SDEC15
SOFTWARE DEVELOPMENT & EVOLUTION CONFERENCE 2015

Think Small
An Introduction to Microservices

Thank you to our Sponsors

Protegra

Red River College of Applied Arts, Science and Technology

InfoQ

MEDIA SPONSOR
Winnipeg Free Press

iQmetrix

Manitoba Institute of Trades and Technology

Northfield IT

Think Small

An Introduction to Microservices

stgodard@ca.ibm.com

IBM

@markstgodard

@markstgodard

- Software Engineer @ IBM

IBM Bluemix

CLOUD FOUNDRY

- Java, Ruby, Go
- Agile / Open Source / Cloud Foundry

- *What are Microservices?*
- *Monolith vs. Microservices*
- *Considerations when implementing Microservices*

Continuous Delivery

What are Microservices?


“Loosely coupled service oriented architecture with bounded contexts”
- Adrian Cockcroft


“Loosely coupled service oriented architecture with bounded contexts”
- Adrian Cockcroft


Small &
Focused
* do one thing and do it well

Independently Deployable

Independently Scalable

Independent Teams

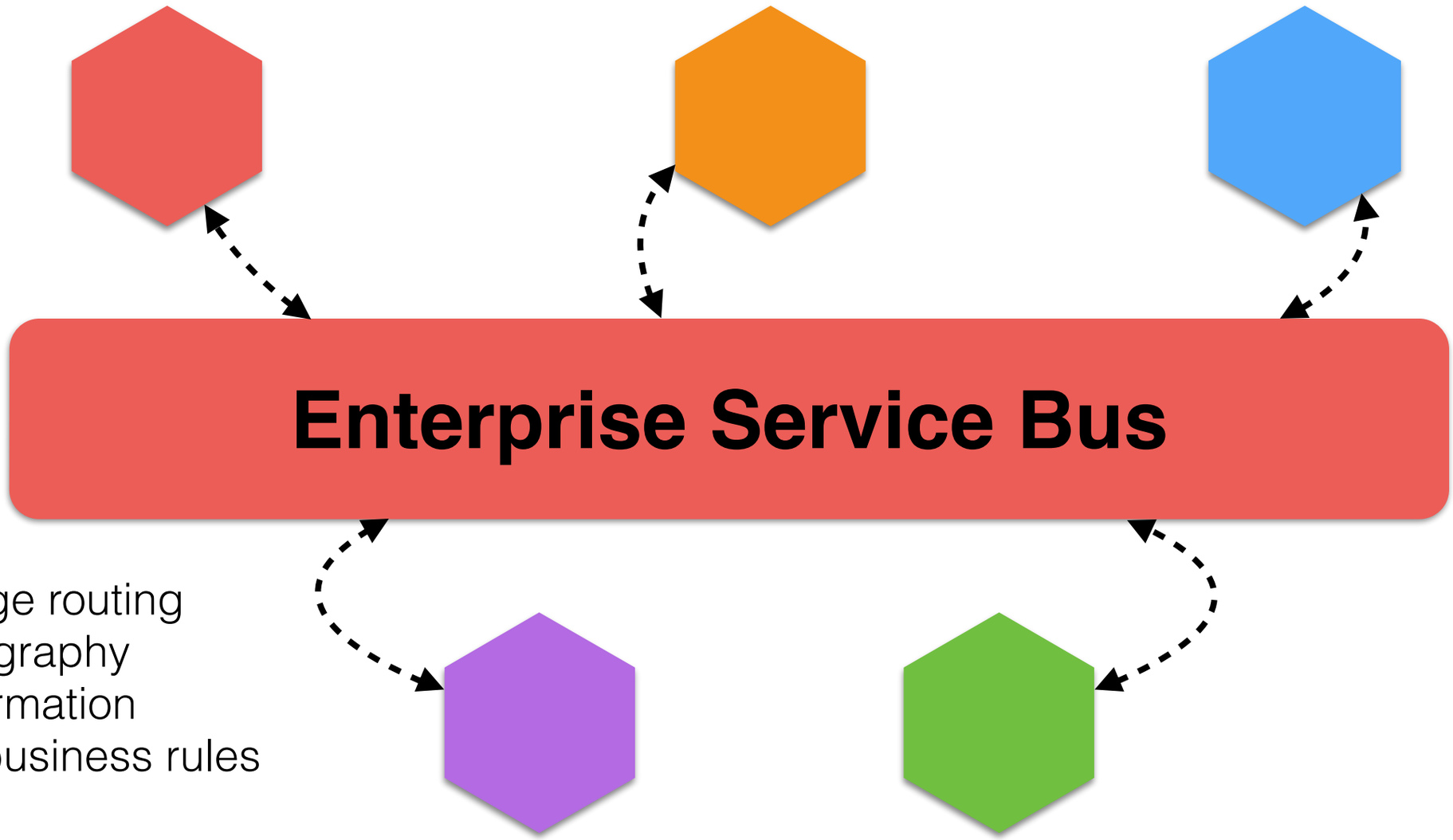
Language Neutral

** expose app functionality using APIs*

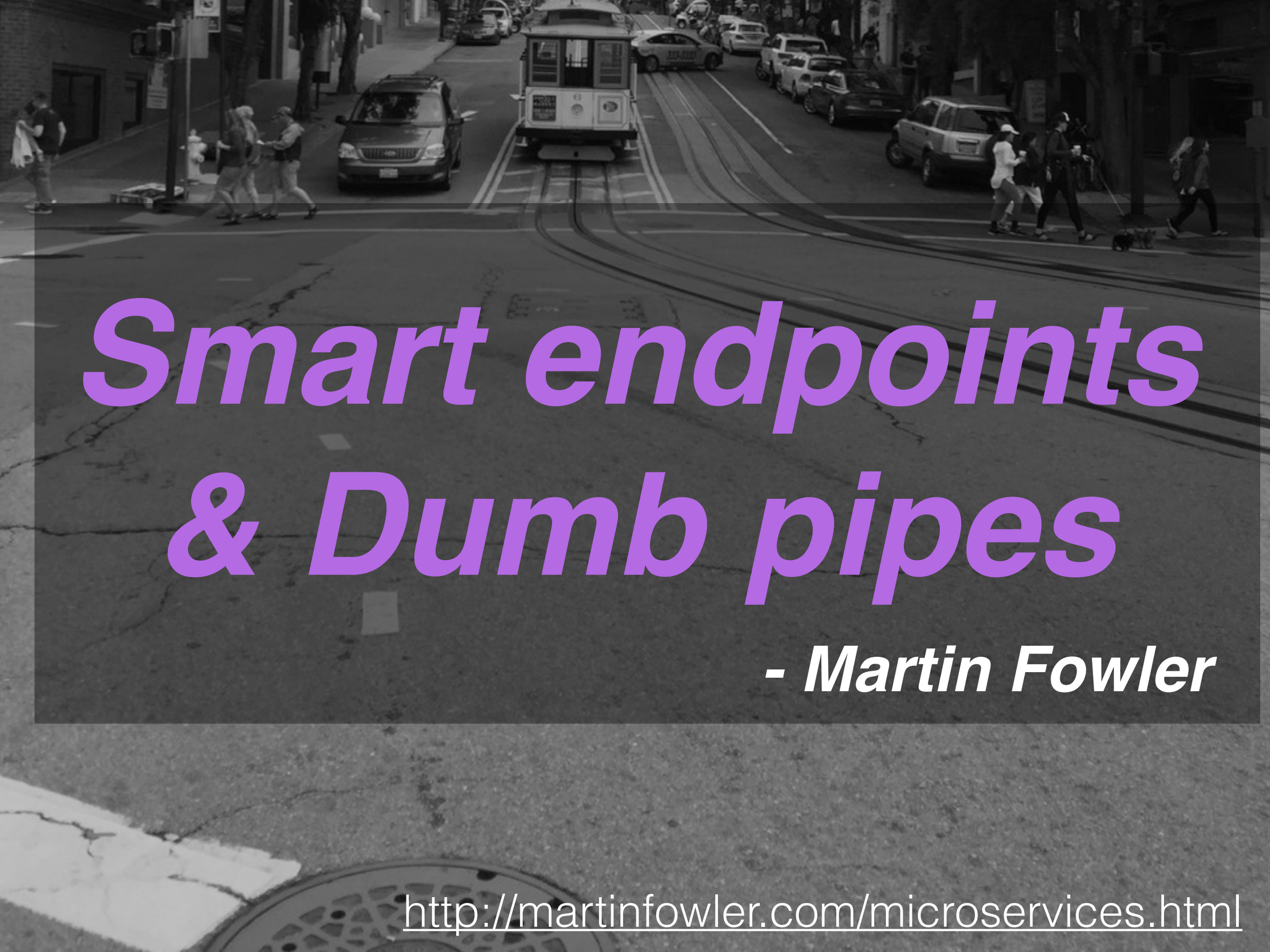
Replaceable

“assume you will build it again” - Jim Benson (this morning)


“Loosely coupled service oriented architecture with bounded contexts”
- Adrian Cockcroft



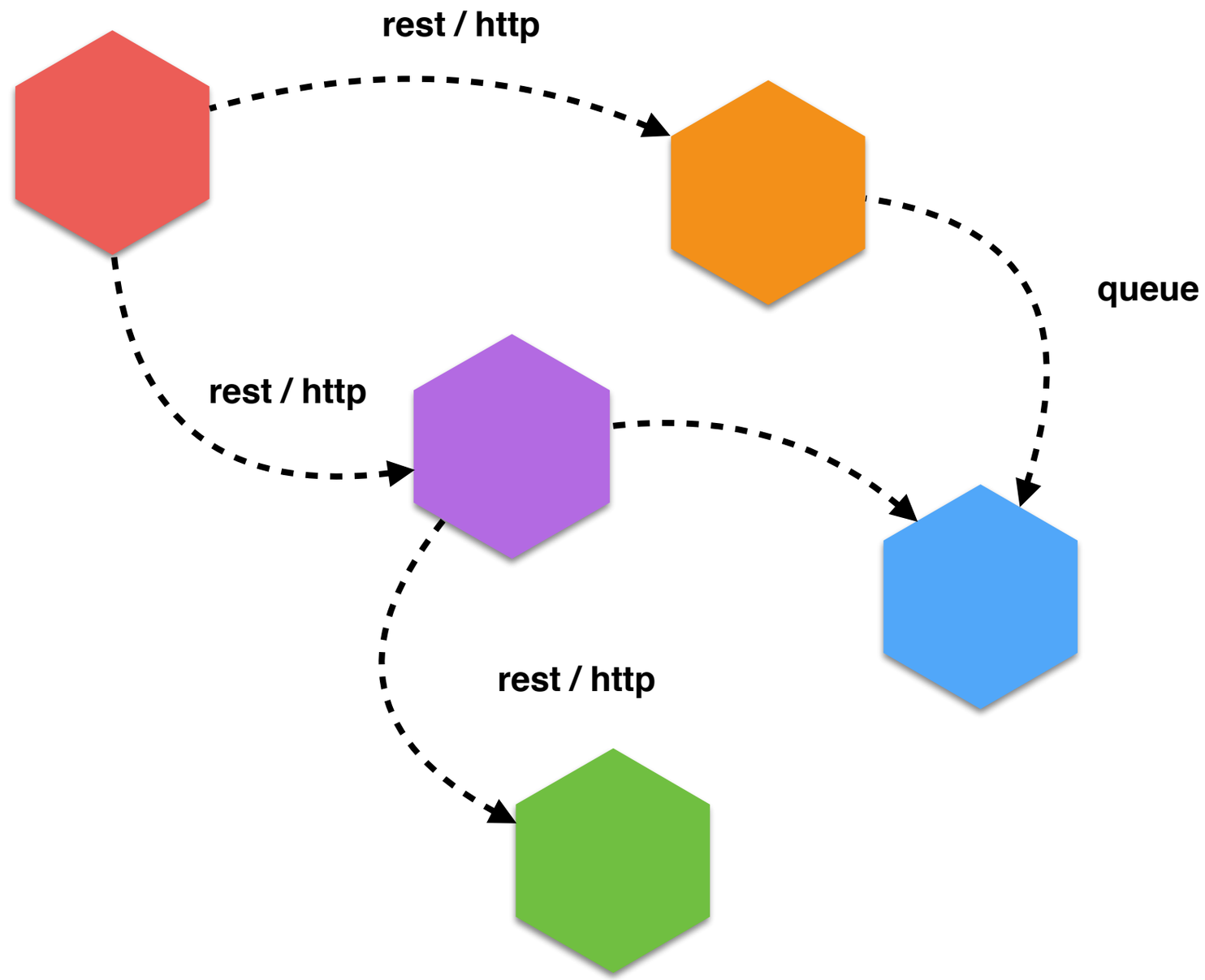
- message routing
- choreography
- transformation
- apply business rules



