Introducing Envoy-based service mesh at Booking.com

Ivan Kruglov

KubeCon Europe 2018
02.05.2018

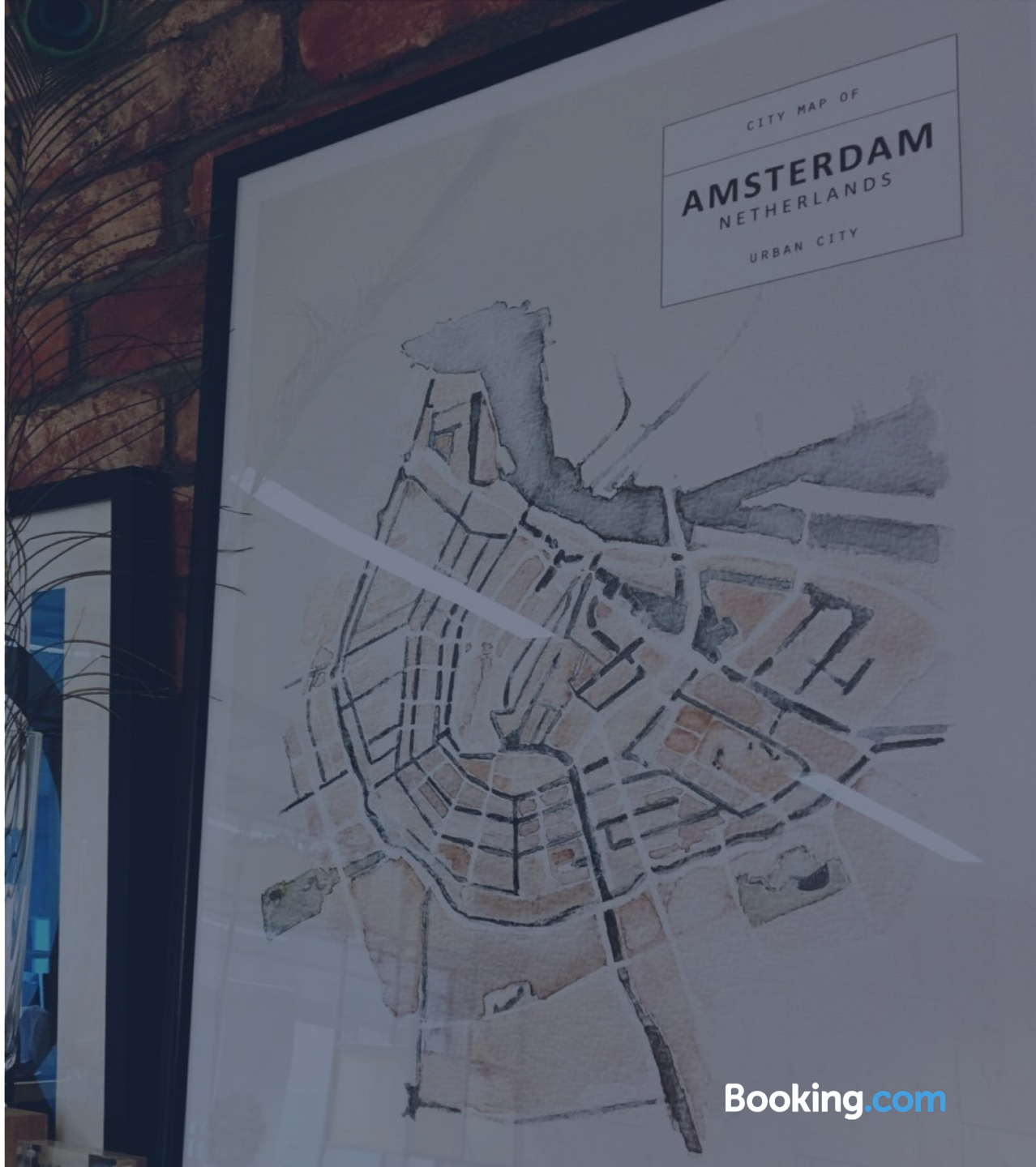
Booking.com

based in Amsterdam

28M listings

1,5M room nights per day

