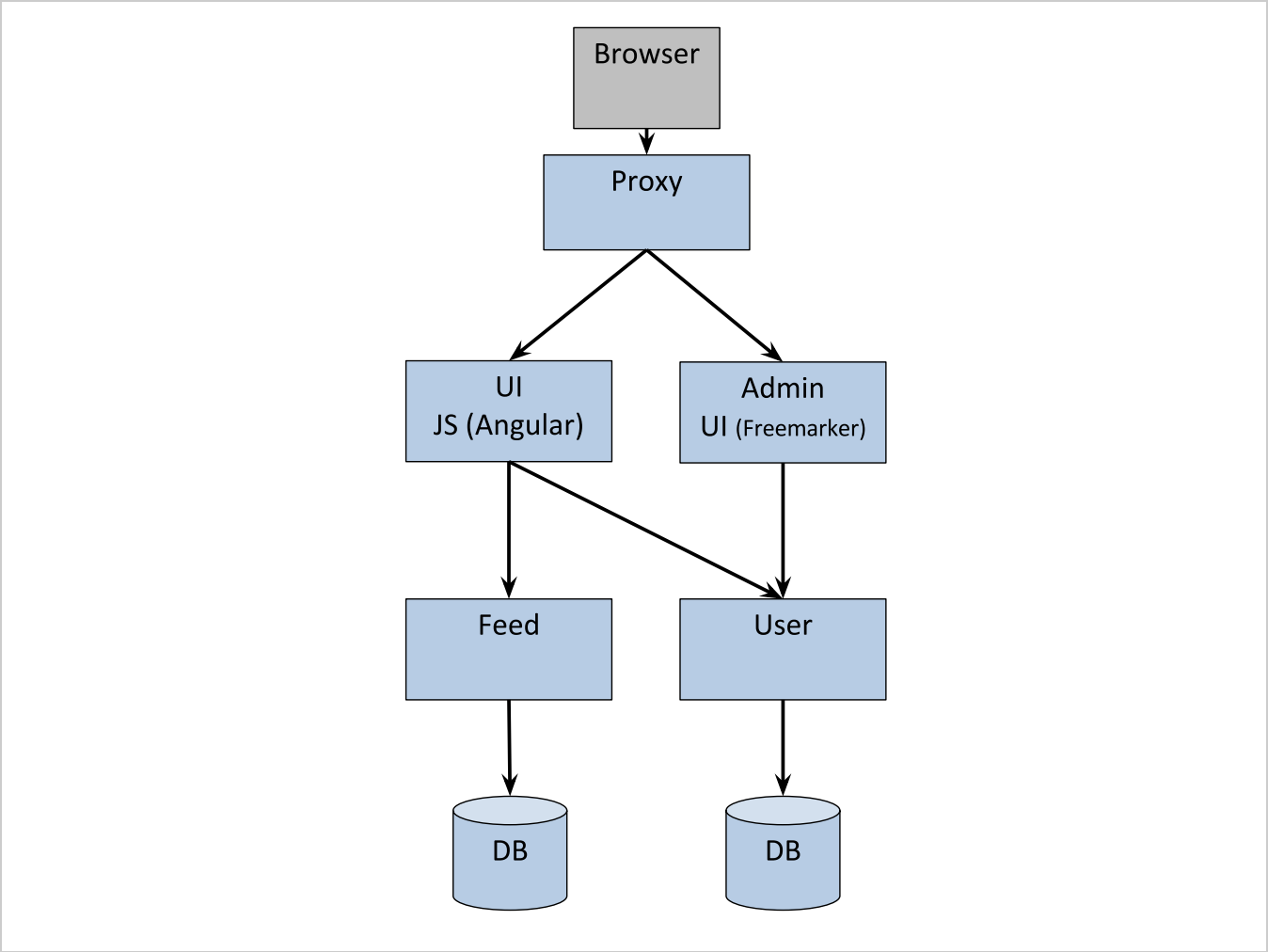
I like 'cloud native' better than microservice. I think its more descriptive and doesn't have the awkwardness of having 'micro" in the name.

Cloud Native with Spring Boot

It needs to be super easy to implement and update a service:

```
@RestController
class ThisWillActuallyRun {
  @RequestMapping("/")
  String home() {
    Hello World!
  }
}
```

Example Distributed System: Minified



No Man (Microservice) is an Island

It's excellent to be able to implement a microservice really easily (Spring Boot), but building a system that way surfaces "non-functional" requirements that you otherwise didn't have.

