Evolution of Integration and Microservices with Service Mesh and Ballerina

@christianposta

Christian Posta

Chief Architect, cloud application development

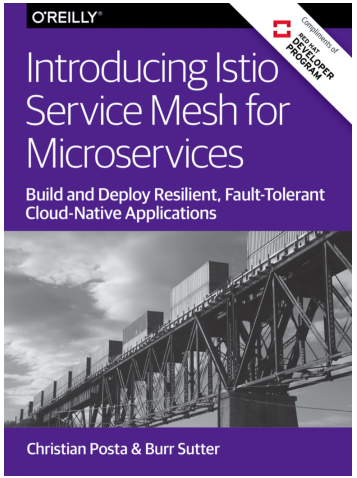
Twitter: @christianposta

Blog: http://blog.christianposta.com

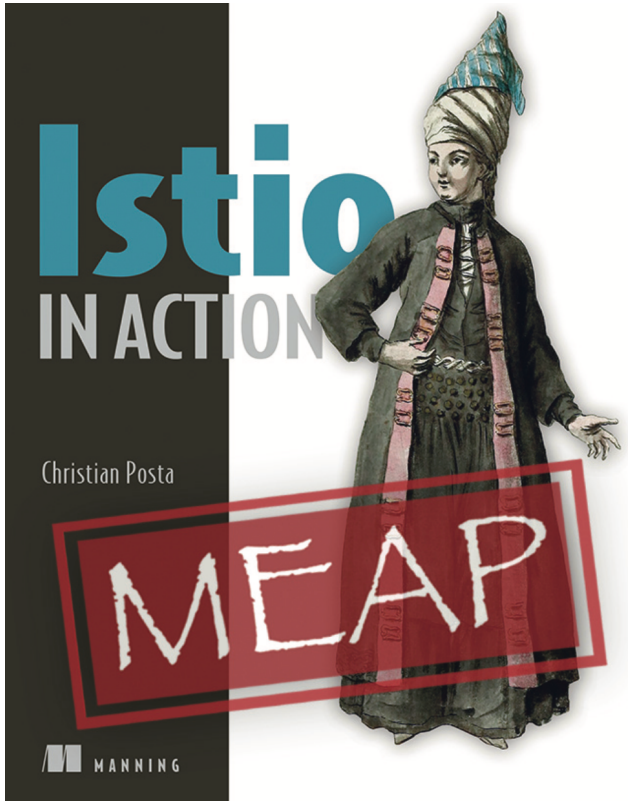
Email: christian@redhat.com

Slides: http://slideshare.net/ceposta

- Author “Microservices for Java developers” and “Introducing Istio Service Mesh”
- Committer/contributor lots of open-source projects
- Blogger, speaker, mentor, leader

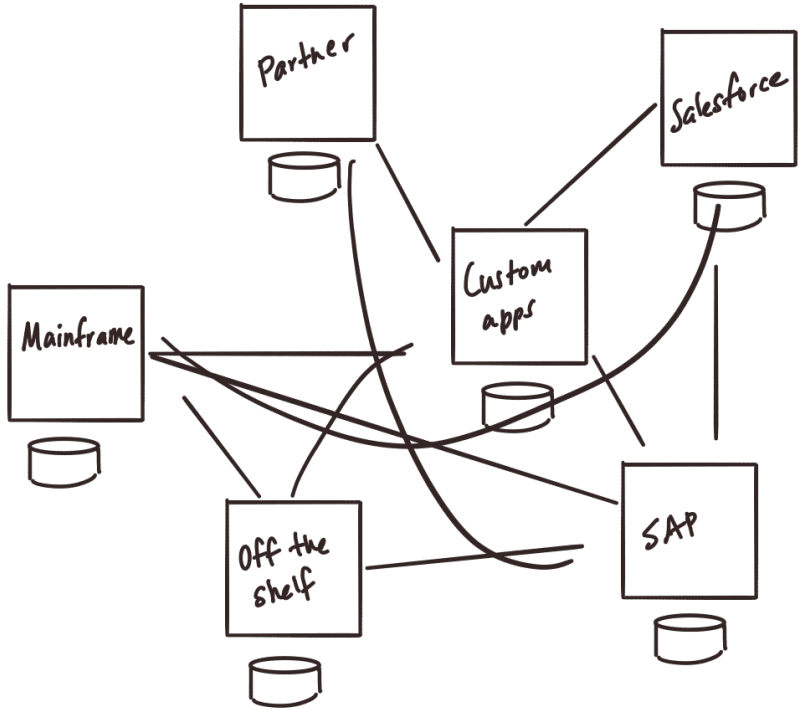


https://www.manning.com/books/istio-in-action

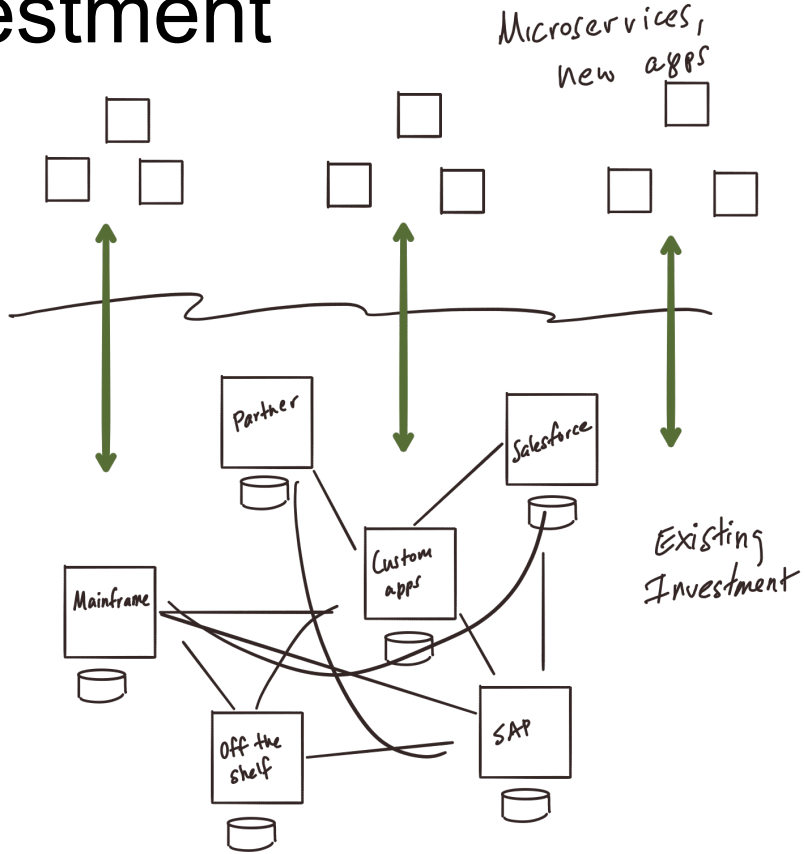


"This really is an innovative approach, but I'm afraid we can't consider it. It's never been done before."