Smart endpoints & Dumb pipes

- Martin Fowler

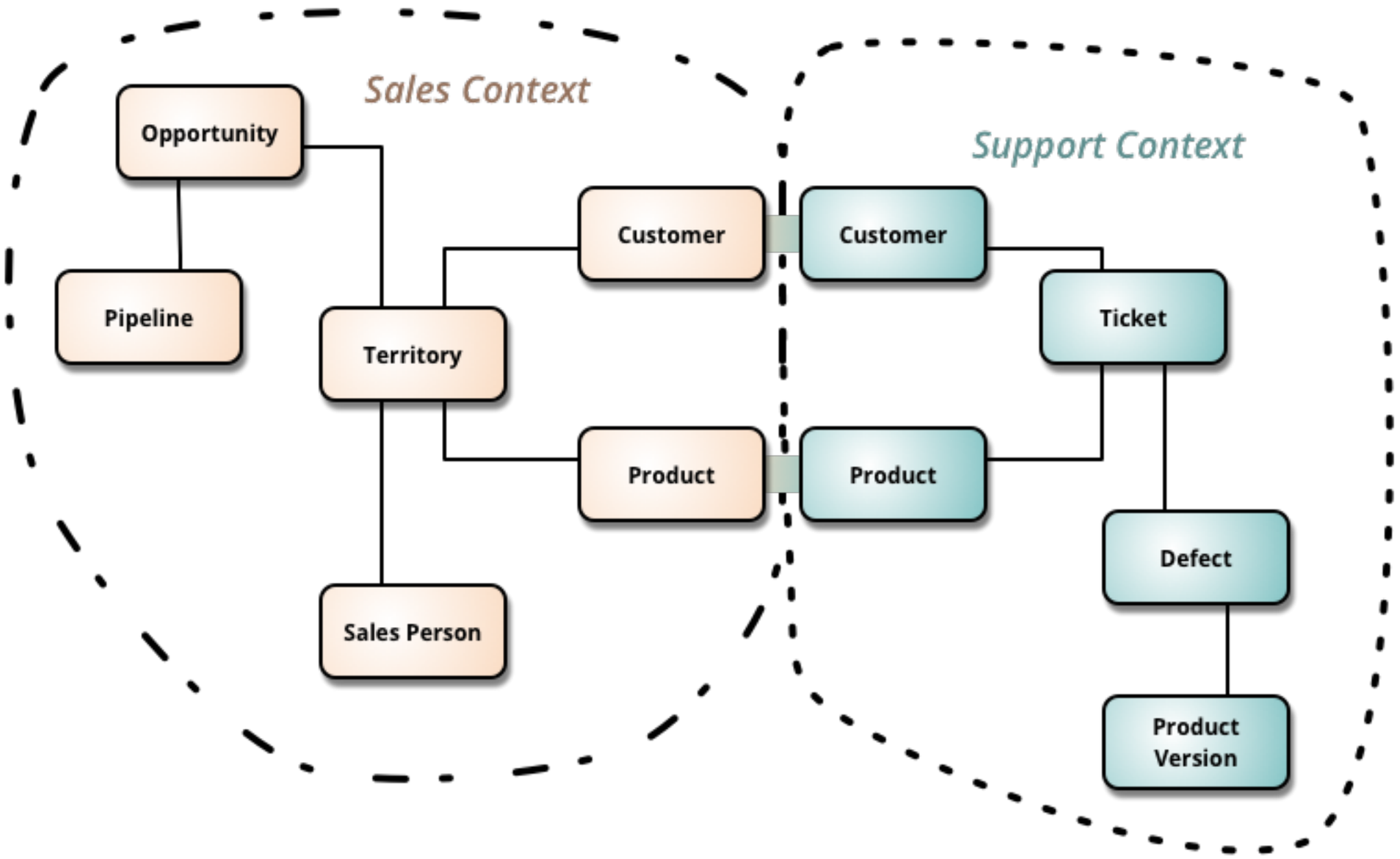
http://martinfowler.com/microservices.html


rest / http
queue
rest / http
rest / http


“Loosely coupled service oriented architecture with bounded contexts”
- Adrian Cockcroft

Business Capability

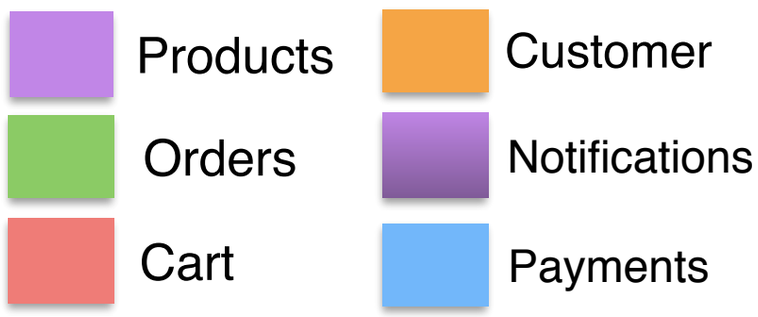
Example Bounded Context



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

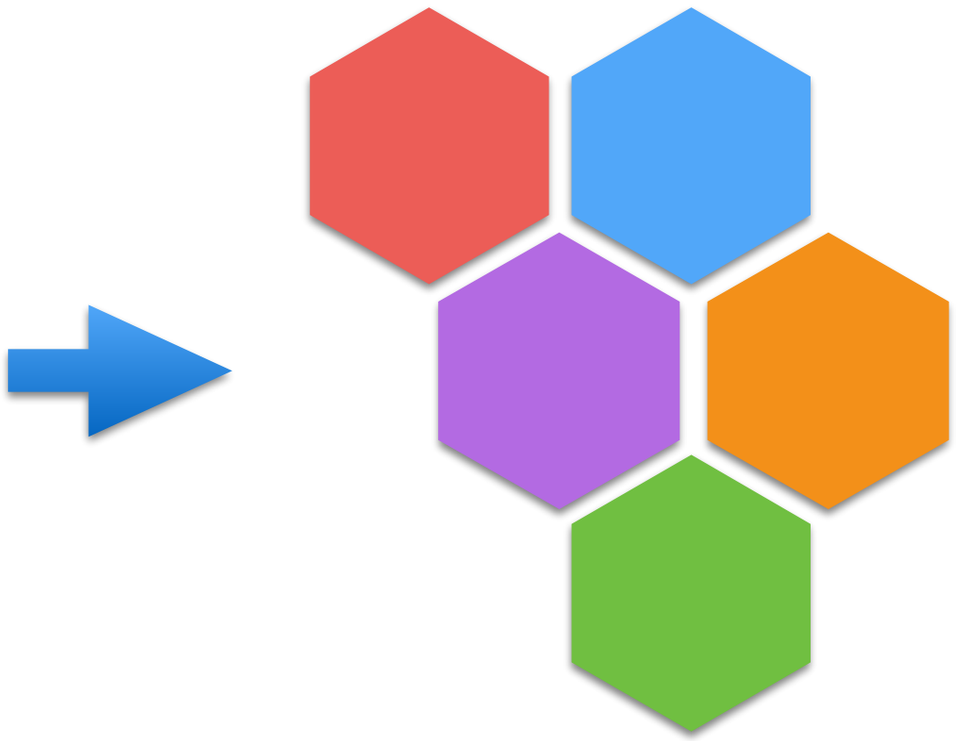
Monolith vs. Microservice

Monolithic



Monolithic

Microservices



Products
Orders
Cart
Customer
Notifications
Payments

Monolith

Products
Orders
Cart
Customer
Notifications
Payments

HTML / HTTP

UI (HTML / Javascript / CSS)

Web Controllers / API

HTTP session

Services / Domain Model

Postgres (DB)

AMQP
(message
queue)

Products
Orders
Cart
Customer
Notifications
Payments

HTML / HTTP

UI (HTML / Javascript / CSS)