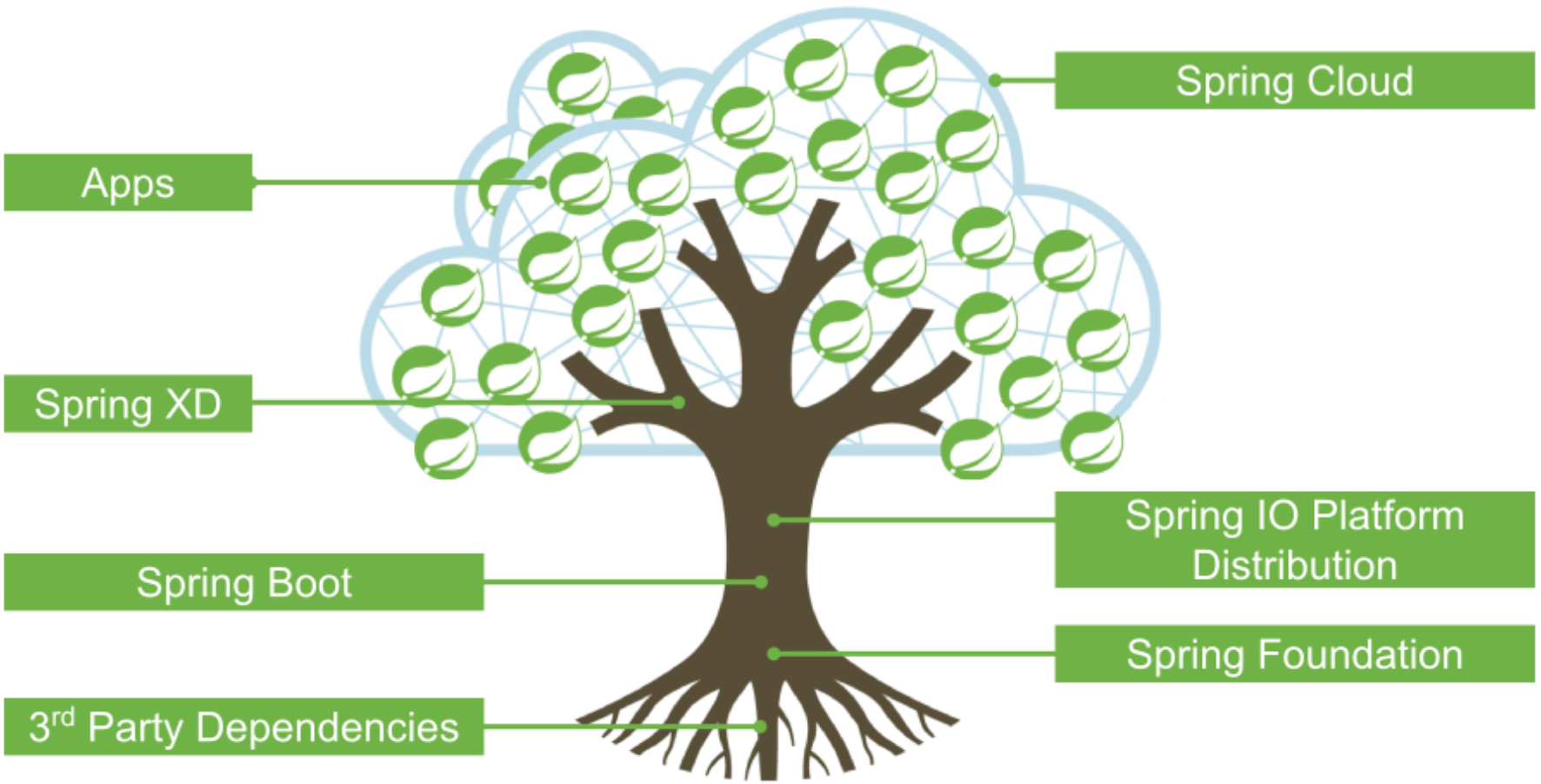
There are laws of physics that make some problems unsolvable (consistency, latency), but brittleness and manageability can be addressed with *generic*, *boiler plate* patterns.