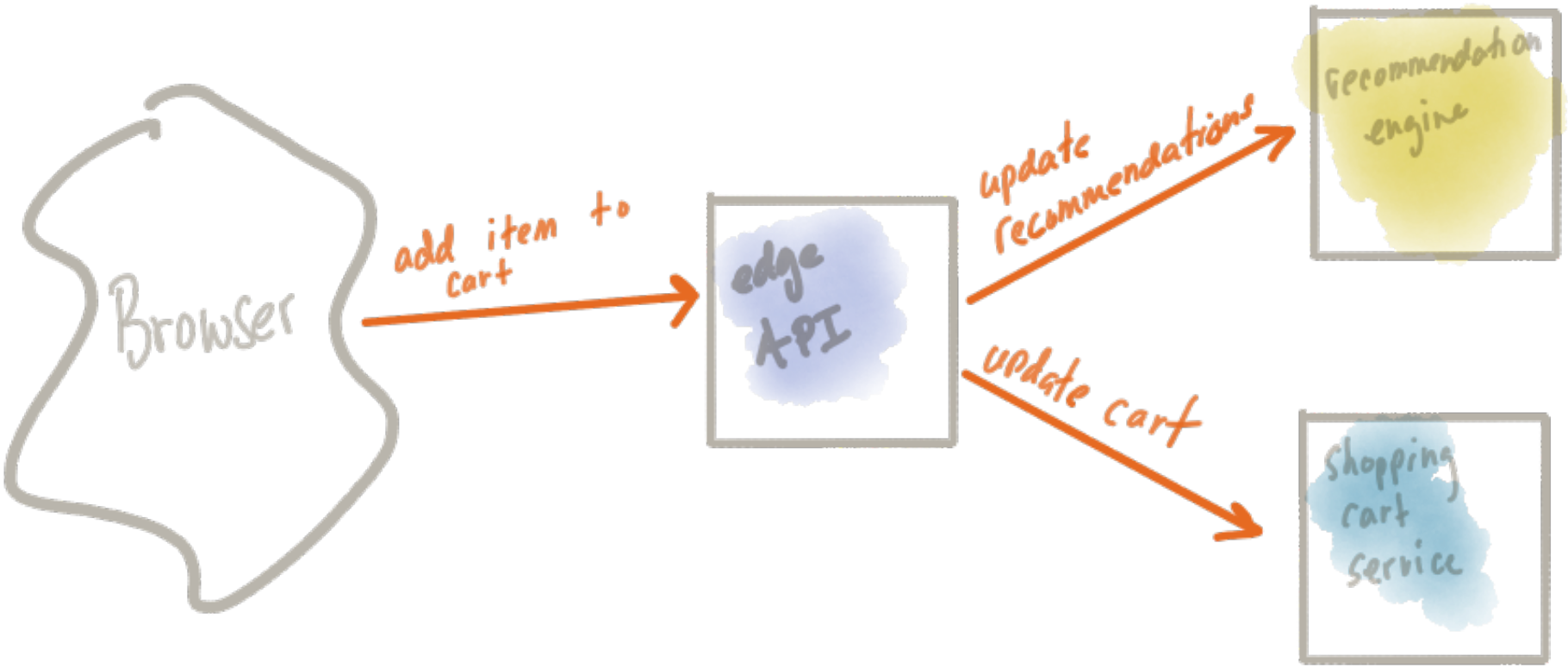
Existing investment



New capabilities need to work with existing investment



As we move to services architectures, we push the complexity to the space *between* our services.



@christianposta

Application integration

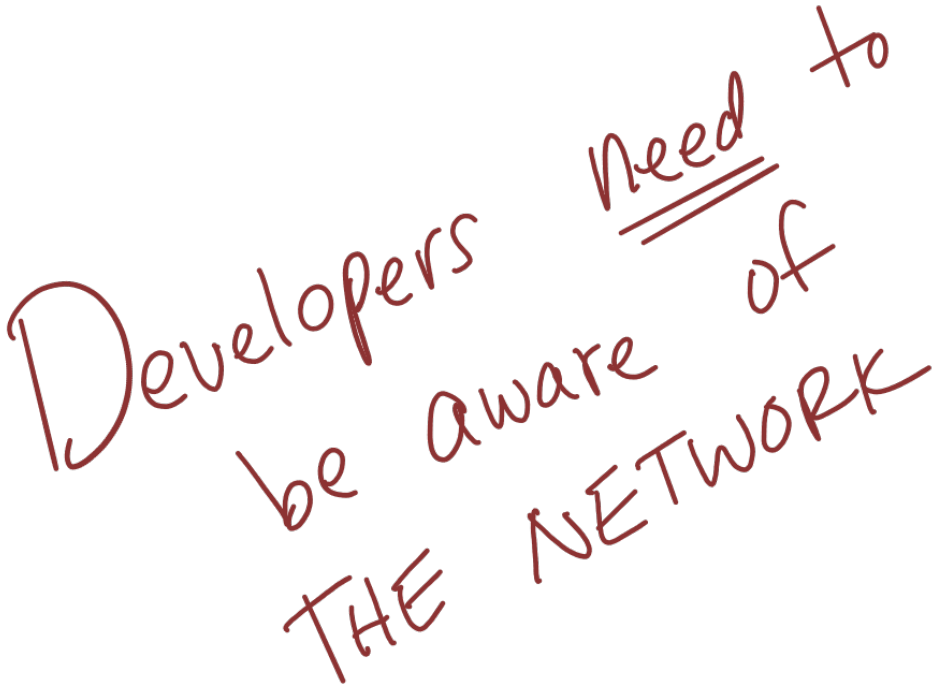
- Orchestrate calls across multiple microservices
- Calls in parallel, sequential, etc
- Aggregate, combine, transform, split, on “messages”
- Deal with errors, unexpected results