Web Controllers / API

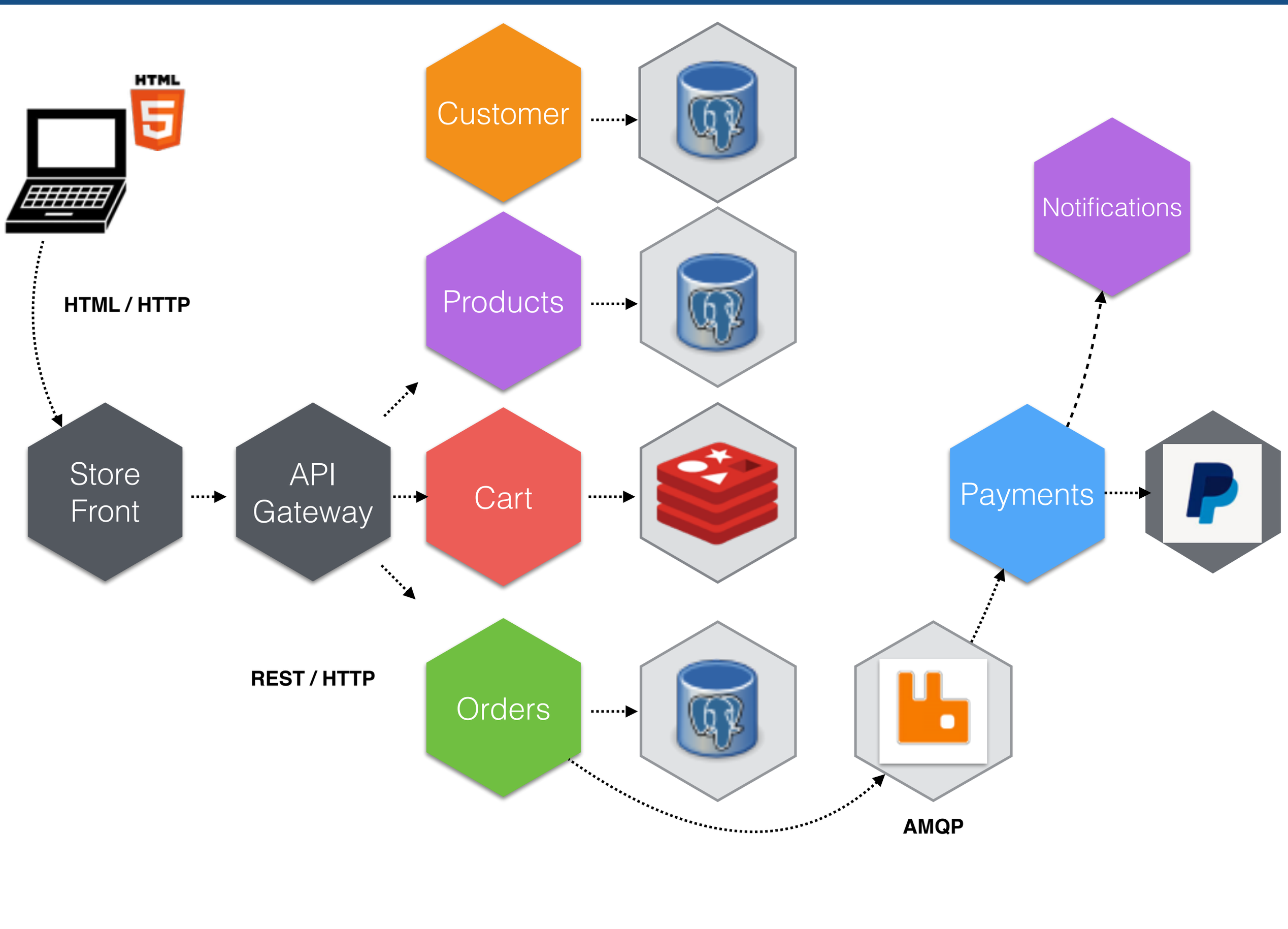
HTTP session

Services / Domain Model

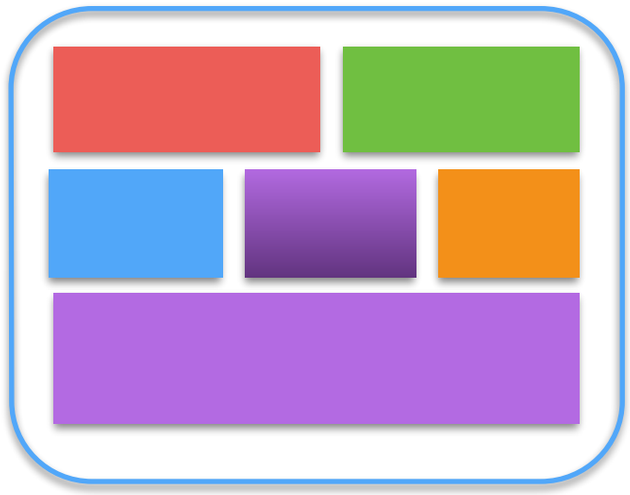
Postgres (DB)

AMQP
(message
queue)

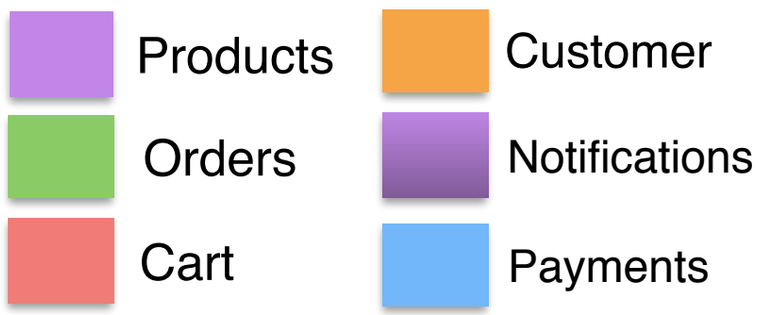
Microservices


HTML
HTML / HTTP
Customer
Products
Notifications
Store Front
API Gateway
Cart
Payments
REST / HTTP
Orders
AMQP

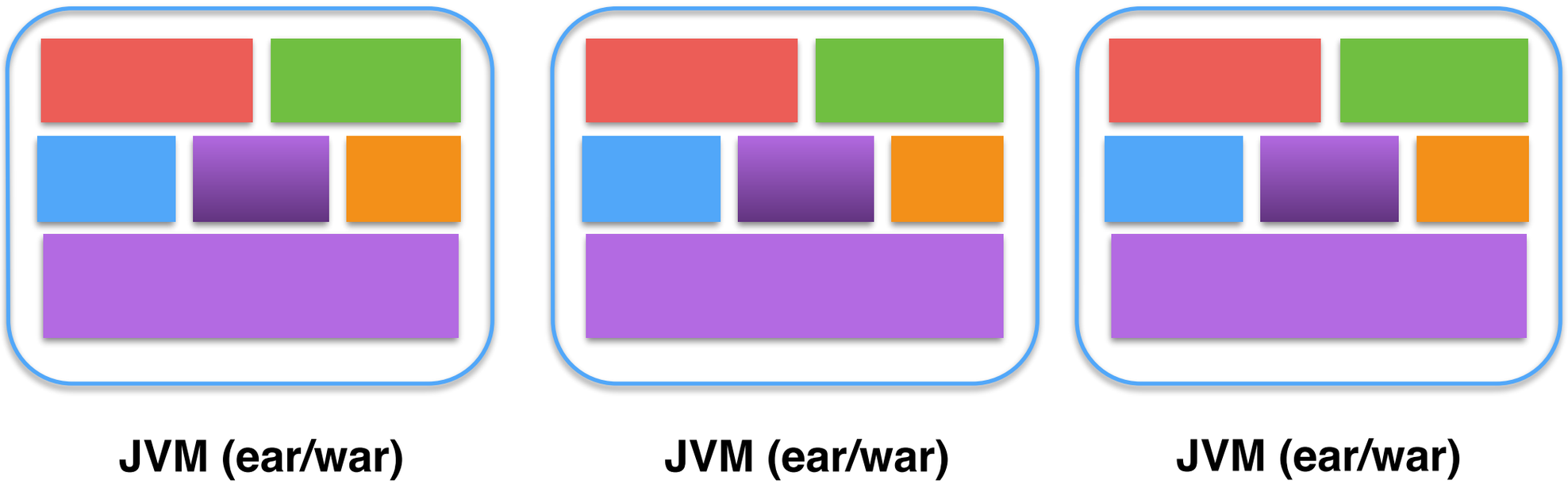
Scaling (Monolith)



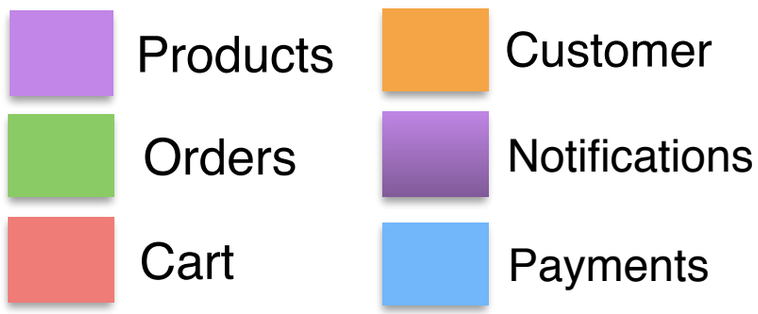
JVM (ear/war)



Scaling (Monolith)



JVM (ear/war) **JVM (ear/war)** **JVM (ear/war)**



Scaling (Micro services)

Products
Customer
Orders
Notifications
Cart
Payments

HTML

HTML / HTTP

Customer

x2

Products

Notifications

Store Front

API Gateway

Cart

x2

Payments

REST / HTTP

Orders

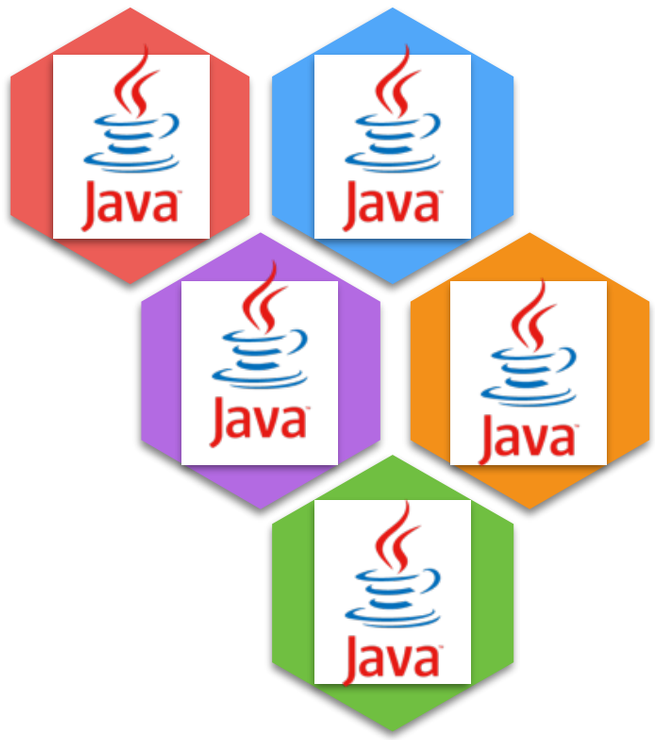
x3

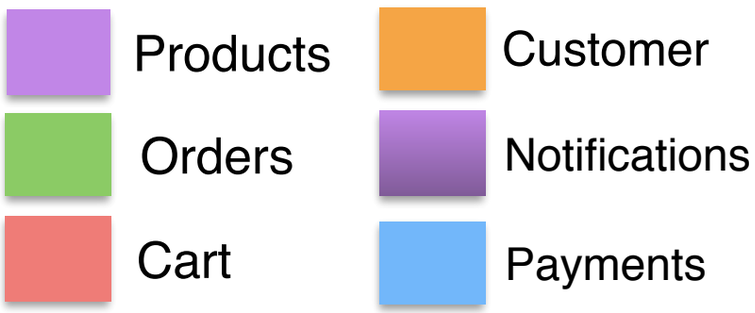
AMQP

Languages & Technology

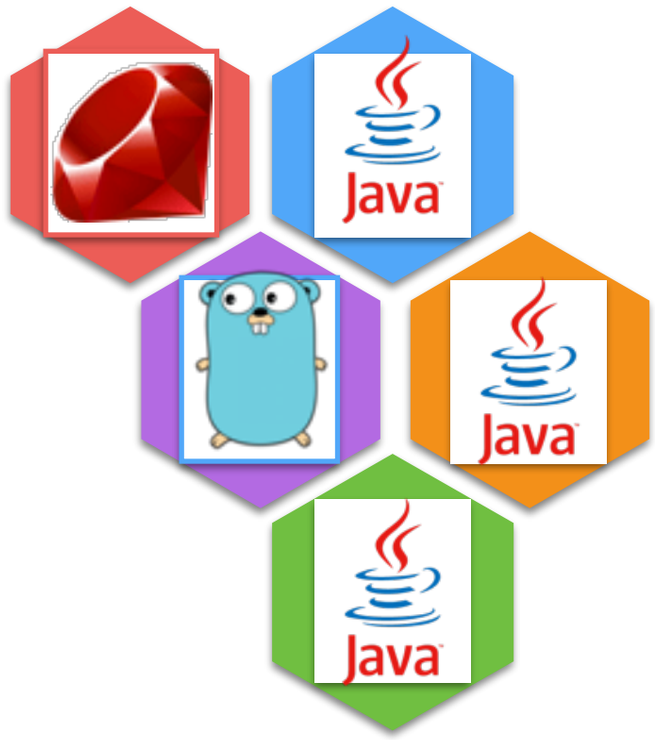
Products
Orders
Cart
Customer
Notifications
Payments