Emergent features of cloud native systems

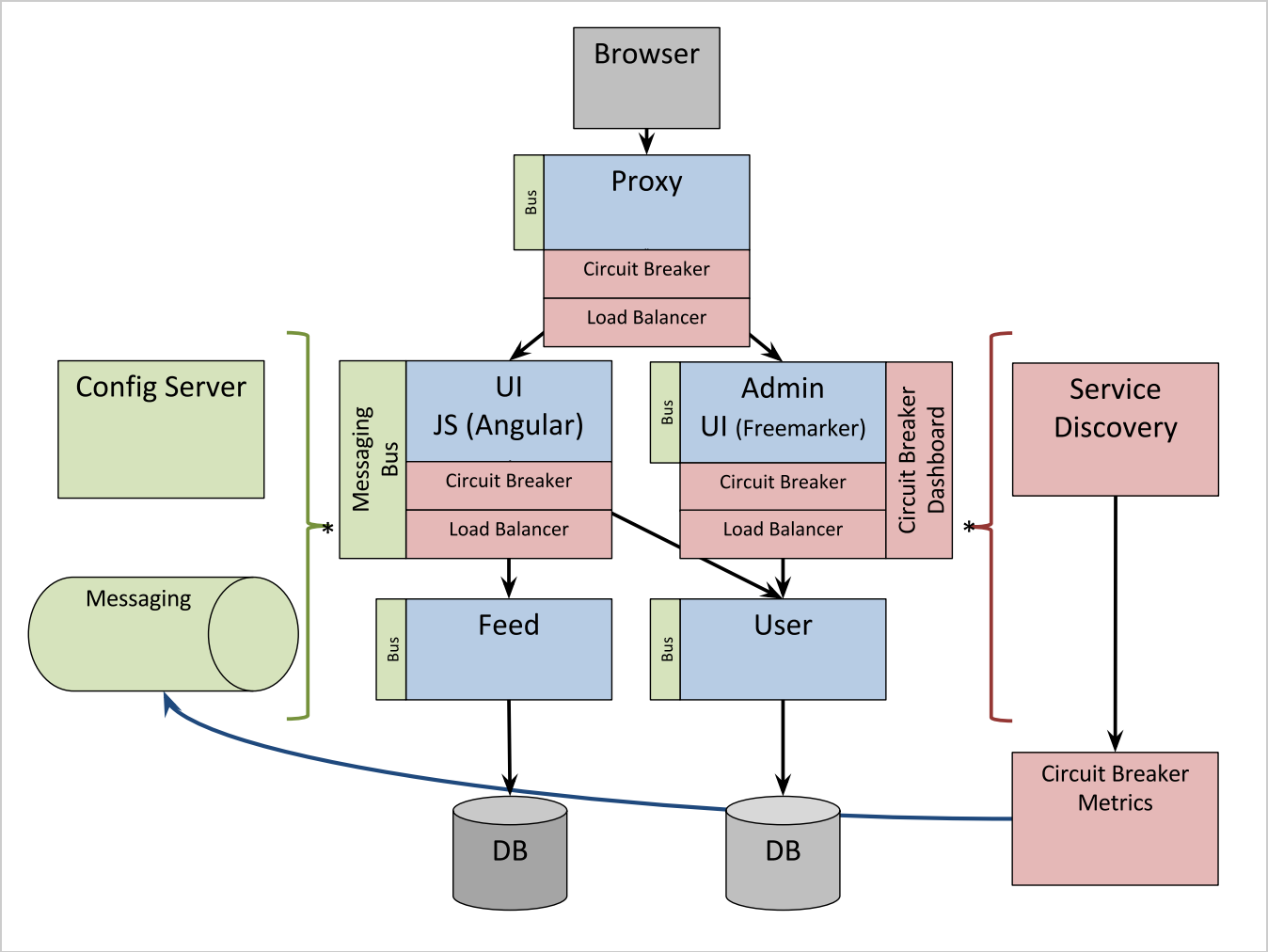
Coordination of distributed systems
leads to boiler plate patterns

- Distributed/versioned configuration
- Service registration and discovery
- Routing
- Service-to-service calls
- Load balancing
- Circuit Breaker
- Asynchronous / Reactive
- Distributed messaging