Application integration

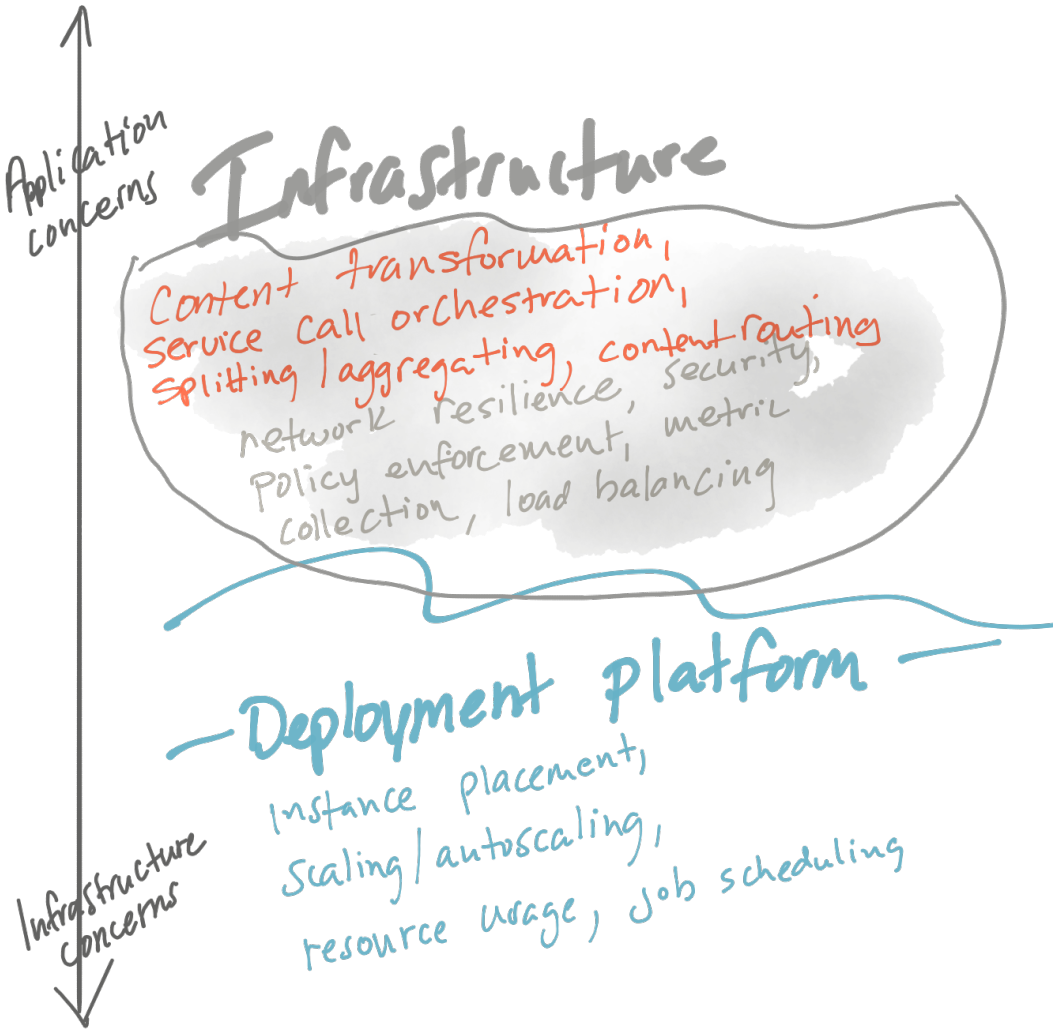
- Deal with atomicity / consistency issues
- Message idempotency / message de-dupe
- APIs / DDD anti-corruption layers / adapters
- Tie in with existing “backend systems”
- Deal with making calls over the network

Application networking

- Service discovery
- Load balancing
- Timeouts
- Retries
- Circuit breaking
- Rate limiting



Application safety and correctness in a distributed system is the *responsibility* of the application teams.


Application concerns
Infrastructure
Content transformation,
service call orchestration,
splitting/aggregating, content routing
network resilience, security,
policy enforcement, metric
collection, load balancing
Deployment platform
instance placement,
scaling/autoscaling,
resource usage, job scheduling
Infrastructure concerns

Integration is part of the application-development process.

Meet Ballerina

http://ballerina.io

Ballerina is…

- A new language purpose-built for creating services/APIs and integrating with existing investments/services
- Built by WSO2
- Static, strong typing with language constructs that make services and service interaction first-class citizens
- Strong focus on network awareness

Ballerina is…

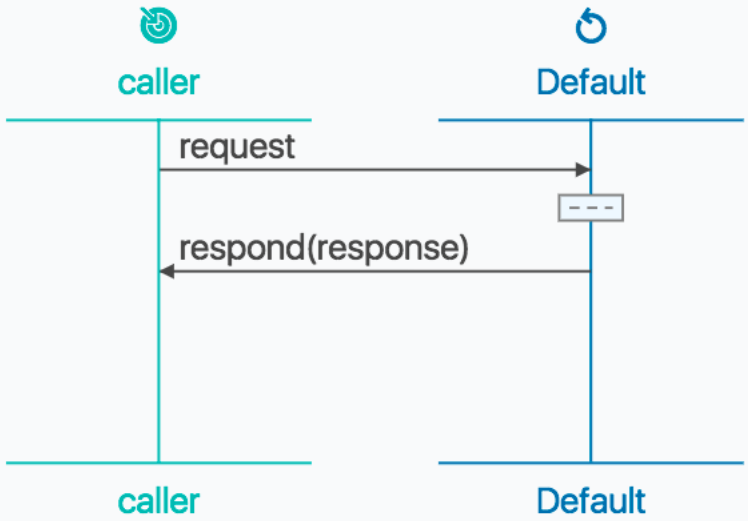
- Built around a “sequence-diagram” mental model
- Round-trip development as a first-class citizen
- Native concurrency model
- Solves problems around application integration like:
  - Service call orchestration
  - Aggregating responses
  - Data security
  - Transactions, compensations
  - Quickly build APIs based on OAPI
  - Reusable protocol / backend adapters
  - Long-running execution with checkpointing
  - Stream based processing

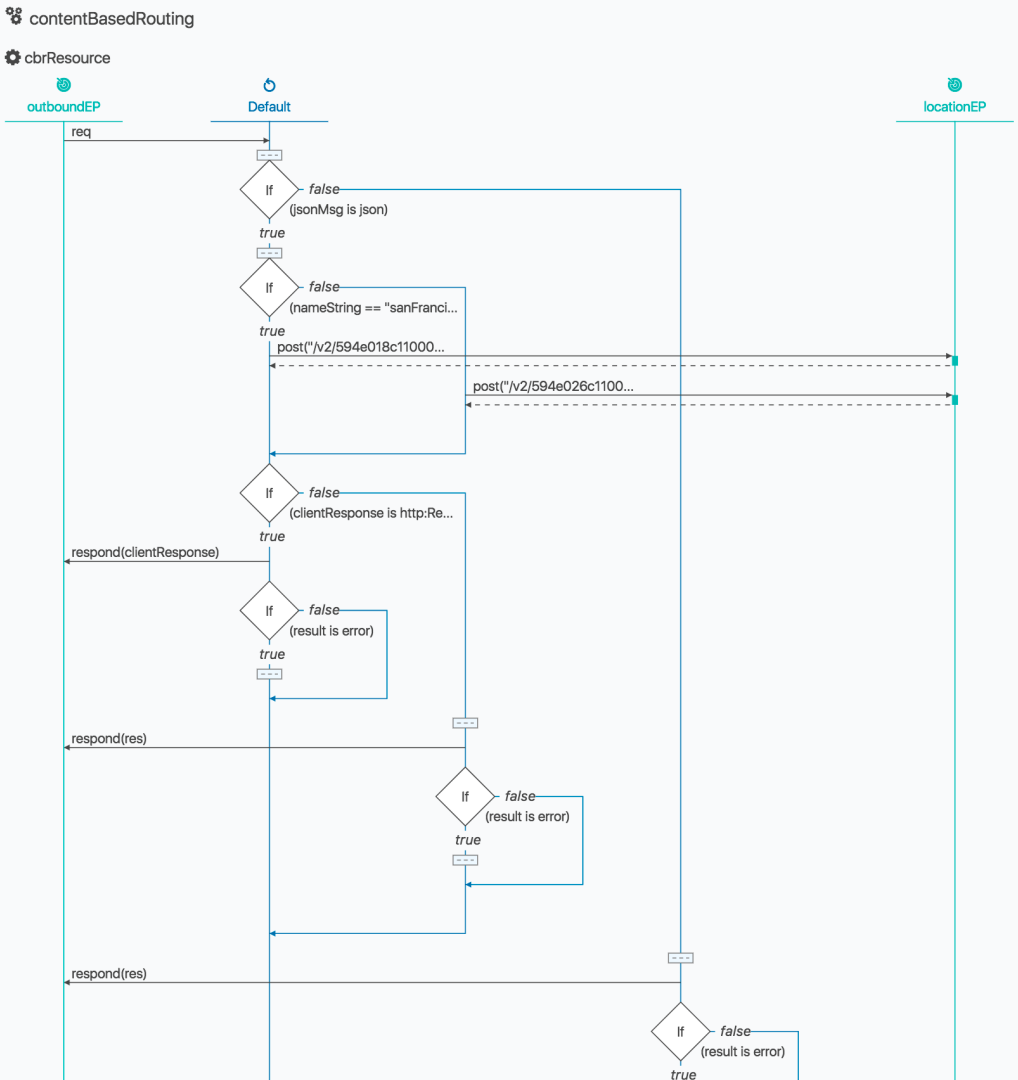
```
// A system module containing protocol access constructs
// Module objects referenced with 'http:' in code
import ballerina/http;
import ballerina/io;

listener http:Listener recommendationListen = new  (8080);

service sayhello on recommendationListen {

    @http:ResourceConfig {
        methods: ["GET"],
        path: "/"
    }
    resource function showRecommendations(http:Caller caller, http:Request request) {

        // Create object to carry data back to caller
        http:Response response = new;
        response.setTextPayload("Hello, KubeCon NA!");
        _ = caller -> respond(response);
    }
}
```