Languages & Technology

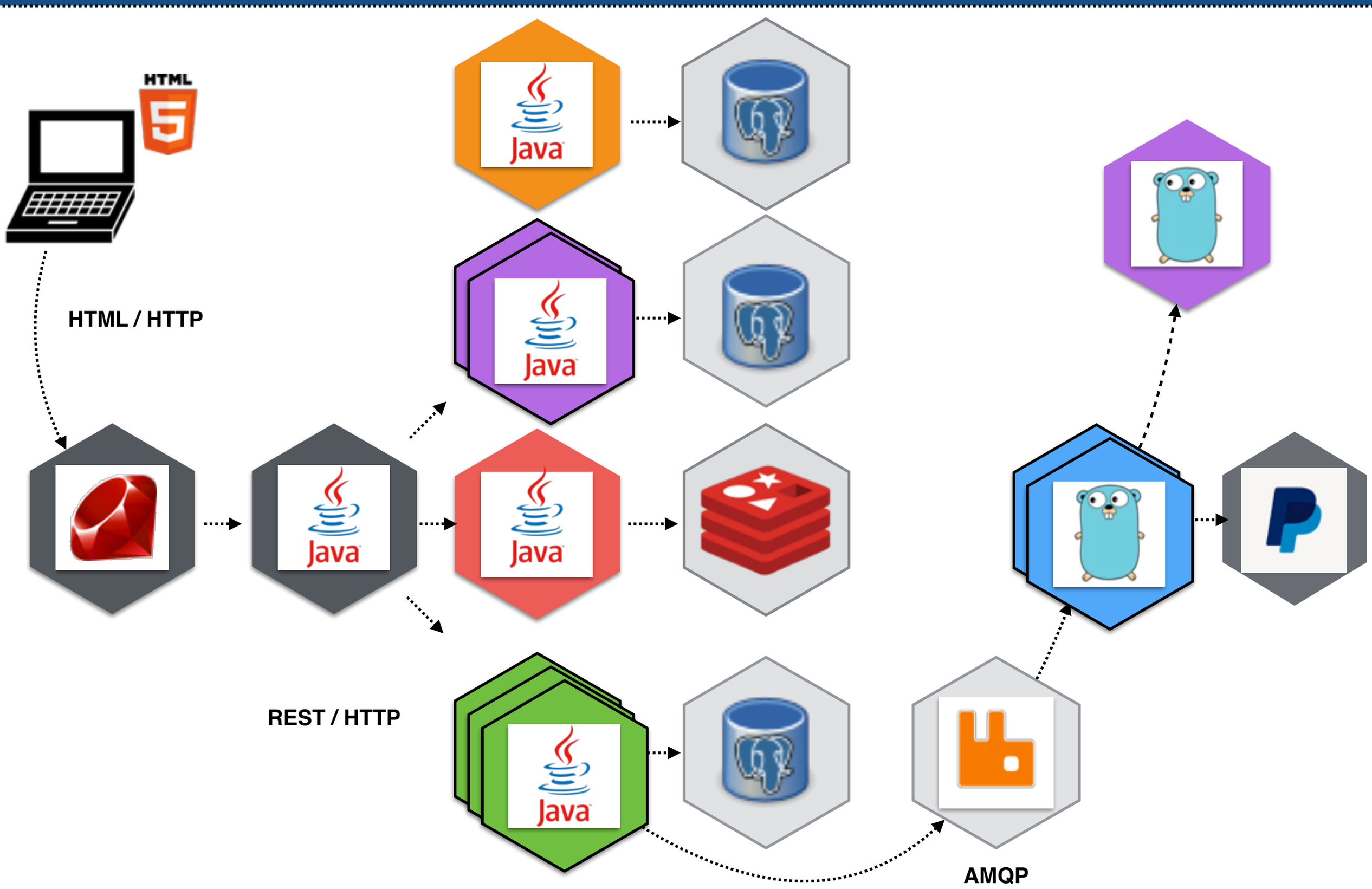




Languages & Technology



Products
Orders
Cart
Customer
Notifications
Payments


HTML
HTML / HTTP
Java
Java
Java
Java
Java
REST / HTTP
AMQP

Key principles when implementing Micro Services

Example App

using

Micro Services

STUFF!

(0) Items Checkout

Breaking Bad S2 $19.99

Walter White, a struggling high school chemistry teacher, is diagnosed with advanced lung cancer.

★★★★☆ 10 reviews

Add to Cart

Mr. Robot S1 $21.99

The show follows Elliot, who is a cyber-security tech by day and vigilante hacker by night.

★★★★☆ 5 reviews

Add to Cart

Fight Club $16.99

A ticking-time-bomb insomniac and a slippery soap salesman channel primal male aggression into a shocking new form of

★★★★★ 2 reviews

Add to Cart

STUFF!

(0) Items Checkout

Breaking Bad S2 $19.99

Walter White, a struggling high school chemistry teacher, is diagnosed with advanced lung cancer.

★★★★☆ 10 reviews

Add to Cart

Mr. Robot S1 $21.99

The show follows Elliot, who is a cyber-security tech by day and vigilante hacker by night.

★★★★☆ 5 reviews

Add to Cart

Fight Club $16.99

A ticking-time-bomb insomniac and a slippery soap salesman channel primal male aggression into a shocking new form of

★★★★★ 2 reviews

Add to Cart

Products

STUFF!

(0) Items Checkout

Breaking Bad S2 $19.99

Walter White, a struggling high school chemistry teacher, is diagnosed with advanced lung cancer.

★★★★☆ 10 reviews

Add to Cart

Mr. Robot S1 $21.99

The show follows Elliot, who is a cyber-security tech by day and vigilante hacker by night.

★★★★☆ 5 reviews

Add to Cart

Fight Club $16.99

A ticking-time-bomb insomniac and a slippery soap salesman channel primal male aggression into a shocking new form of

★★★★★ 2 reviews

Add to Cart

Cart

STUFF!

(0) Items

Checkout

Breaking Bad S2 $19.99

Walter White, a struggling high school chemistry teacher, is diagnosed with advanced lung cancer.

★★★★☆ 10 reviews

Add to Cart

Mr. Robot S1 $21.99

The show follows Elliot, who is a cyber-security tech by day and vigilante hacker by night.

★★★★☆ 5 reviews

Add to Cart

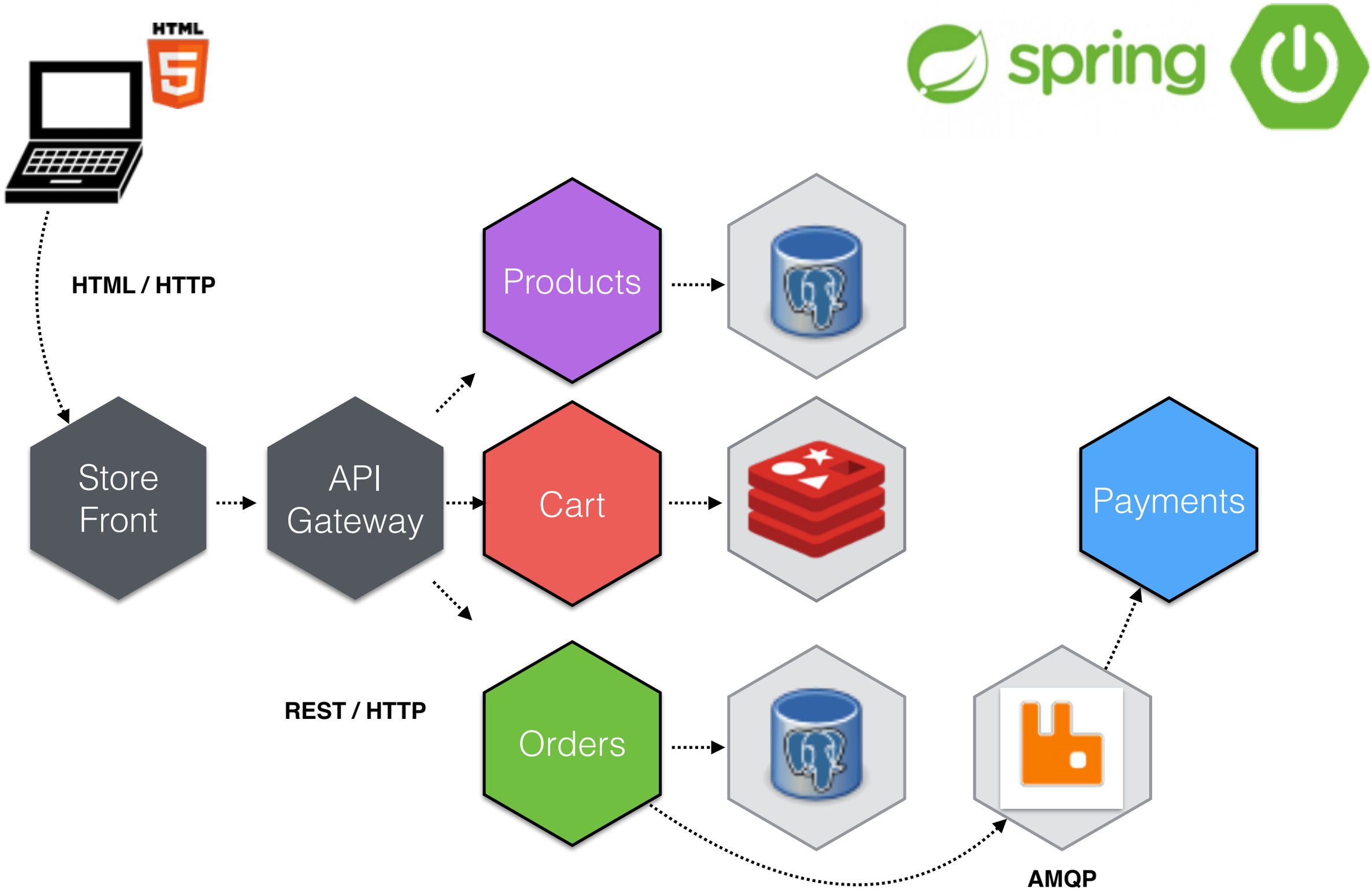
Fight Club $16.99

A ticking-time-bomb insomniac and a slippery soap salesman channel primal male aggression into a shocking new form of