1700 IT staff



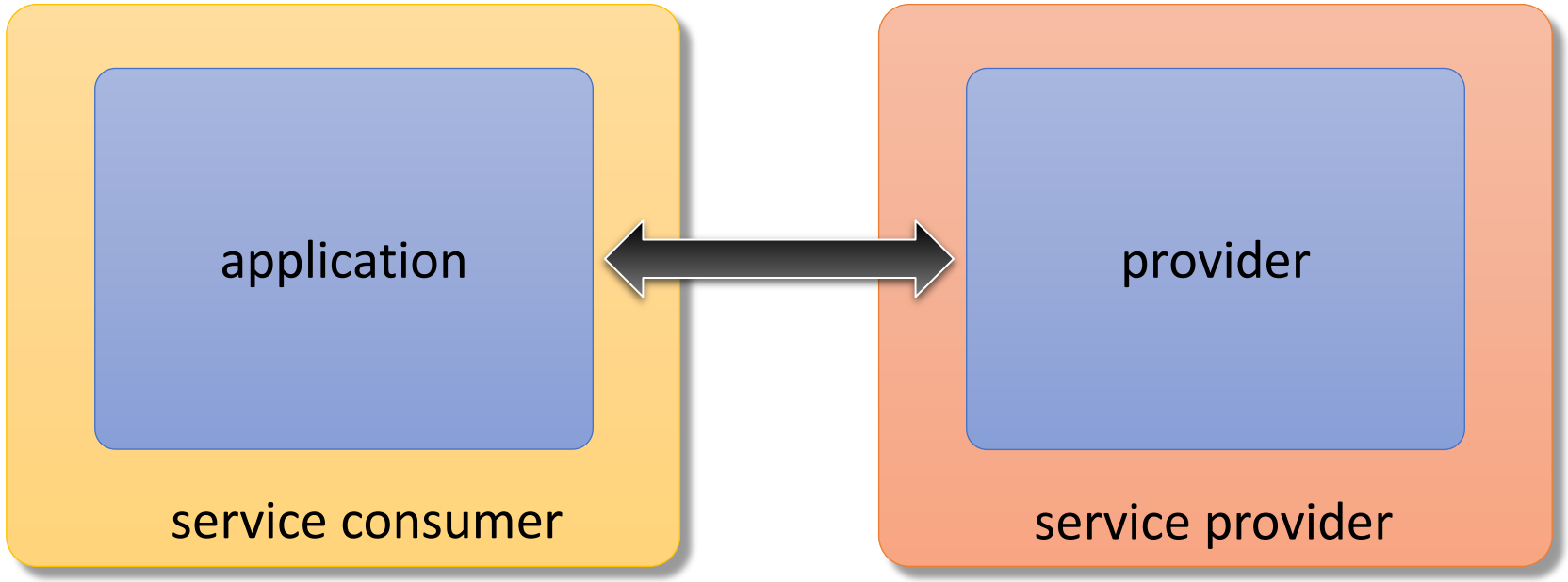
Agenda

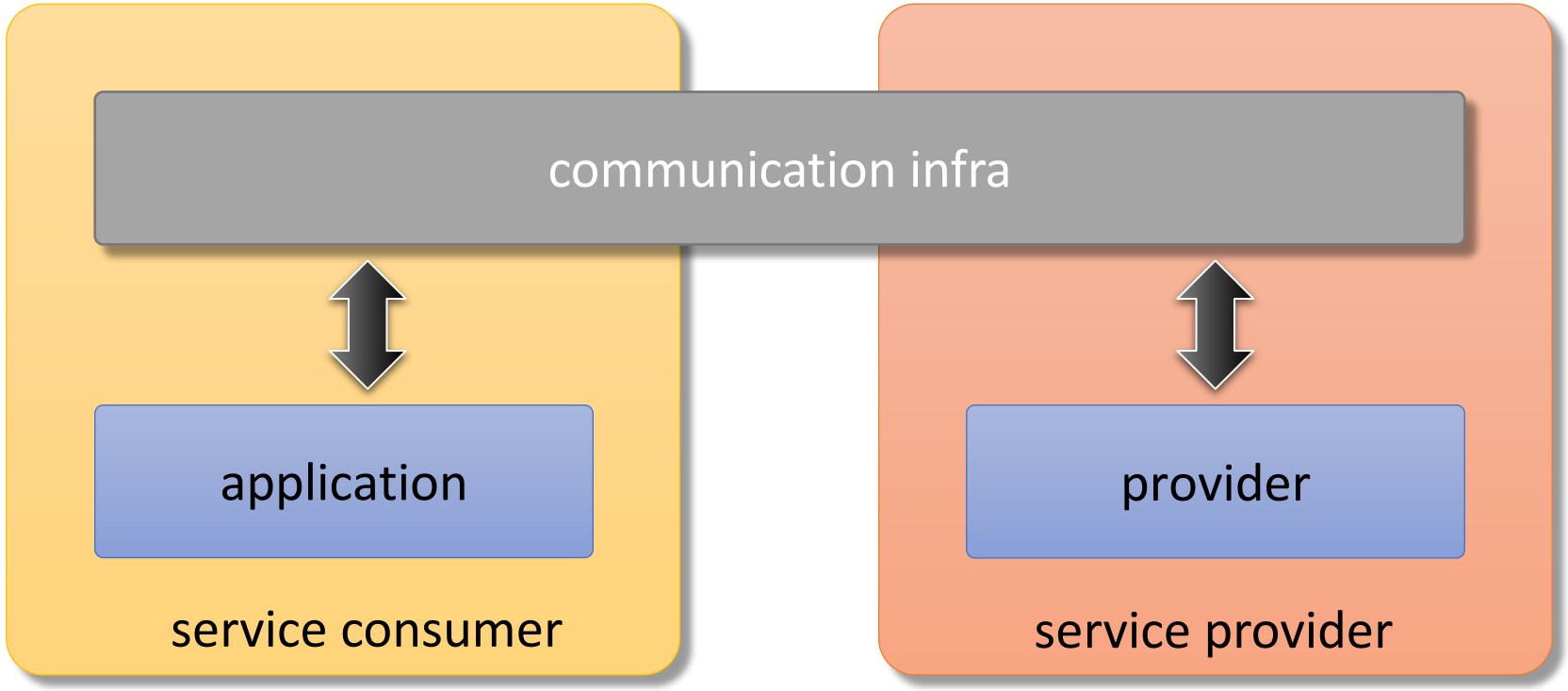
- what is service mesh?
- why did we start the project?
- our setup
- learnings
- conclusion