Edit

sayhello

showRecommendations




contentBasedRouting
cbrResource
outboundEP
Default
locationEP
req
If
false
(jsonMsg is json)
true
If
false
(nameString == "sanFranci...
true
post("/v2/594e018c11000...
post("/v2/594e026c1100...
If
false
(clientResponse is http:Re...
true
respond(clientResponse)
If
false
(result is error)
true
respond(res)
If
false
(result is error)
true
respond(res)
If
false
(result is error)
true

So: what about application networking?

Application networking

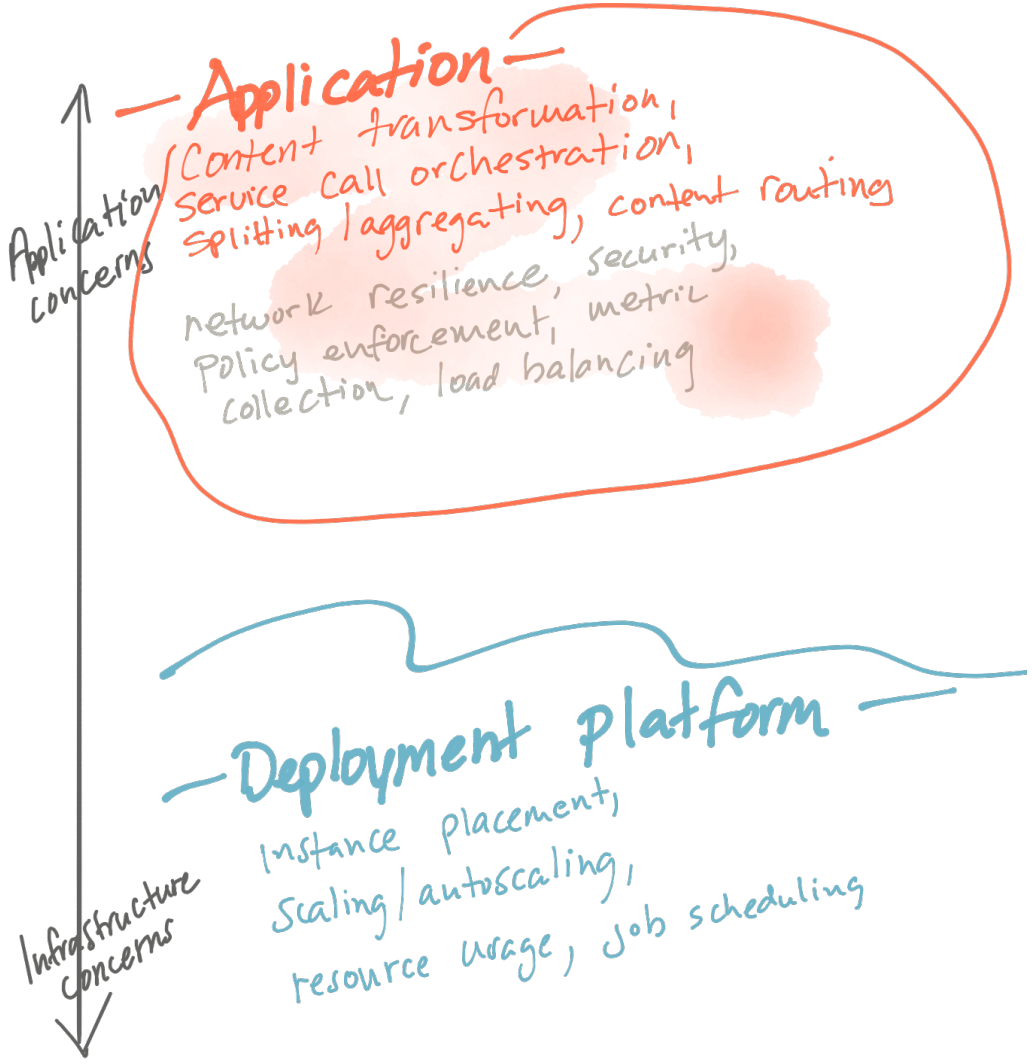
- Service discovery
- Retries
- Timeouts
- Load balancing
- Rate limiting
- Thread bulk heading
- Circuit breaking