★★★★★ 2 reviews

Add to Cart

Orders


HTML
HTML / HTTP
Products
Store Front
API Gateway
Cart
Payments
REST / HTTP
Orders
AMQP

java -jar <app>.jar

One Codebase

per

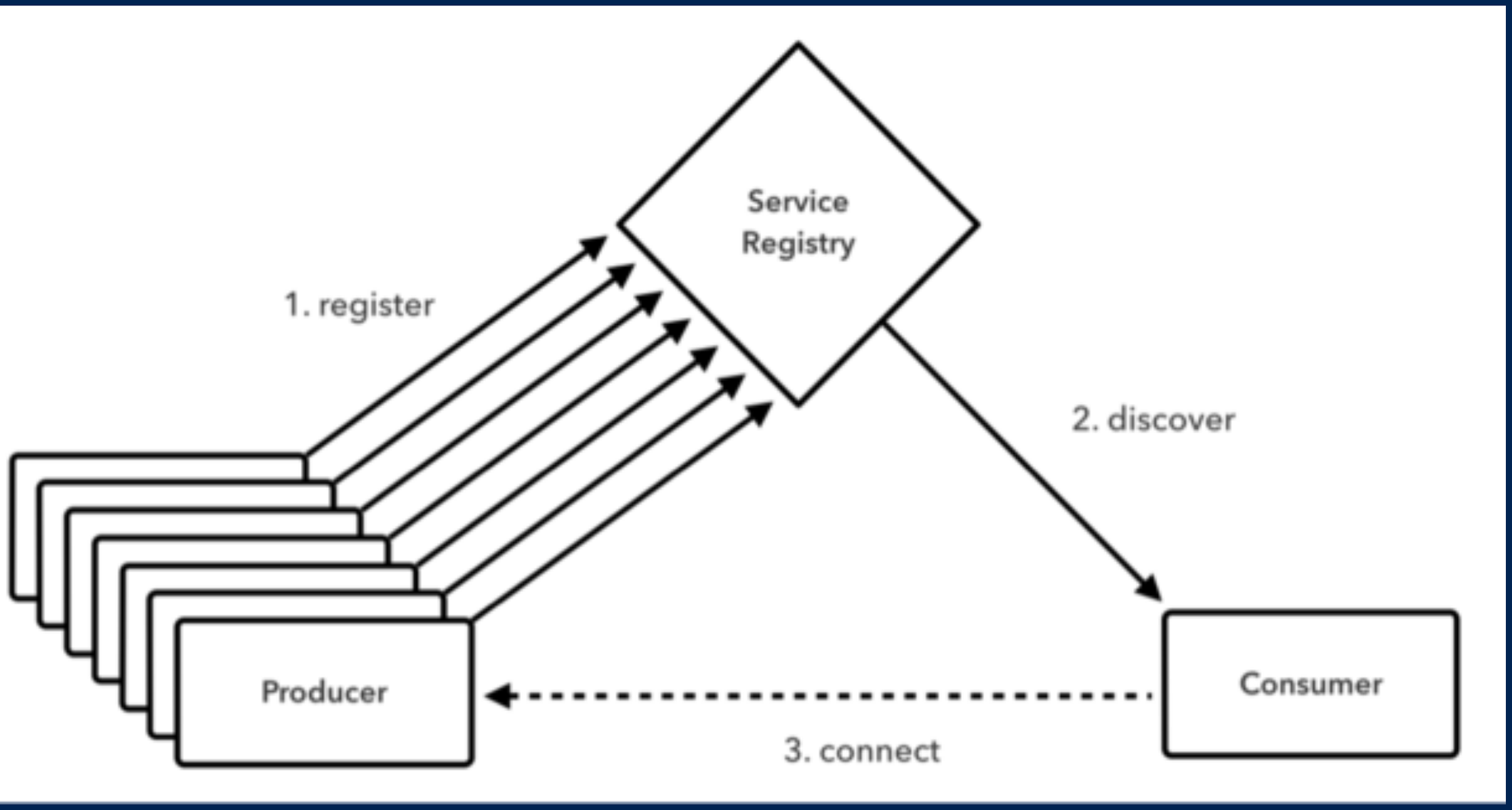
Micro Service

Service Registry

&

Discovery

Service Registry & Discovery



http://docs.pivotal.io/spring-cloud-services/service-registry/index.html

Service Registry (Eureka)

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
STUFF-CART	n/a (1)	(1)	UP (1) - 9.80.81.186
STUFF-ORDERS	n/a (1)	(1)	UP (1) - 9.80.81.186
STUFF-PRODUCTS	n/a (1)	(1)	UP (1) - 9.80.81.186
STUFF-STORE-GATEWAY	n/a (1)	(1)	UP (1) - 9.80.81.186

Service Registry

```
public ResponseEntity getProducts() {
    ParameterizedTypeReference<Resource> responseType =

    // use the "smart" Eureka-aware RestTemplate
    ResponseEntity exchange =
        this.restTemplate.exchange(
                "http://stuff-products/products",
                HttpMethod.GET,
                null,
                responseType);
```

Service Registry (DNS)

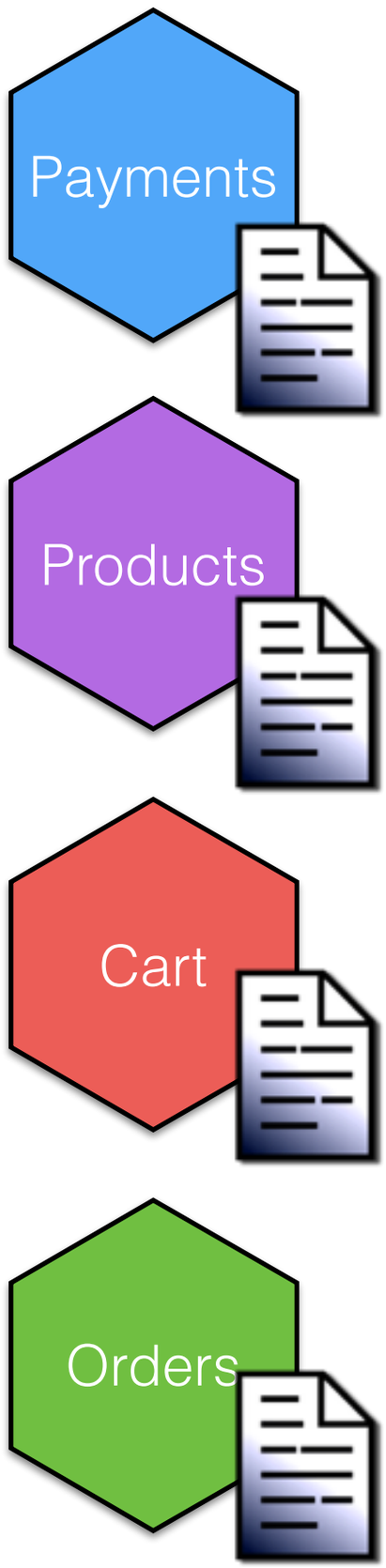
```
~(master ✓) $ cf apps
Getting apps in org stgodard@ca.ibm.com / space demo as stgodard@ca.ibm.com...
OK

name                 requested state   instances   memory   disk   urls
cart                 started           1/1         256M     1G     stuff-cart.mybluemix.net
orders               started           1/1         256M     1G     stuff-orders.mybluemix.net
products             started           1/1         256M     1G     stuff-products.mybluemix.net
service-registry     started           1/1         256M     1G     stuff-eureka.mybluemix.net
store-api-gateway    started           1/1         256M     1G     stuff-api-gateway.mybluemix.net
storefront           started           1/1         256M     1G     stuff-store.mybluemix.net
~(master ✓) $
```

Service Registry (DNS)

App Name	Runtime	Health
cart http://stuff-cart.mybluemix.net	Liberty for Java™	Running
orders http://stuff-orders.mybluemix.net	Liberty for Java™	Running
products http://stuff-products.mybluemix.net	Liberty for Java™	Running
service-registry http://stuff-eureka.mybluemix.net	Liberty for Java™	Running
store-api-gateway http://stuff-api-gateway.mybluemix.n	Liberty for Java™	Running
storefront http://stuff-store.mybluemix.net	SDK for Node.js™	Running

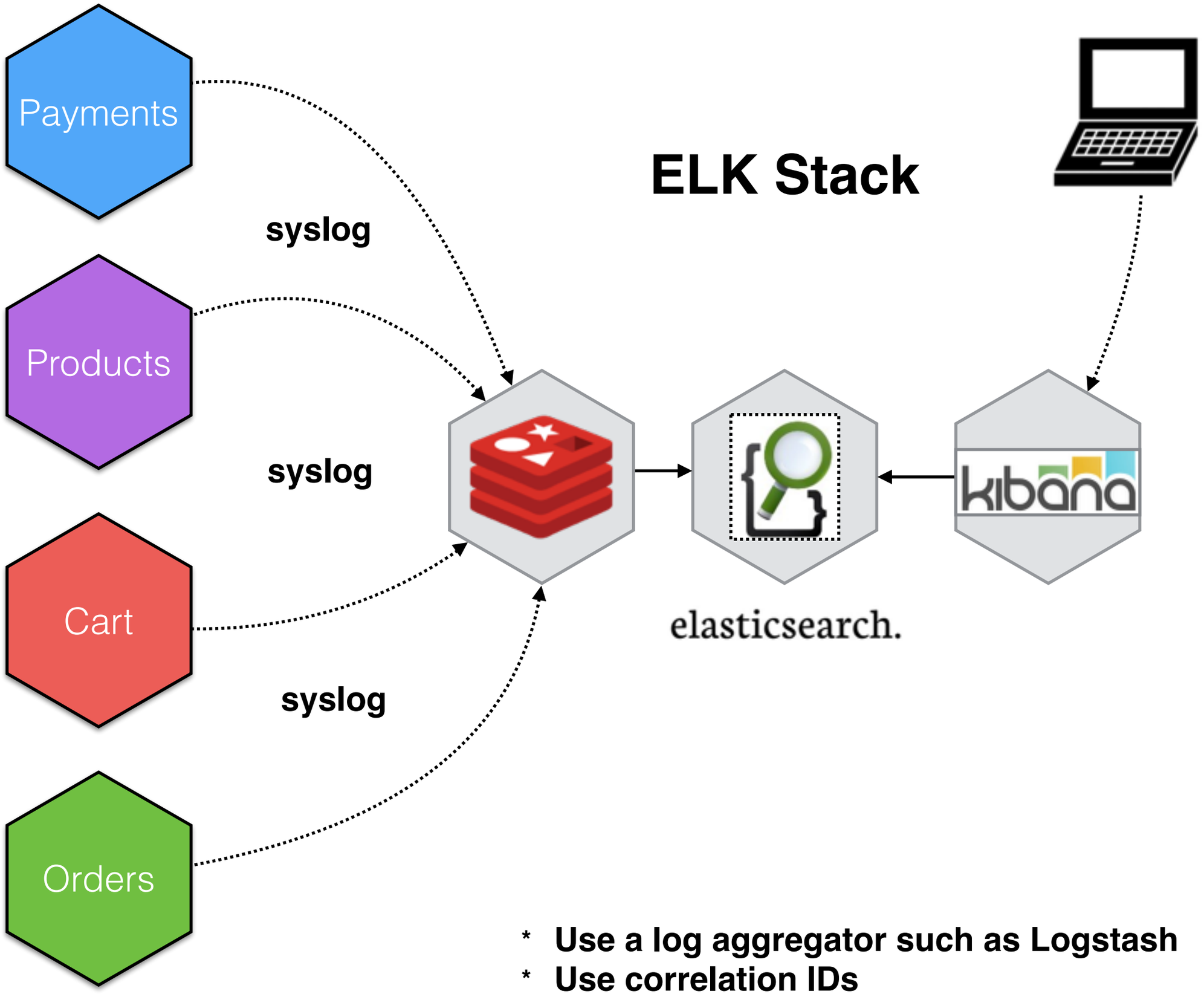
Treat logs as Event Streams

** use centralized logging / aggregation*



Logs

- multiple microservices
- potentially multiple instances of services
- containers in cloud platforms are ephemeral
- Don’t log to disk in a cloud platform