Spring IO Platform



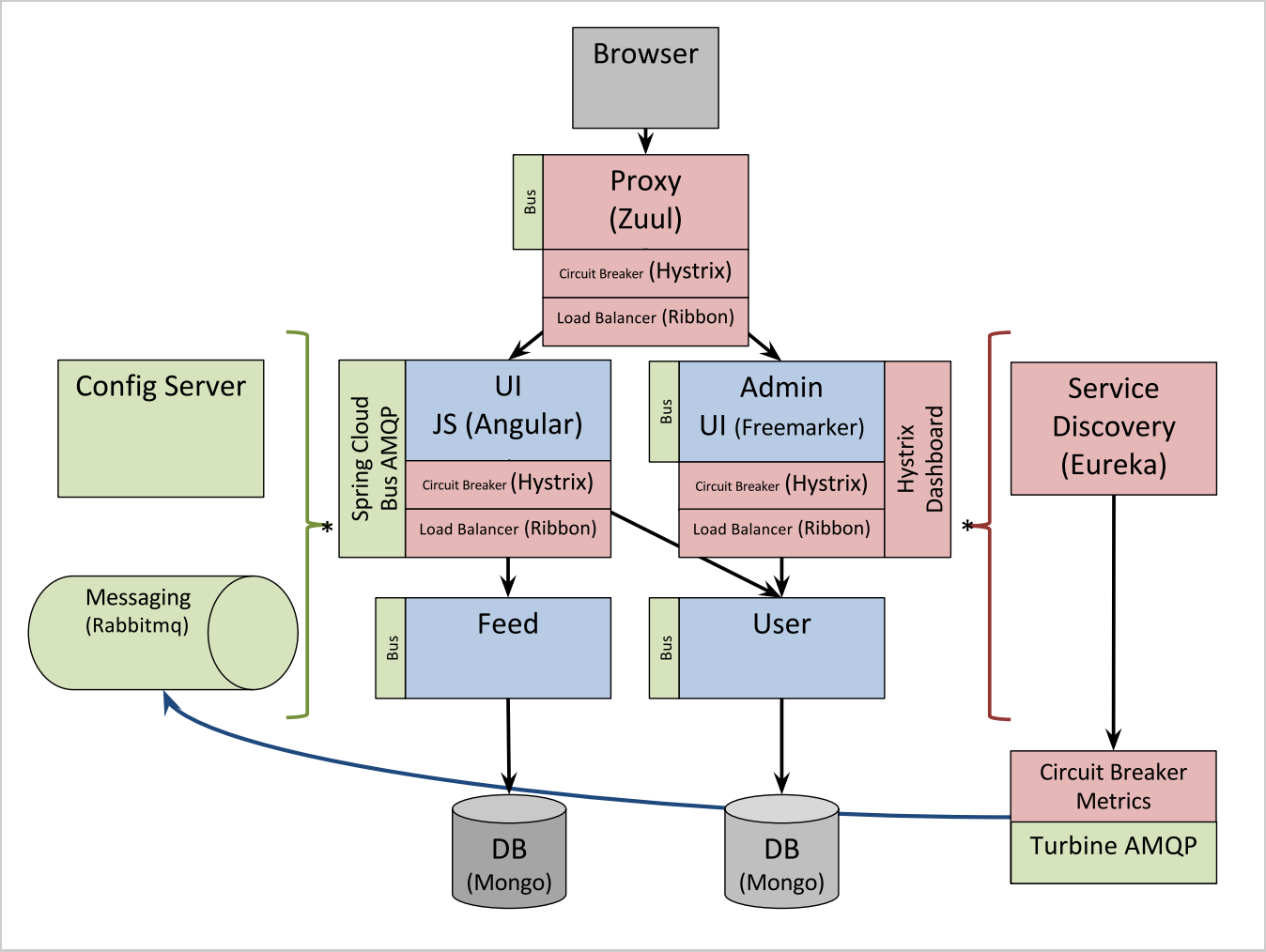
Example: Coordination Boiler Plate



Netflix OSS

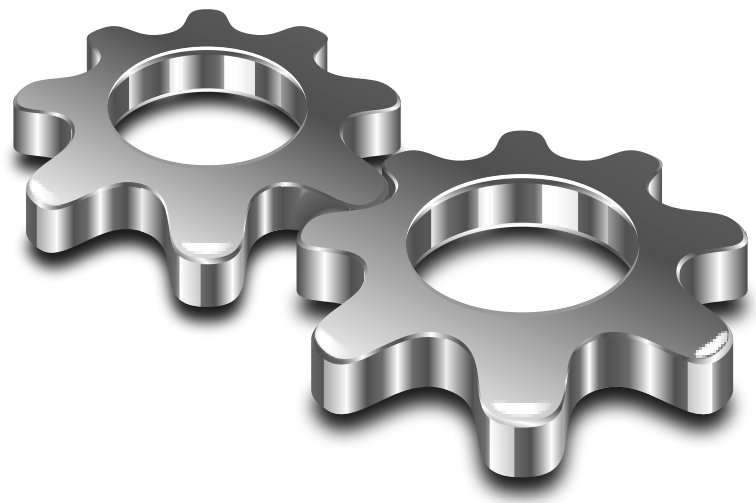
- Eureka
- Hystrix & Turbine
- Ribbon
- Feign
- Zuul
- Archaius
- Curator
- Asgaard
- ...

Example: Spring Cloud and Netflix



Configuration Server

- Pluggable source
- Git implementation
  - Per service repos
- SVN implementation
- Versioned
- Rollback-able
- Configuration client