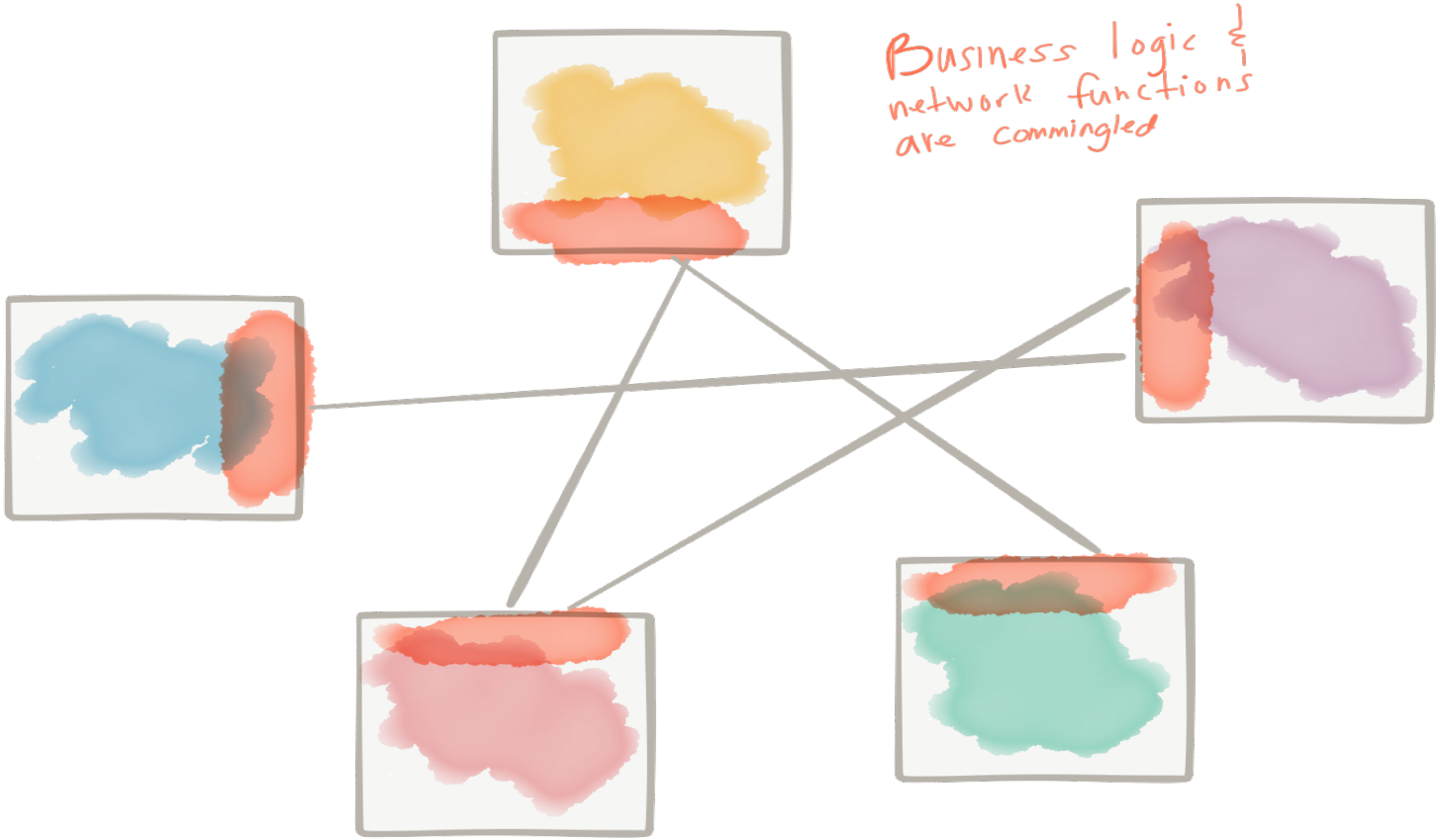
- Edge/DMZ routing
- Surgical, per-request routing
- A/B rollout
- Traffic shaping
- Internal releases / dark launches
- Request shadowing
- Fault injection

- adaptive, zone-aware
- Deadlines
- Health checking
- Stats, metric, collection
- Logging
- Distributed tracing
- Security

“Microservices” patterns

- Netflix Hystrix (circuit breaking / bulk heading)
- Netflix Zuul (edge router)
- Netflix Ribbon (client-side service discovery / load balance)
- Netflix Eureka (service discovery registry)
- Brave / Zipkin (tracing)
- Netflix spectator / atlas (metrics)


Application
Content transformation,
service call orchestration,
splitting/aggregating, content routing
network resilience, security,
policy enforcement, metric
collection, load balancing
Application concerns
Deployment platform
instance placement,
scaling/autoscaling,
resource usage, job scheduling
Infrastructure concerns



@christianposta

http://bit.ly/application-networking

Screw Java - I’m using NodeJS!

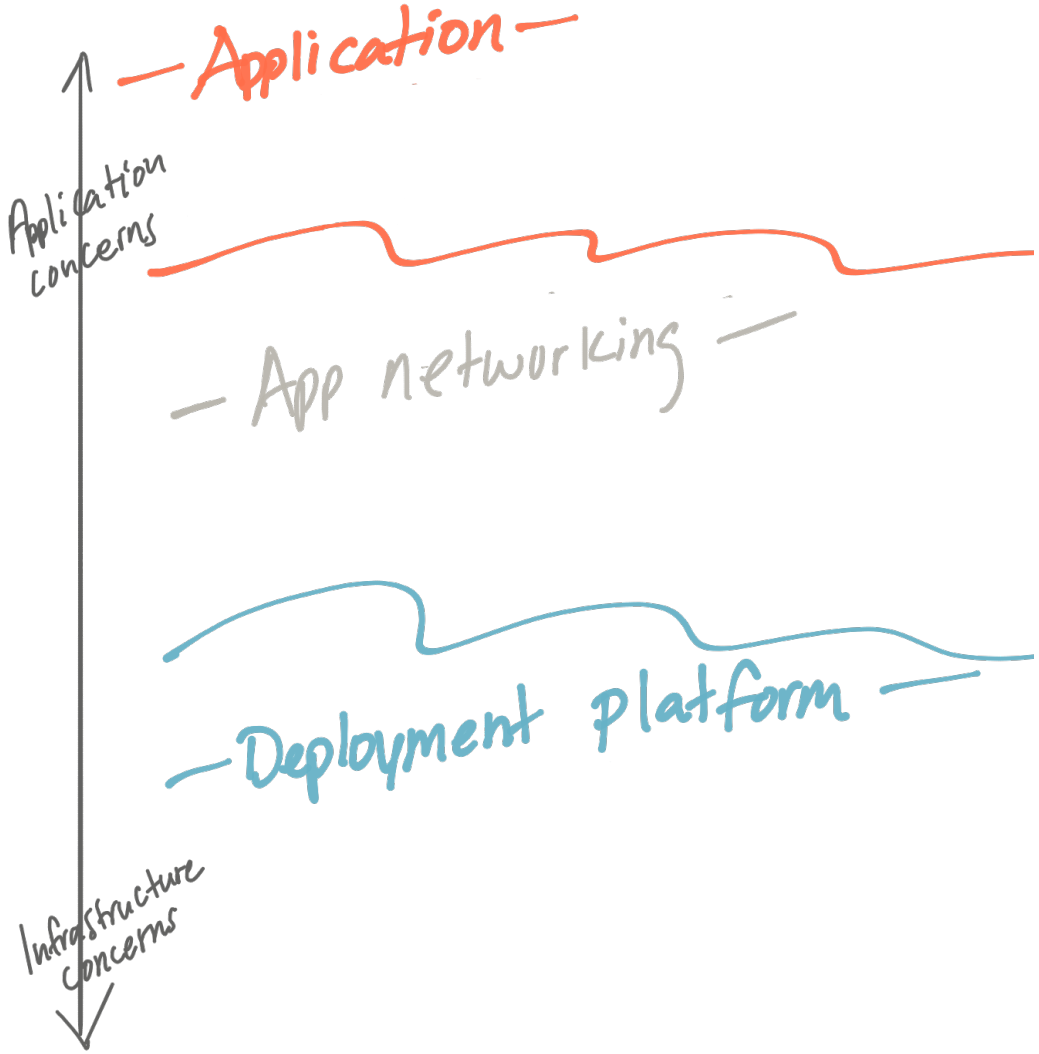
JavaScript is for rookies, I use Go!