* **Use a log aggregator such as Logstash**
* **Use correlation IDs**

Design for Failure

** use Circuit Breaker pattern*

Circuit Breaker

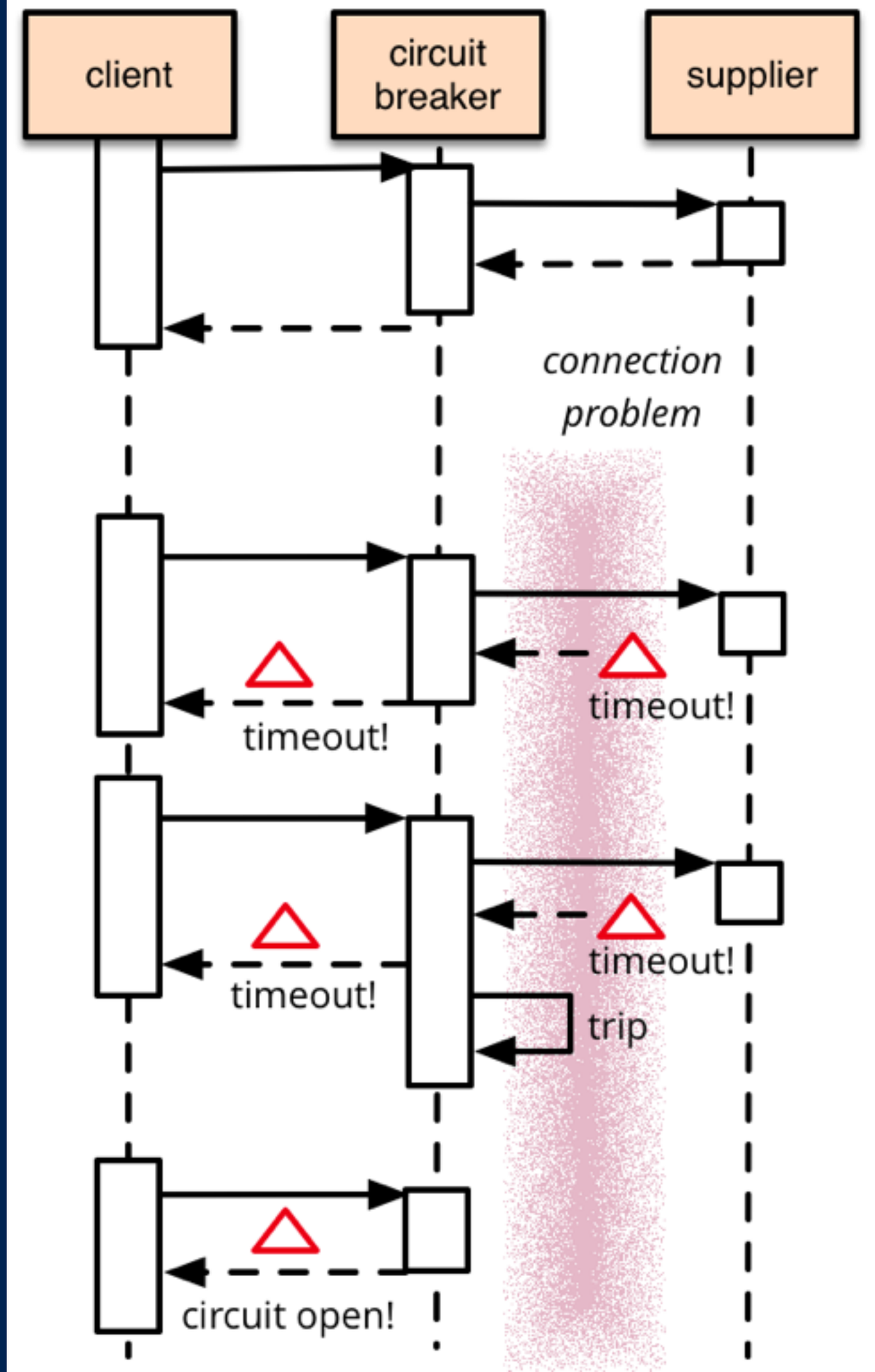
- wrap calls to services, monitor for failure

- if failures reach threshold, circuit breaker “trips”

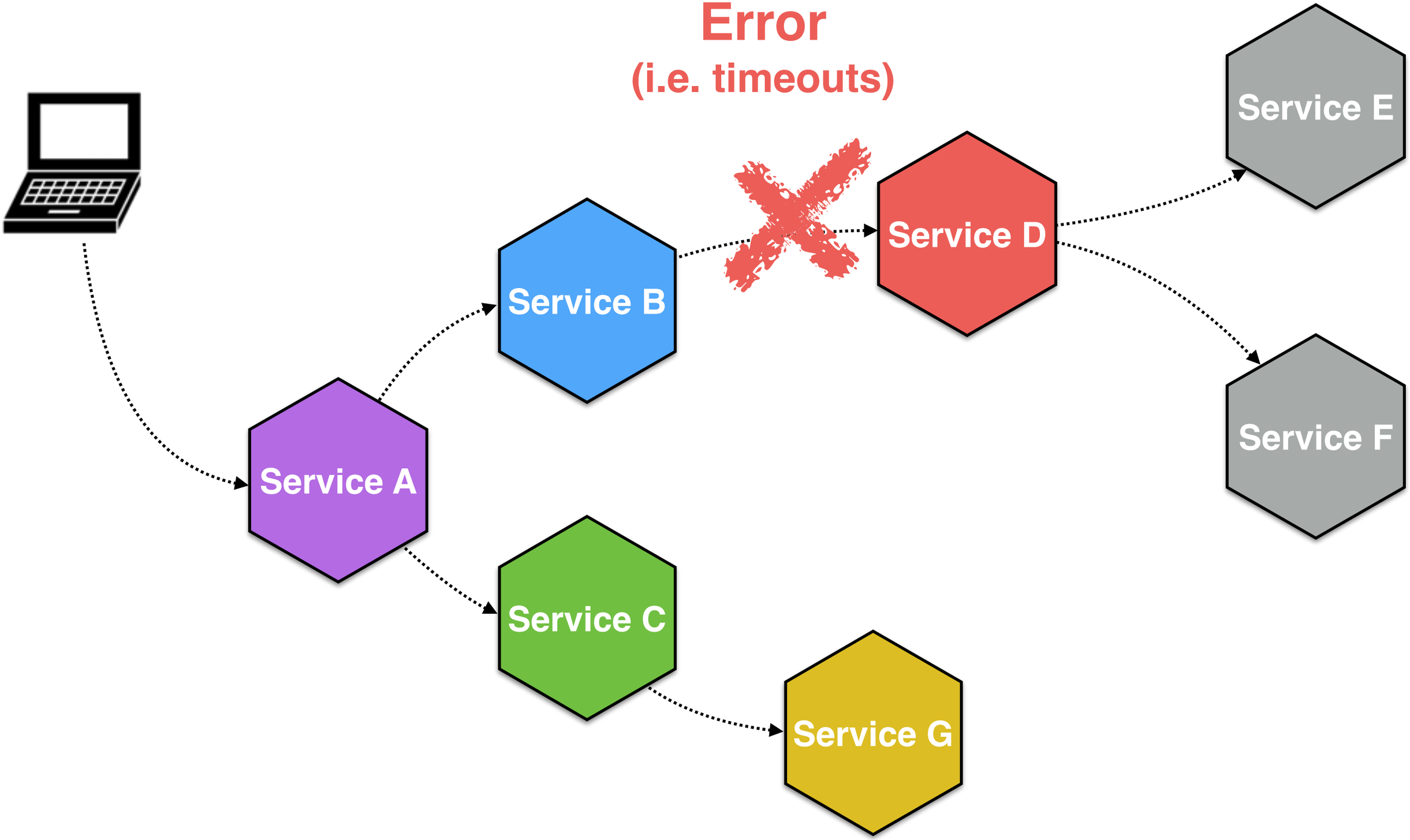
- subsequent calls return err or “fallback” behavior

- circuit checks when service becomes available

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



Error
(i.e. timeouts)

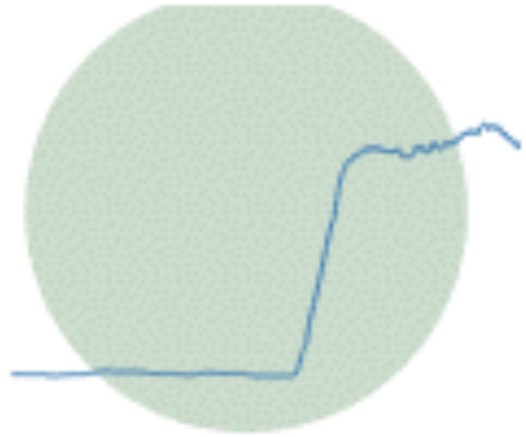


Circuit Breaker

```
@Component
public class ProductsClient {

    @Autowired
    private RestTemplate restTemplate;

    @HystrixCommand(fallbackMethod = "defaultProducts")
    public ResponseEntity getProducts() {
        ParameterizedTypeReference<Resource> responseType = new ParameterizedTyp

        ResponseEntity exchange =
                this.restTemplate.exchange(
                        "http://stuff-products/products",
                        HttpMethod.GET,
                        null,
                        responseType);
        return exchange;
    }

    public ResponseEntity defaultProducts(){
        ResponseEntity resp = new ResponseEntity(HttpStatus.SERVICE_UNAVAILABLE)
        return resp;
    }
```

Circuit Breaker (Hystrix)

Circuit Sort: Error then Volume | Alphabetical | Volume | Error | Mean | Median | 90 | 99 | 99.



getProducts

583	0	0.0 %
0	0	
	0	

Host: 58.3/s
Cluster: 58.3/s
Circuit Closed

Hosts	1	90th	17ms
Median	14ms	99th	28ms
Mean	14ms	99.5th	30ms

getCart

0	0	0.0 %
0	0	
	0	

Host: 0.0/s
Cluster: 0.0/s
Circuit Closed

Hosts	1	90th	0ms
Median	0ms	99th	0ms
Mean	0ms	99.5th	0ms

Thread Pools

Sort: Alphabetical | Volume |

ProductsClient

Host: 55.5/s
Cluster: 55.5/s

Active	1	Max Active	1
Queued	0	Executions	555
Pool Size	10	Queue Size	5

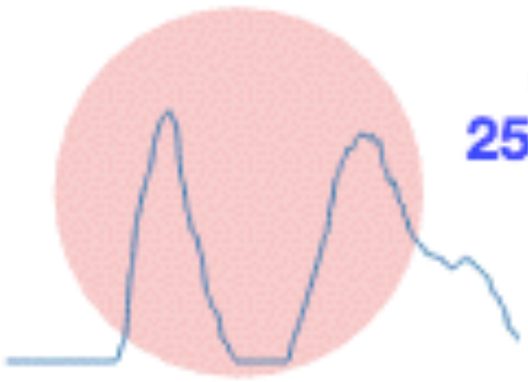
CartClient

Host: 0.0/s
Cluster: 0.0/s

Active	0	Max Active	0
Queued	0	Executions	0
Pool Size	10	Queue Size	5

Circuit Breaker (Hystrix)