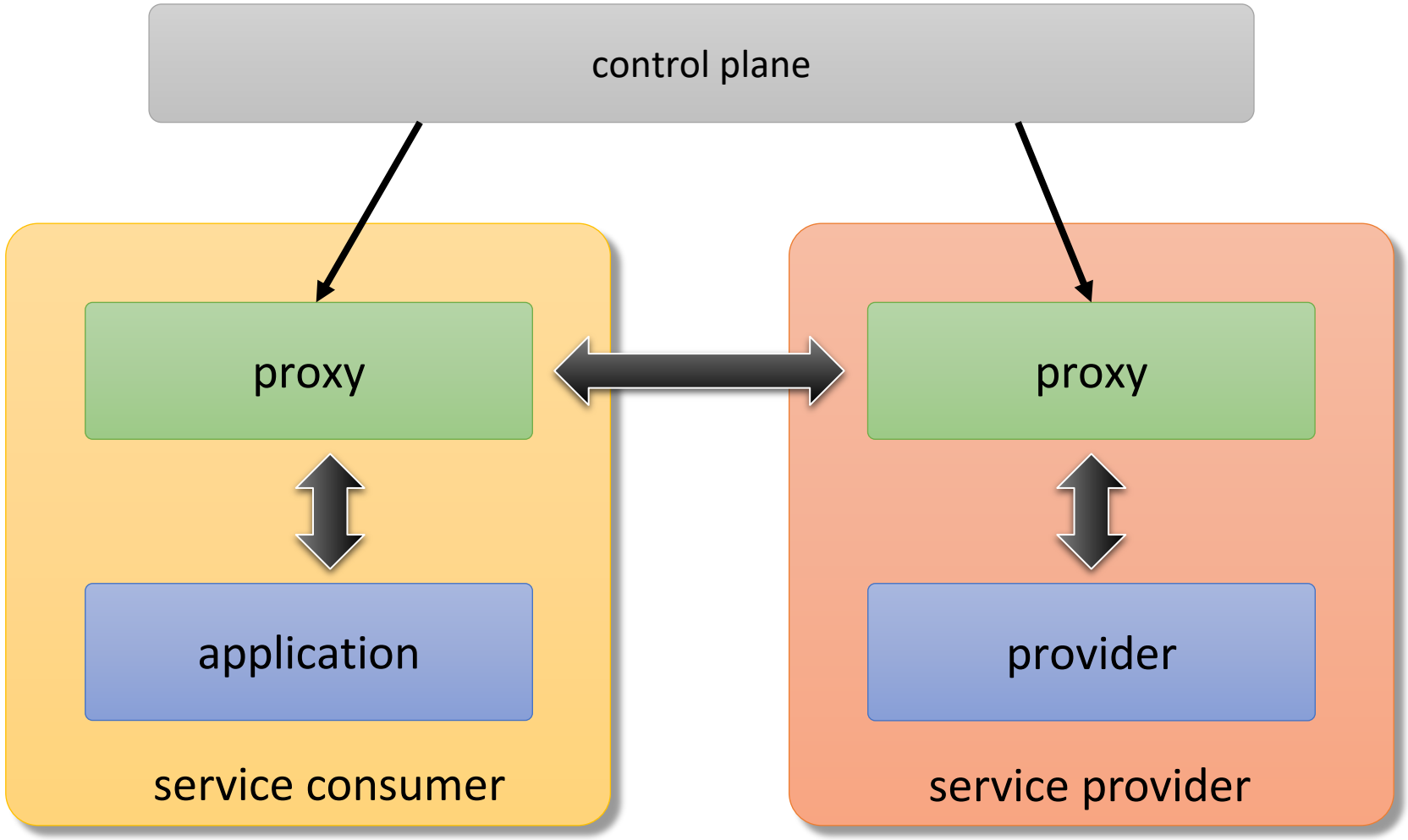
Booking.com

What is service mesh*?