But python is so pretty!

I prefer unreadability… Perl for me!

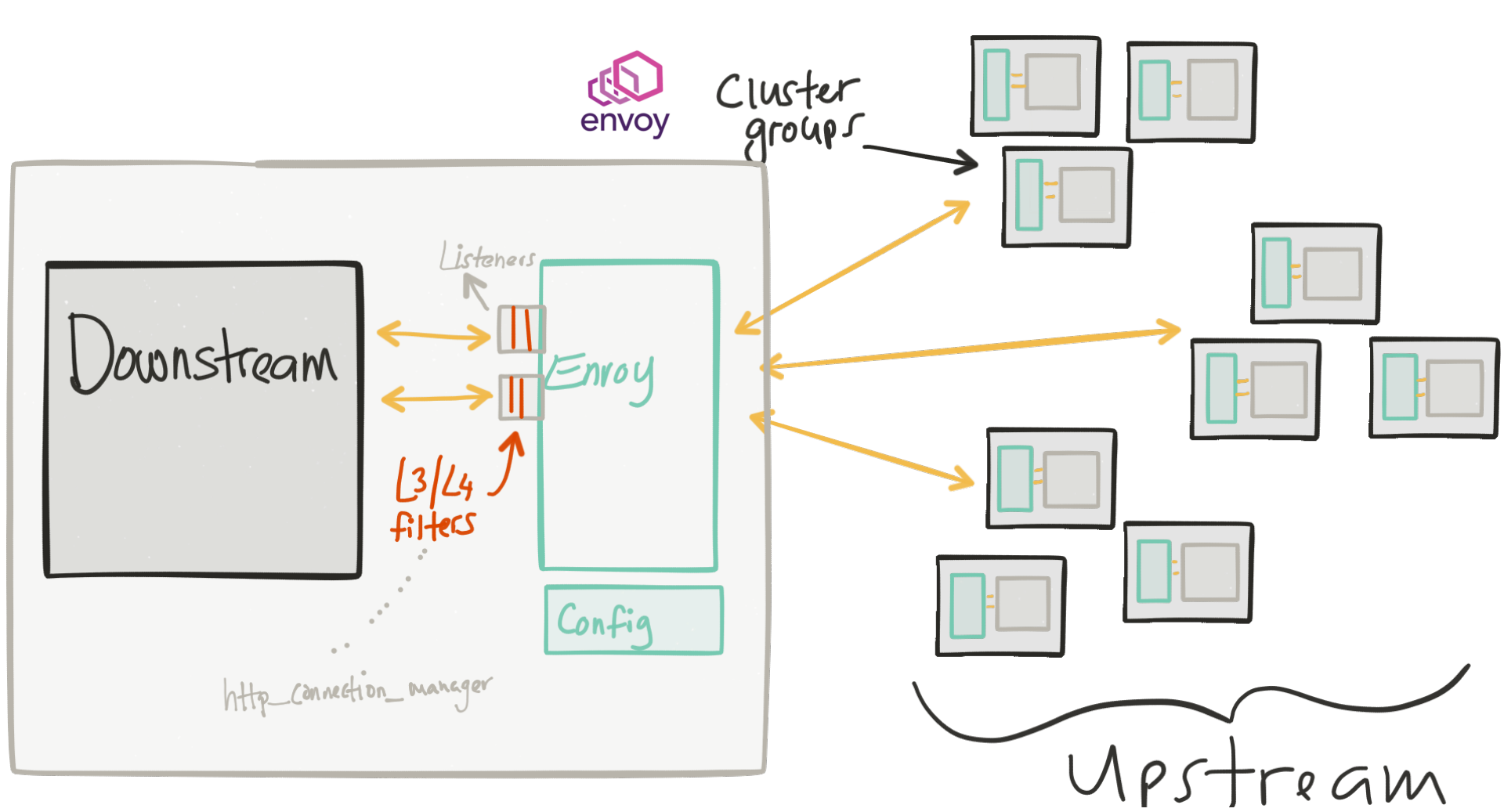
In practice, *operability* of our services becomes a top priority very fast