Circuit Sort: Error then Volume | Alphabetical | Volume | Error | Mean | Median | 90 | 99 | 99.5



getProducts

0	0	100.0 %
252	0	
	0	

Host: **38.5/s**
Cluster: **38.5/s**
Circuit Open

Hosts	1	90th	0ms
Median	0ms	99th	0ms
Mean	0ms	99.5th	0ms

getCart

0	0	0.0 %
0	0	
	0	

Host: **0.0/s**
Cluster: **0.0/s**
Circuit Closed

Hosts	1	90th	0ms
Median	0ms	99th	0ms
Mean	0ms	99.5th	0ms

Thread Pools

Sort: Alphabetical | Volume |

ProductsClient

Host: **0.0/s**
Cluster: **0.0/s**

Active	0	Max Active	0
Queued	0	Executions	0
Pool Size	10	Queue Size	5

CartClient

Host: **0.0/s**
Cluster: **0.0/s**

Active	0	Max Active	0
Queued	0	Executions	0
Pool Size	10	Queue Size	5

Circuit Breaker

- deal with latency and failures
- builds resiliency into your application
- avoid cascading failures
- Hystrix Dashboard provides a view of endpoints and monitor failures

https://github.com/Netflix/Hystrix

Scale the App via Process Model

Scaling Microservices

$ cf push <app>

```
name                requested state   instances   memory   disk   urls
cart                started           1/1         256M     1G     stuff-cart.mybluemix.net
orders              started           1/1         256M     1G     stuff-orders.mybluemix.net
products            started           1/1         256M     1G     stuff-products.mybluemix.net
service-registry    started           1/1         256M     1G     stuff-eureka.mybluemix.net
store-api-gateway   started           1/1         256M     1G     stuff-api-gateway.mybluemix.net
storefront          started           1/1         256M     1G     stuff-store.mybluemix.net
```

Scaling Microservices

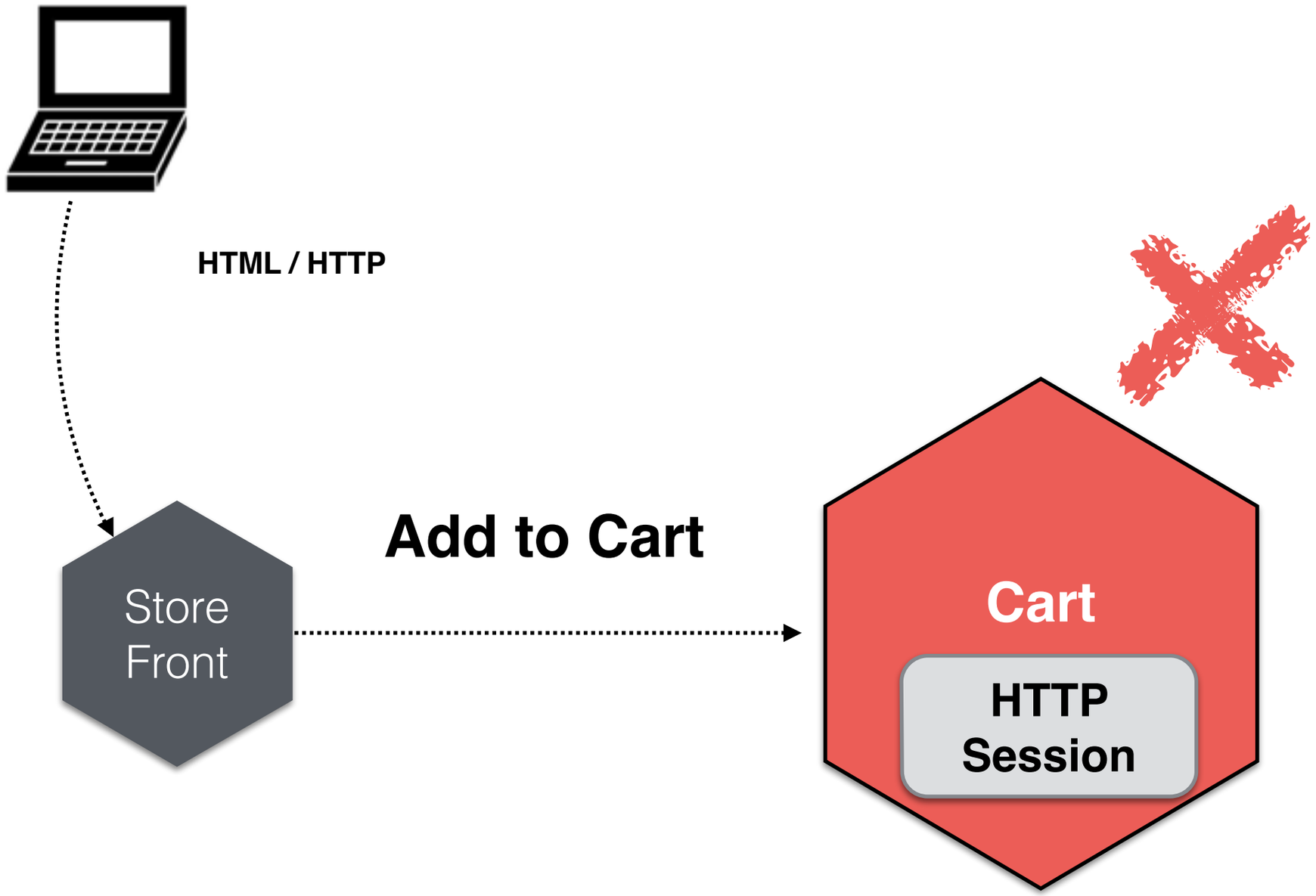
$ cf scale <app> -i 2

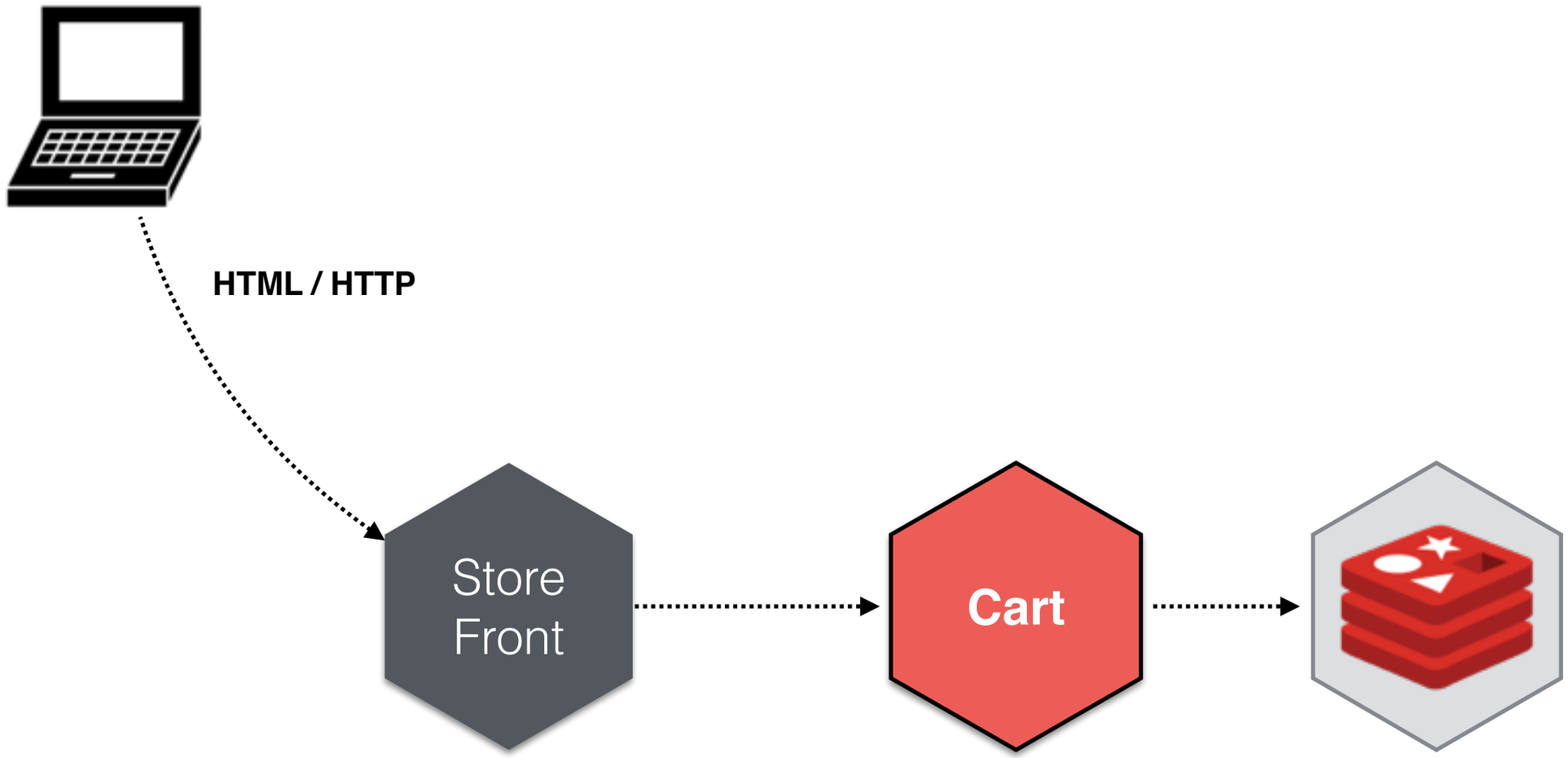
```
~/Development/microservice-demo/products(master ✓) $ cf scale products -i 2
Scaling app products in org stgodard@ca.ibm.com / space demo as stgodard@ca.ibm.com...
OK
~/Development/microservice-demo/products(master ✓) $ cf a
Getting apps in org stgodard@ca.ibm.com / space demo as stgodard@ca.ibm.com...
OK

name                requested state   instances   memory   disk   urls
cart                started           1/1         256M     1G     stuff-cart.mybluemix.net
orders              started           1/1         256M     1G     stuff-orders.mybluemix.net
products            started           2/2         256M     1G     stuff-products.mybluemix.net
service-registry    started           1/1         256M     1G     stuff-eureka.mybluemix.net
store-api-gateway   started           1/1         256M     1G     stuff-api-gateway.mybluemix.net
storefront          started           1/1         256M     1G     stuff-store.mybluemix.net
~/Development/microservice-demo/products(master ✓) $
```