* the way I understand it



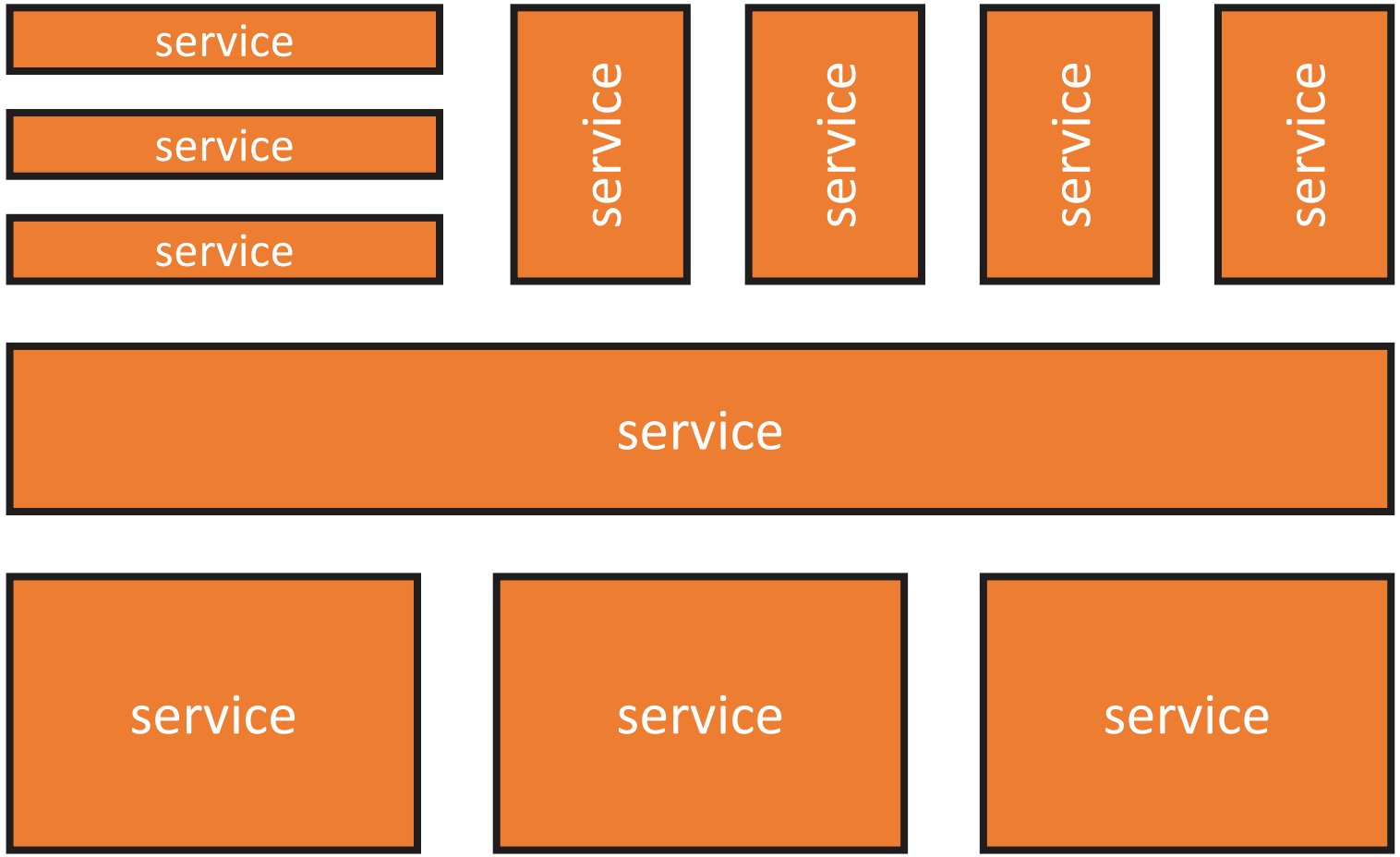

communication infra
application
provider
service consumer
service provider


control plane