  auto-configured via starter



Discovery: Eureka

- Service Registration Server
- Highly Available
- In AWS terms, multi Availability Zone and Region aware

http://techblog.netflix.com/2012/09/eureka.html

Client Side Load-balancing: Ribbon

- Client side load balancer
- Pluggable
- Round robin, random, weighted response time

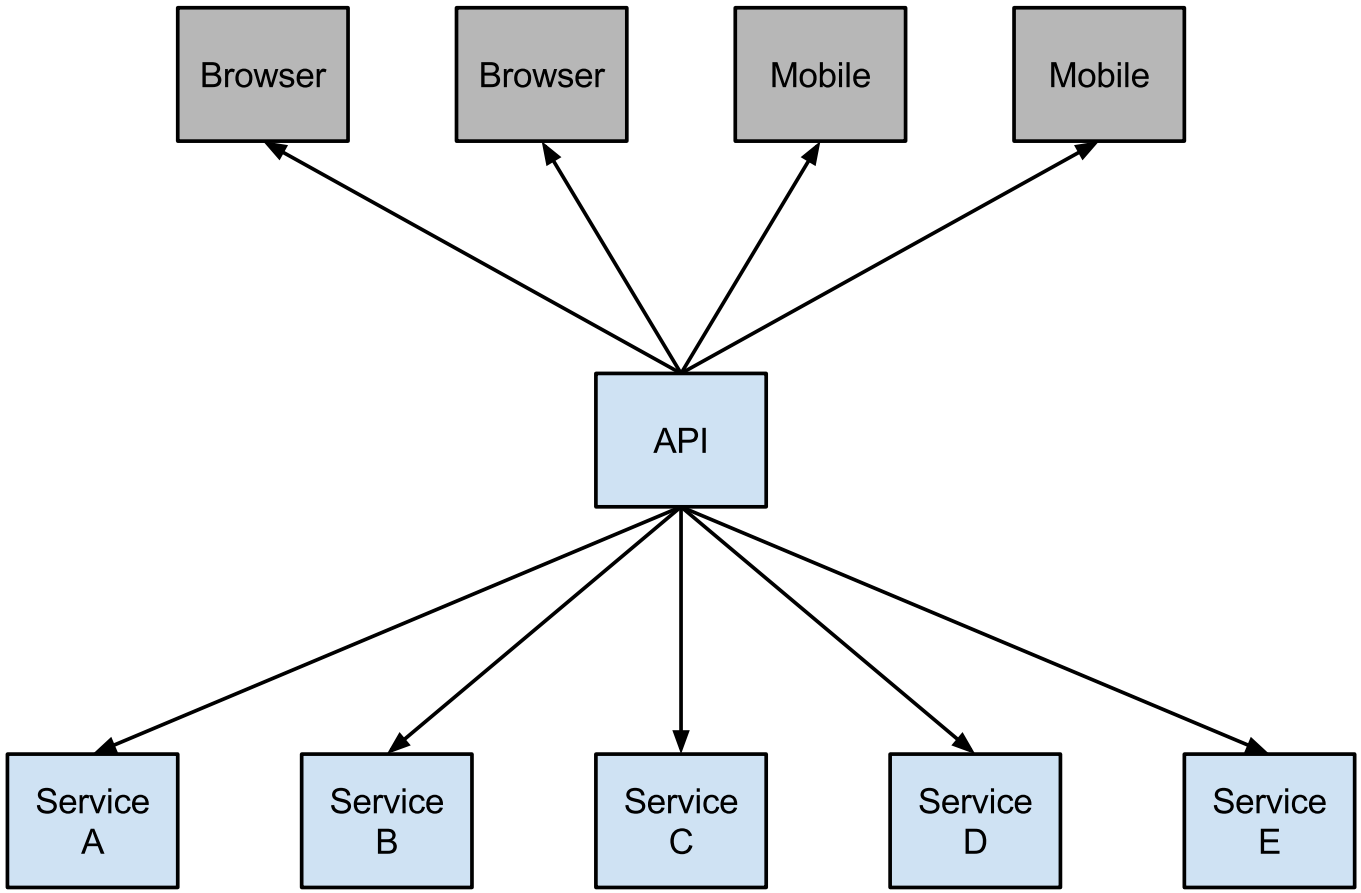
http://techblog.netflix.com/2013/01/announcing-ribbon-tying-netflix-mid.html