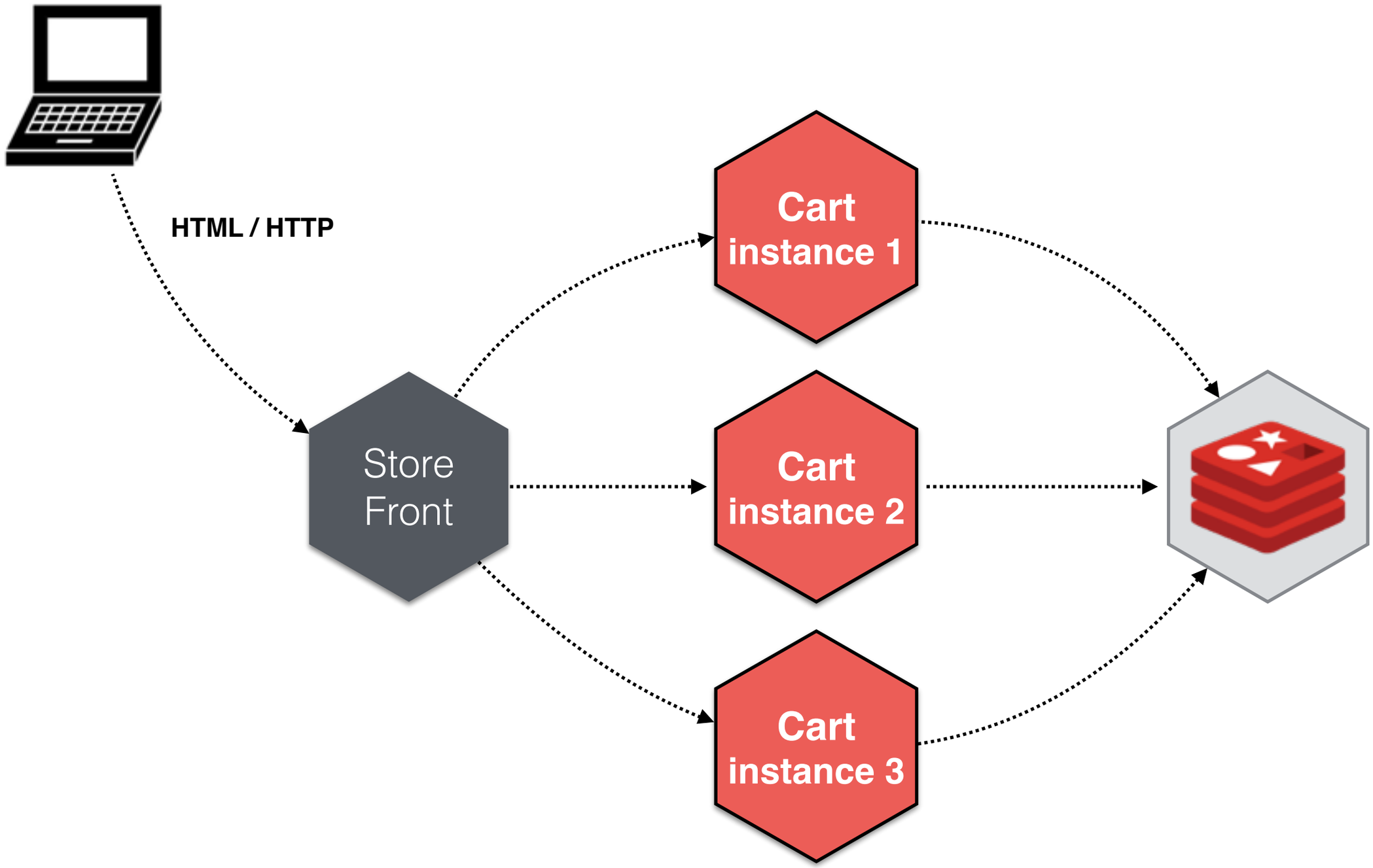
Stateless

Processes

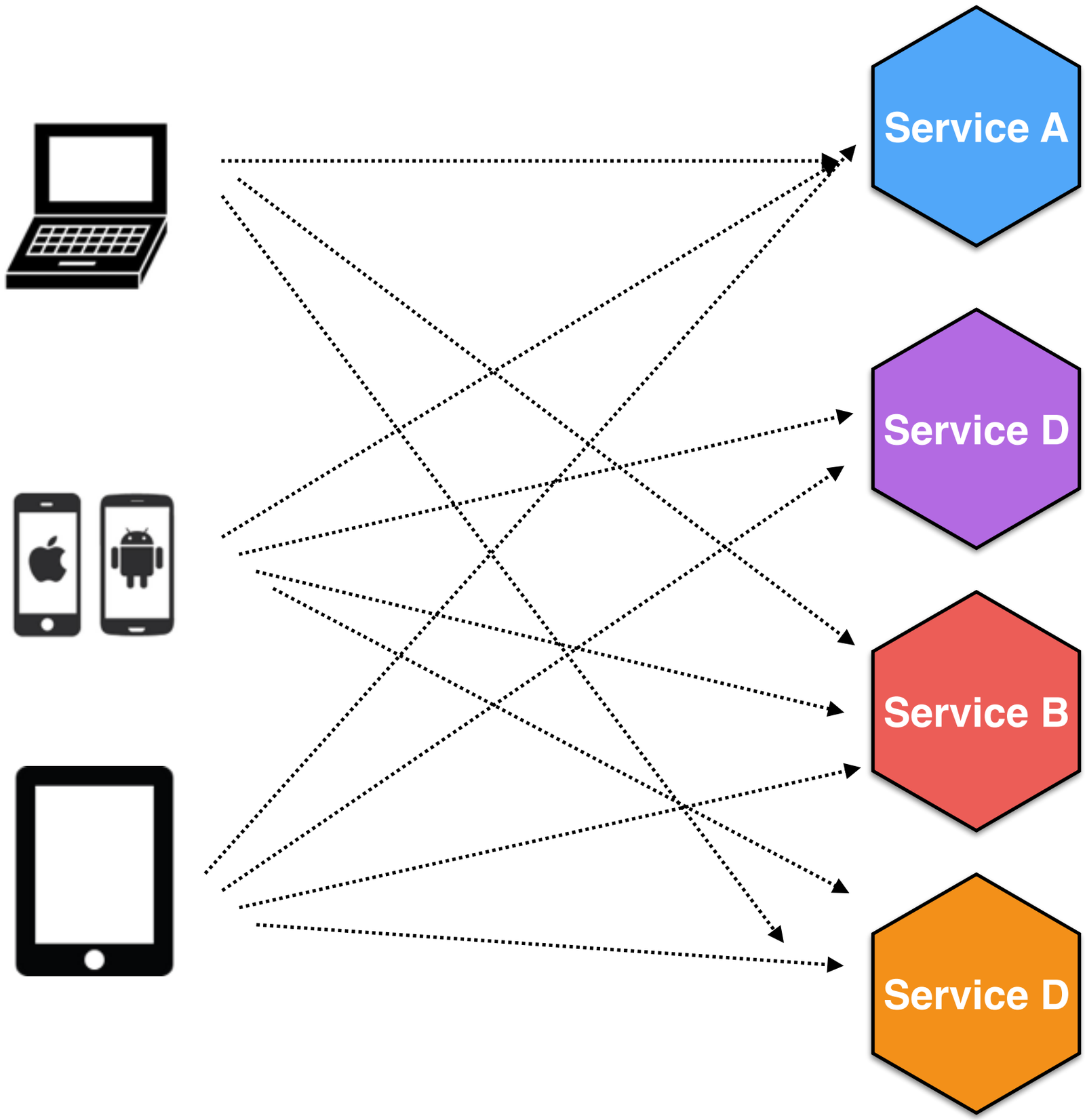
** store state in distributed shared storage*

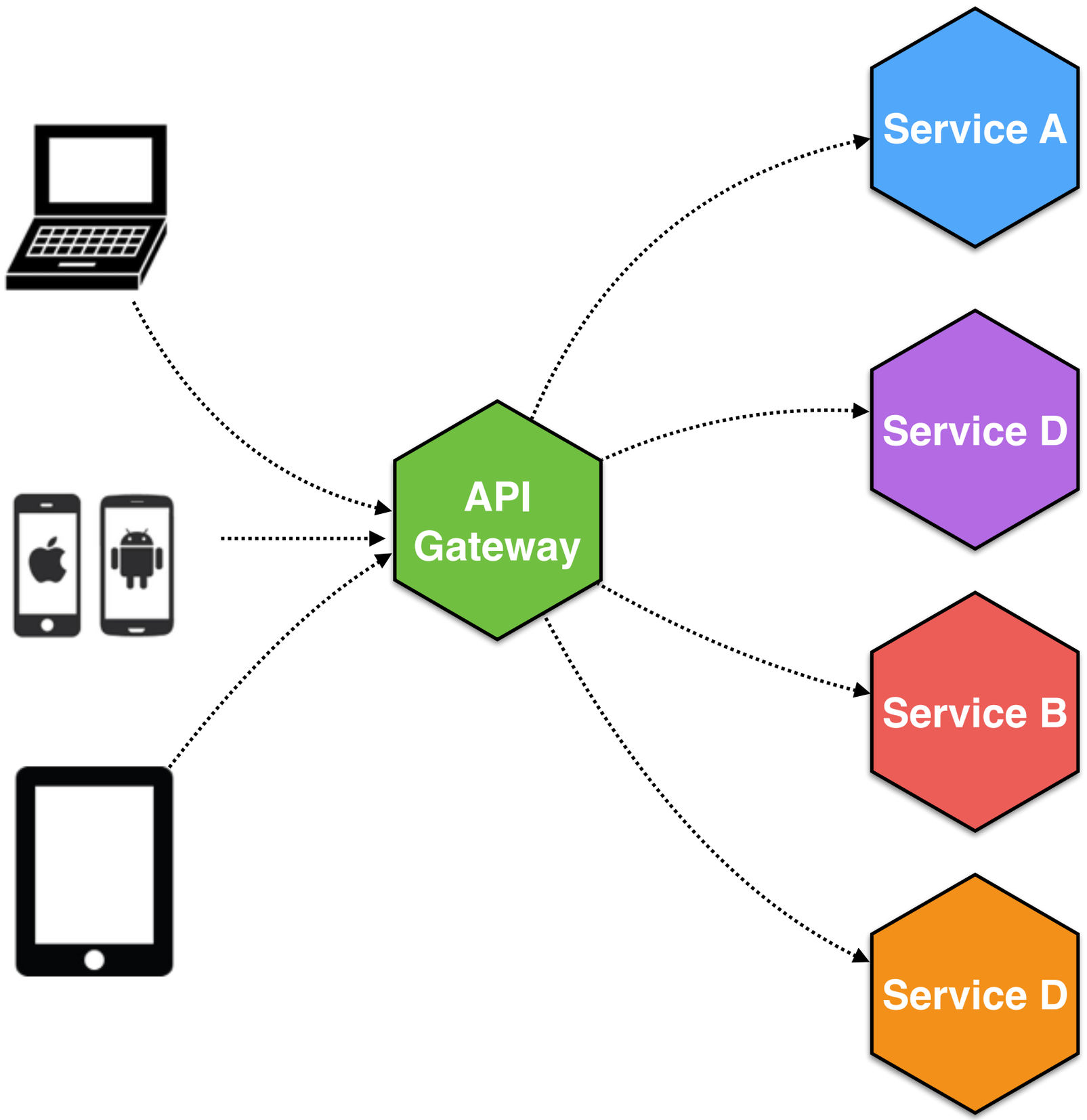

HTML / HTTP
Add to Cart
Store Front
Cart
HTTP Session


HTML / HTTP
Store Front
Cart


HTML / HTTP
Store Front
Cart instance 1
Cart instance 2
Cart instance 3

API Gateway


Service A
Service D
Service B
Service D


Service A
API Gateway
Service D
Service B
Service D

12factor.net

For more info on key practices for building apps that run “as a service”

Microservices

- Don’t start with micro services
- Distribution Complexity
  - Logging, monitoring, decentralization

- Requires a culture of automation
  - automated testing (CI)
  - automated deployments (CD)
  - rapid provisioning

- Team structure / organizational change
- Avoid creating too many micro services

Microservices

- Loosely coupled
- Independently deployable
- Single code base per micro service
  - smaller teams
  - quicker app startup times / productivity
- Easier to
  - scale specific workloads
  - move to new technology
  - replace a micro service
- Smaller / more frequent releases
  - Continuous Delivery

References

- **Microservices:**
  - http://martinfowler.com/microservices.html
- **Spring Boot / Cloud Netflix**
  - http://projects.spring.io/spring-boot/
  - http://cloud.spring.io/spring-cloud-netflix/

@markstgodard

Thanks

markstgodard@gmail.com