Let’s optimize for operability

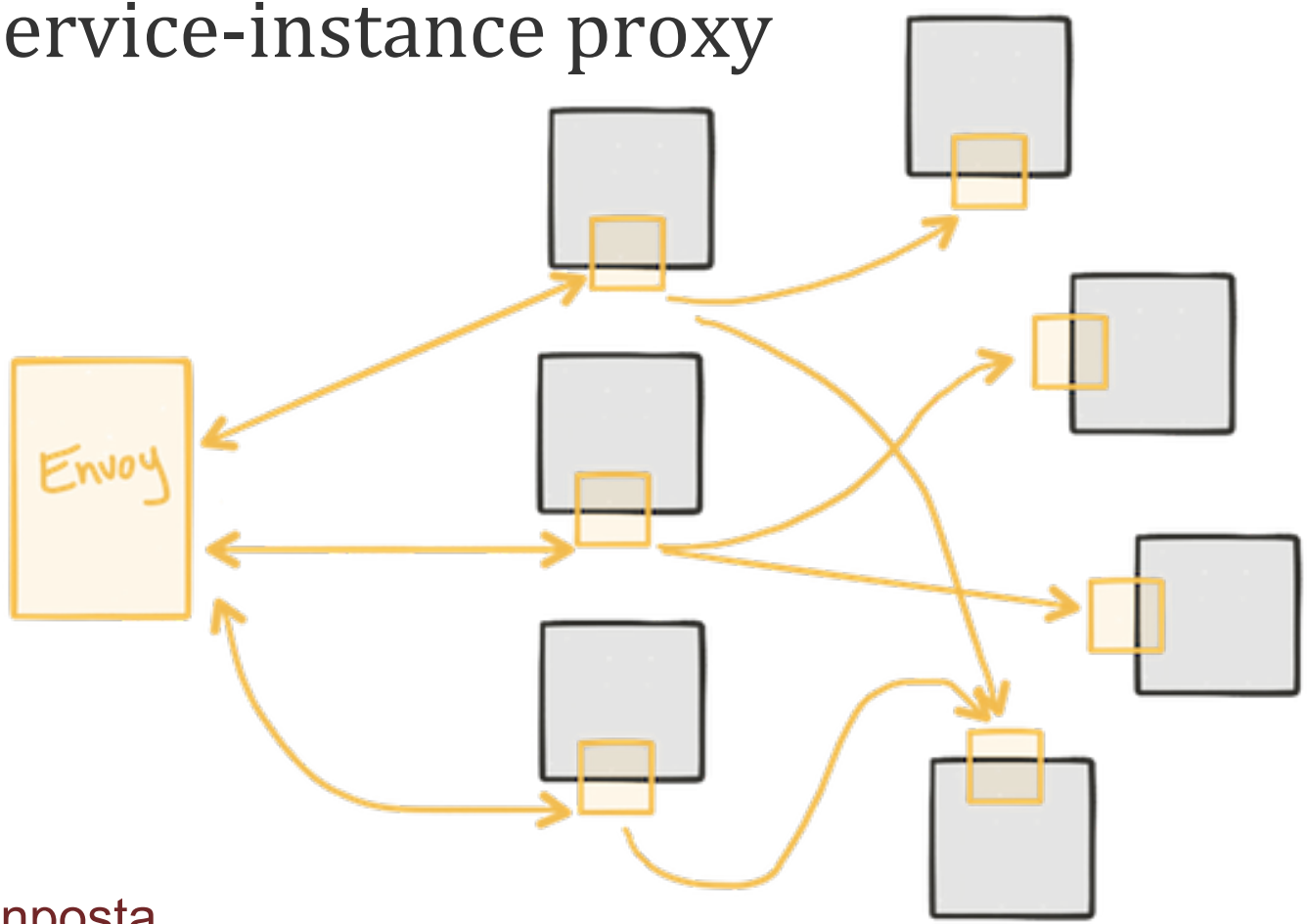

Application
Application concerns
App networking
Deployment platform
Infrastructure concerns

Meet Envoy Proxy

http://envoyproxy.io


envoy
Cluster groups
Listeners
Downstream
Envoy
L3/L4 filters
Config
http_connection_manager
Upstream

As a service-instance proxy



Time for definitions:

A *service mesh* is a distributed application infrastructure that is responsible for handling network traffic on behalf of the application in a transparent, out of process manner.

A service mesh helps to solve problems related to *resiliency*, *security*, *observability*, and *routing control*.

Service mesh technologies typically provide:

- Service discovery / Load balancing
- Secure service-to-service communication
- Traffic control / shaping / shifting
- Policy / Intention based access control
- Traffic metric collection
- Service resilience