Circuit Breaker: Hystrix

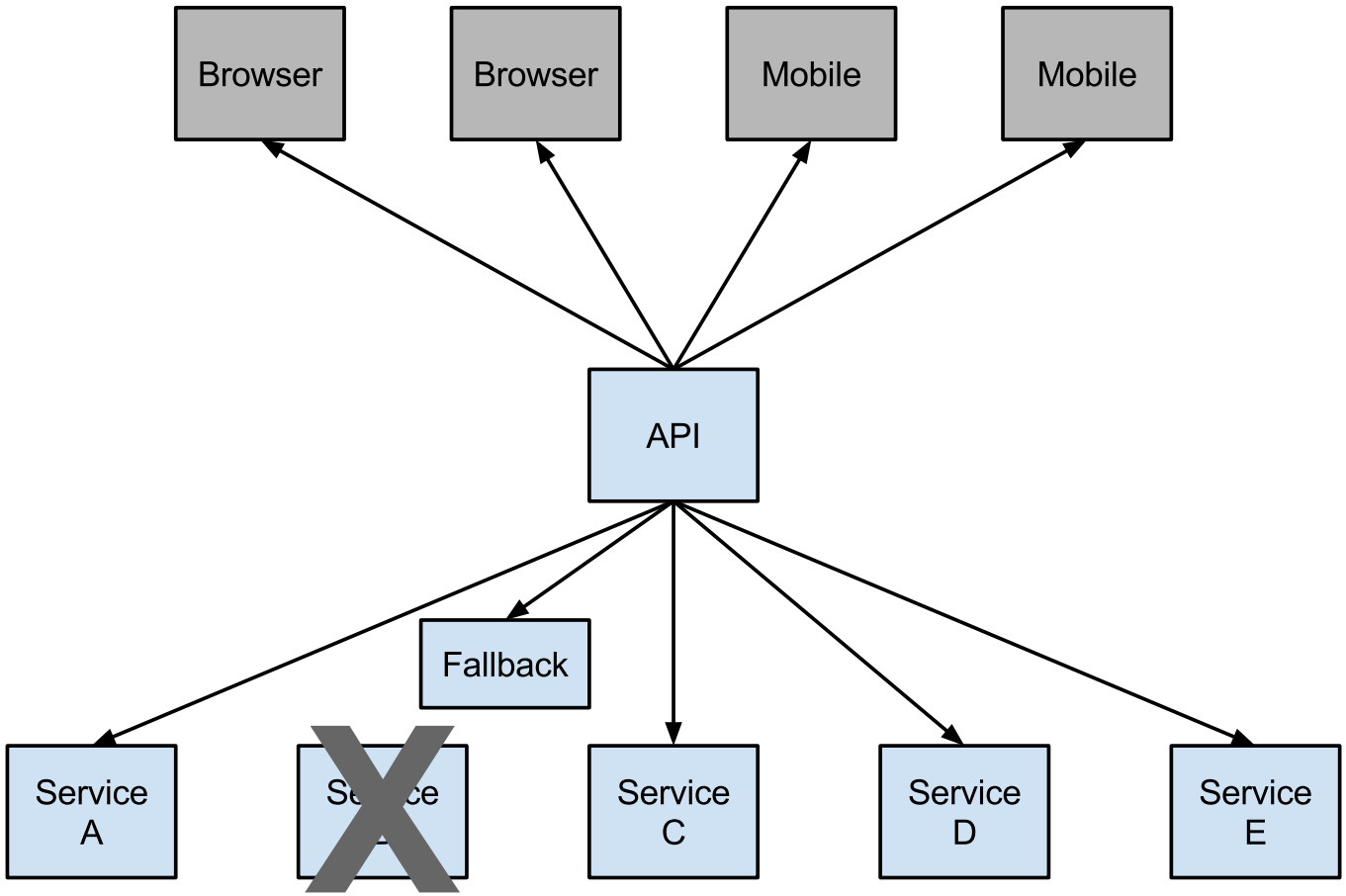
- latency and fault tolerance
- isolates access to other services
- stops cascading failures
- enables resilience
- circuit breaker pattern
- dashboard

http://techblog.netflix.com/2012/11/hystrix.html

Hystrix


Browser
Browser
Mobile
Mobile
API
Service A
Service B
Service C
Service D
Service E

Hystrix Fallback


Browser
Browser
Mobile
Mobile
API
Fallback
Service A
Service C
Service D
Service E

Circuit Breaker Metrics

- Via actuator `/metrics`
- Server side event stream `/hystrix.stream`
  - also via rabbitmq
- Dashboard app via `@EnableHystrixDashboard`

Rx Java

- Reactive: push vs. pull
- Functional
- Composable
- Return `Observable` from Spring MVC Controller (soon)
- API Gateway combining services

Rx Java Example

```
public static void hello(String... names) {
    Observable.from(names).subscribe(s -> {
            System.out.println("Hello " + s + "!");
    });
}
```

Sample functions:

- map
- flatMap
- zip
- take
- merge
- 350+ operators!

http://techblog.netflix.com/2013/02/rxjava-netflix-api.html

Routing: Zuul

- JVM based router and filter
- Similar routing role as httpd, nginx, or CF go router
- Fully programmable rules and filters
- Groovy
- Java
- any JVM language

http://techblog.netflix.com/2013/06/announcing-zuul-edge-service-in-cloud.html

How Netflix uses Zuul

- Authentication
- Insights
- Stress Testing
- Canary Testing
- Dynamic Routing
- Service Migration
- Load Shedding
- Security
- Static Response handling
- Active/Active traffic management

Spring Cloud Security

Enable Single Sign On (SSO) with an OAuth2 provider declared in external properties.

```
@EnableOAuth2Sso
```

Enable security using OAuth2 access tokens

```
@EnableOAuth2Resource
```

Spring Cloud Bus

- Distributed actuator
- `/bus/env` and `/bus/refresh` actuator endpoints
- uses Spring Messaging and Spring Integration

No soup for you!

Spring Cloud Sidecar

- For you non-java apps
- Modeled after Netflix Prana
- Built in zuul proxy

http://techblog.netflix.com/2014/11/prana-sidecar-for-your-netflix-paas.html

Spring Cloud Future

Previews, experiments or ideas (ie: **no guarantees!**)

- Distributed Locks, Leader election
- **Consul**: Config, Discovery, Bus, Locks
- **Zookeeper** or **etcd**: Locks, Leader Election, Discovery, Config
- **Zipkin** for distributed tracing
- Moar Bus! Moar Messaging!

Links

- https://github.com/spring-cloud
- https://github.com/spring-cloud-samples
- http://blog.spring.io
- http://spencer.gibb.us/preso/cloud-native-devnexus.html
- https://github.com/spencergibb/myfeed
- Twitter: @spencerbgibb, @david_syer
- Email: sgibb@pivotal.io, dsyer@pivotal.io

Notes

- https://speakerdeck.com/mstine/architecting-for-continuous-delivery-microservices-with-pivotal-cf-and-spring-cloud
- http://www.slideshare.net/ewolff/micro-services-small-is-beautiful
- http://martinfowler.com/articles/microservices.html
- http://davidmorgantini.blogspot.com/2013/08/micro-services-what-are-micro-services.html

Notes cont.