Open-source, service-mesh implementations

- Istio.io

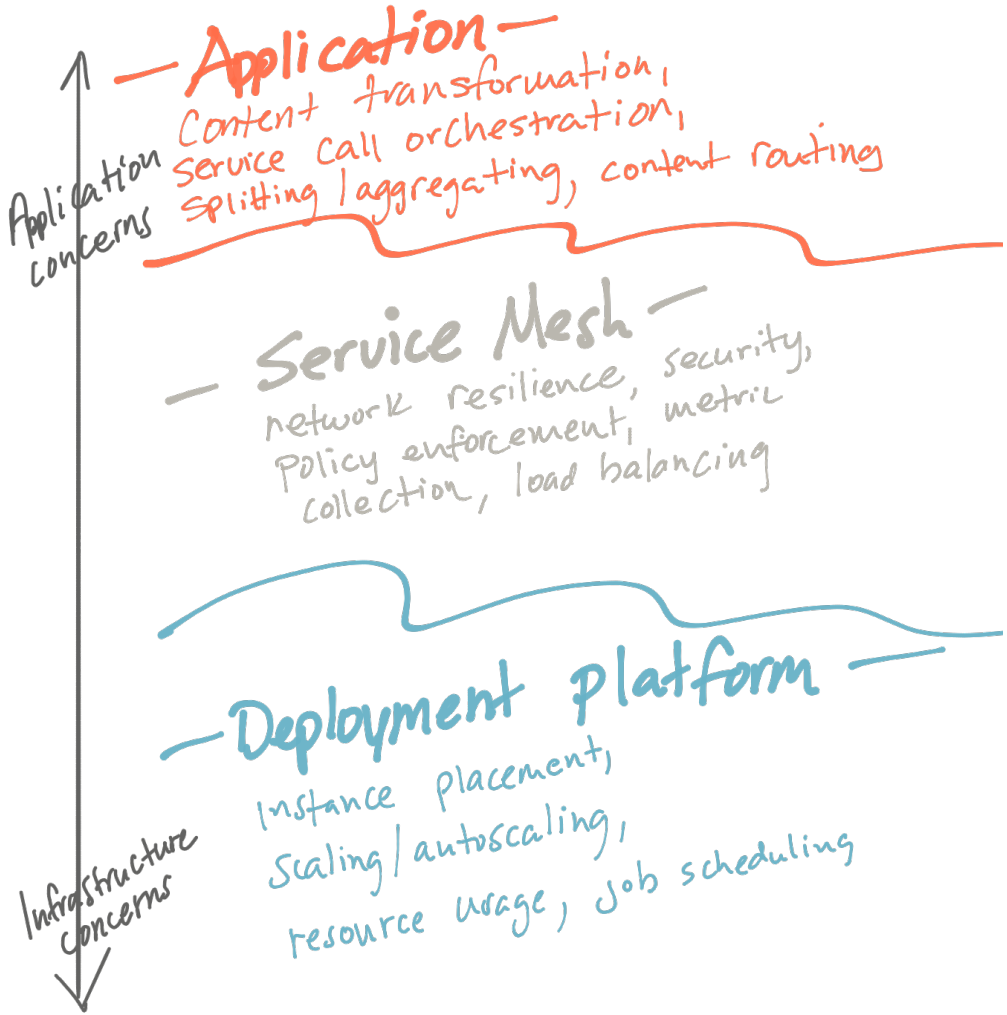
  http://istio.io

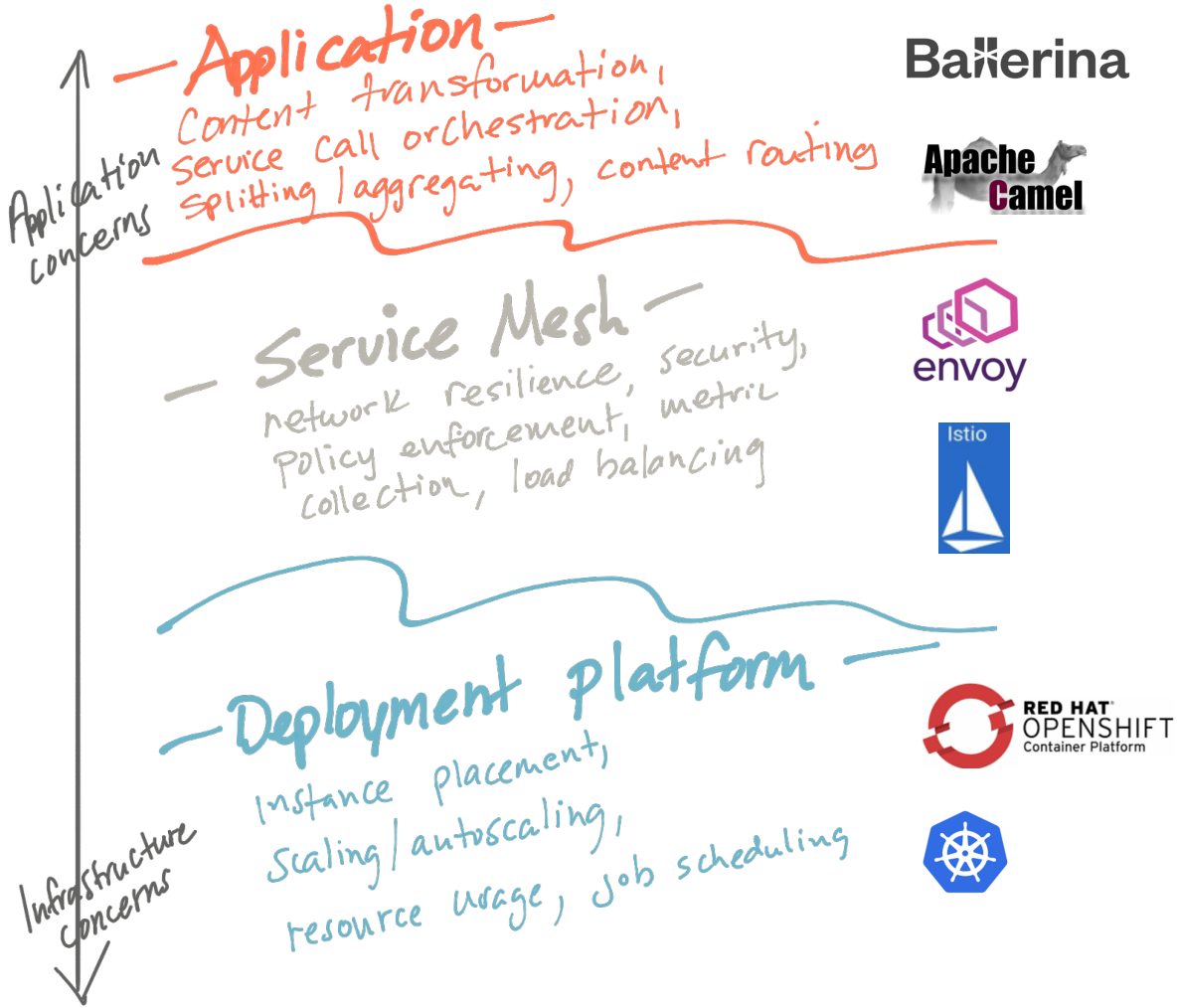
- Consul Connect

  http://consul.io

- LinkerD

  http://linkerd.io


Application
Content transformation,
service call orchestration,
splitting/aggregating, content routing
Application concerns
Service Mesh
network resilience, security,
policy enforcement, metric
collection, load balancing
Deployment platform
instance placement,
scaling/autoscaling,
resource usage, job scheduling
Infrastructure concerns


Application
Content transformation,
service call orchestration,
splitting/aggregating, content routing
Application concerns
Service Mesh
network resilience, security,
policy enforcement, metric
collection, load balancing
Deployment platform
instance placement,
scaling/autoscaling,
resource usage, job scheduling
Infrastructure concerns
Ballerina
Apache Camel
envoy
Istio
RED HAT OPENSHIFT Container Platform

Isn’t there overlap??

- Service discovery
- Load balancing
- Timeouts
- Retries
- Circuit breaking
- Rate limiting
- Distributed tracing
- … some others …

Key takeaways:

Leverage the service mesh for key, consistent application-networking behavior.

Develop a workflow for application teams that includes configuration of the service mesh as part of the application.

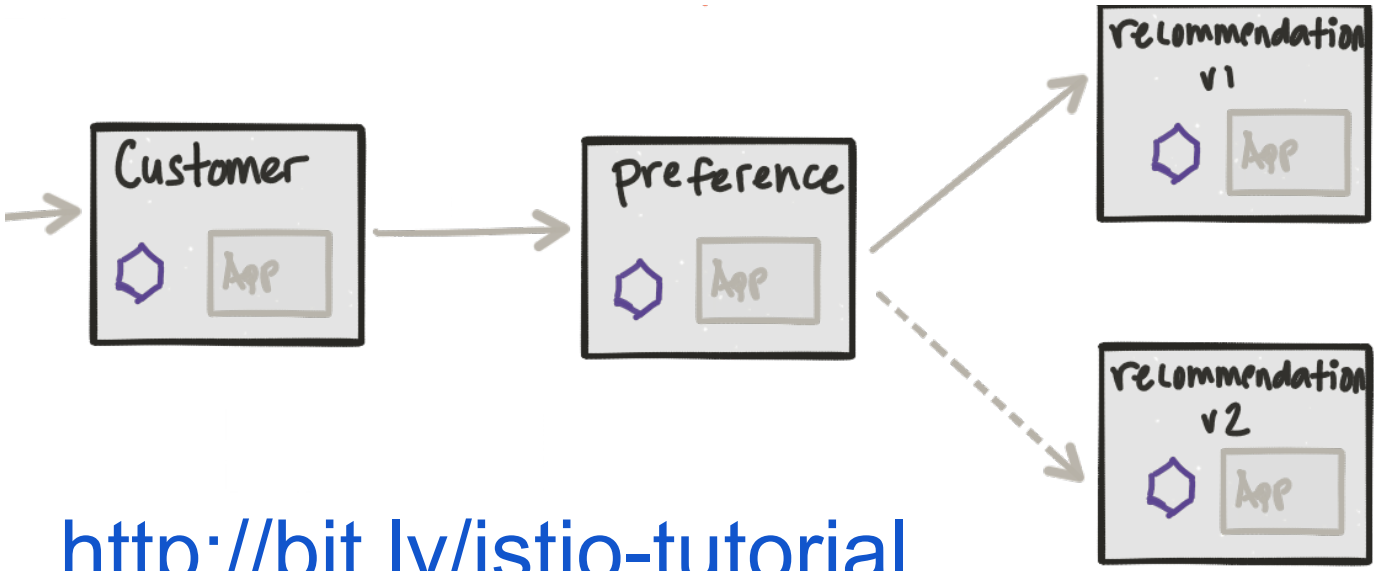
Opt for language-specific implementations when the general service mesh solution doesn't adequately solve a specific problem.

Key takeaways:

Resist the urge to put application-integration logic into the service mesh.

Understand the “why” of service mesh and seek to keep the boundary delineated

Demo!



http://bit.ly/istio-tutorial

Thanks!

Twitter: @christianposta

Blog: http://blog.christianposta.com

Email: christian@redhat.com

Slides: http://slideshare.net/ceposta

Follow up links:

- http://ballerina.io
- http://istio.io
- http://envoyproxy.io
- http://developers.redhat.com/blog
- http://blog.christianposta.com/istio-workshop/slides/
- http://blog.openshift.com
- https://www.redhat.com/en/open-innovation-labs

BTW: Hand drawn diagrams made with Paper by FiftyThree.com ☺