- Book (Humble and Farley): http://continuousdelivery.com
- http://techblog.netflix.com/2013/08/deploying-netflix-api.html
- Mikey Cohen Netflix edge architecture, http://goo.gl/M159zi
- https://pragprog.com/book/mnee/release-it
- https://github.com/ReactiveX/RxJava
- http://reactivex.io

Spring Restdocs

- Programatically generated snippets from unit tests!
- Hand write **asciidoc** docs including generated snippets

```
[source,http]
----
HTTP/1.1 200 OK
Content-Type: application/hal+json

{
  "_links" : {
    "users" : {
      "href" : "http://localhost:11070/users{?page,size,sort}",
      "templated" : true
    },
    "profile" : {
      "href" : "http://localhost:11070/alps"
    }
  }
}
```

https://github.com/spring-projects/spring-restdocs

Spring Session

- Generic vendor session abstraction
- Can be used in any environment, not just web

```
@EnableRedisHttpSession
```

http://projects.spring.io/spring-session

Continuous Delivery

- Microservices lend themselves to continuous delivery.
- You actually *need* continuous delivery to extract maximum value.
- **New:** ALM support in Cloudfoundry from Cloudbees

Cloudfoundry

- Environment Provisioning
- On-Demand/Automatic Scaling
- Failover/Resilience
- Routing/Front-end Load Balancing
- Monitoring

Deploying services needs to be simple and reproducible

```
$ cf push app.groovy
```

and you don't get much more convenient than that.

(Same argument for other PaaS solutions)

Micro vs Monolithic... is NOT new

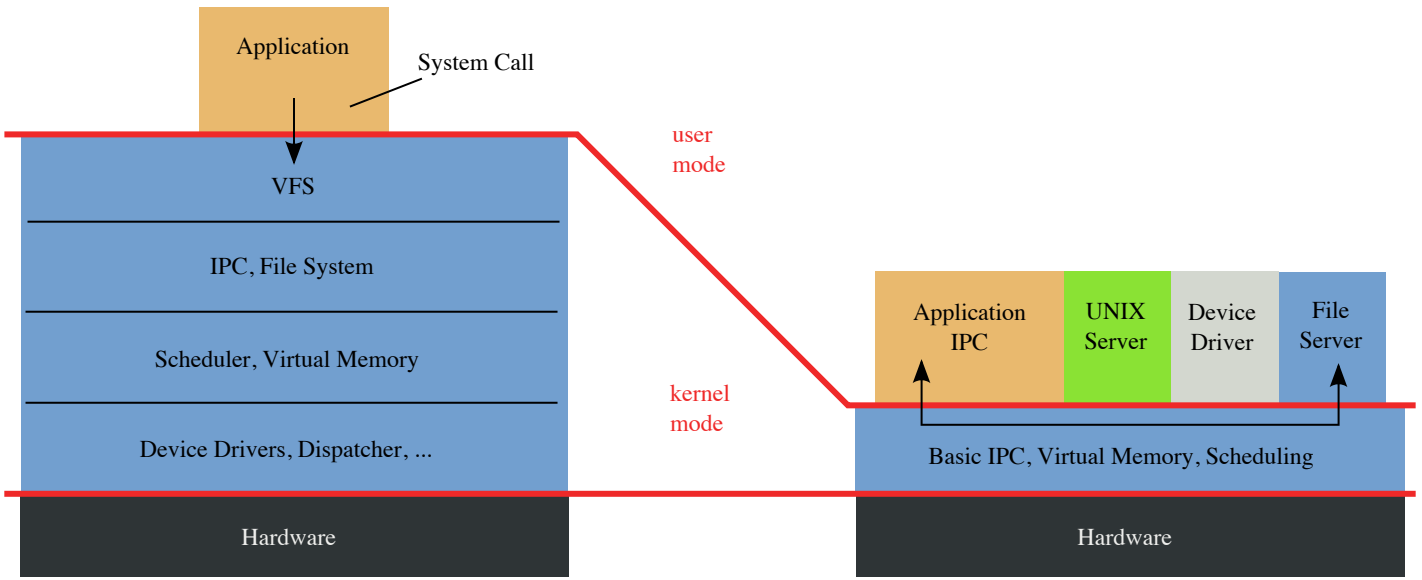
```
From:         kt4@prism.gatech.EDU (Ken Thompson)
Subject:      Re: LINUX is obsolete
Date:         3 Feb 92 23:07:54 GMT
Organization: Georgia Institute of Technology

I would generally agree that microkernels are probably the wave
of the future. However, it is in my opinion easier to implement
a monolithic kernel. It is also easier for it to turn into a
mess in a hurry as it is modified.

Regards, Ken
```

Monolithic Kernel
based Operating System

Microkernel
based Operating System



What's wrong with a monolith?