proxy
application
service consumer
proxy
provider
service provider

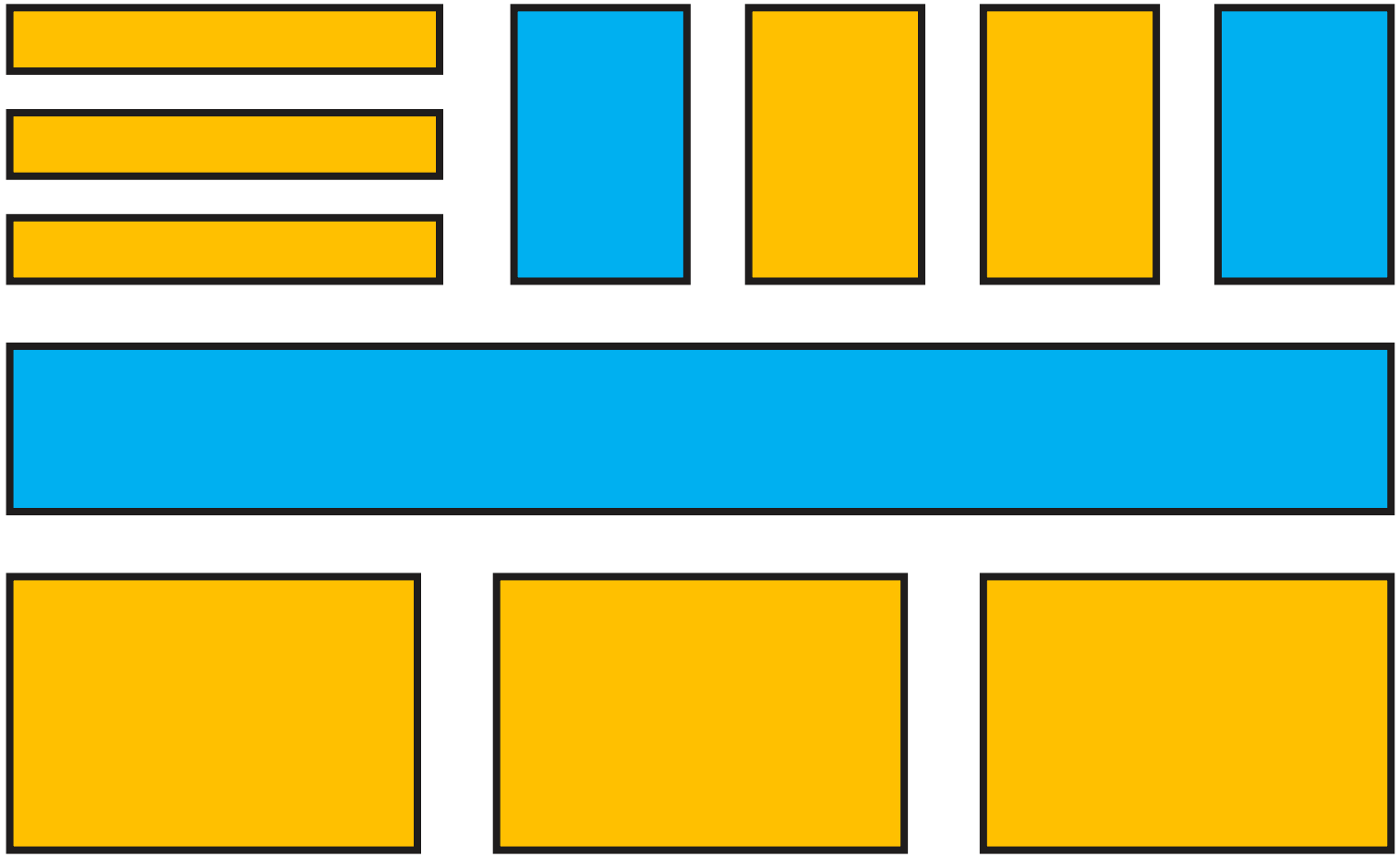
Booking.com

Why did we start service mesh project?

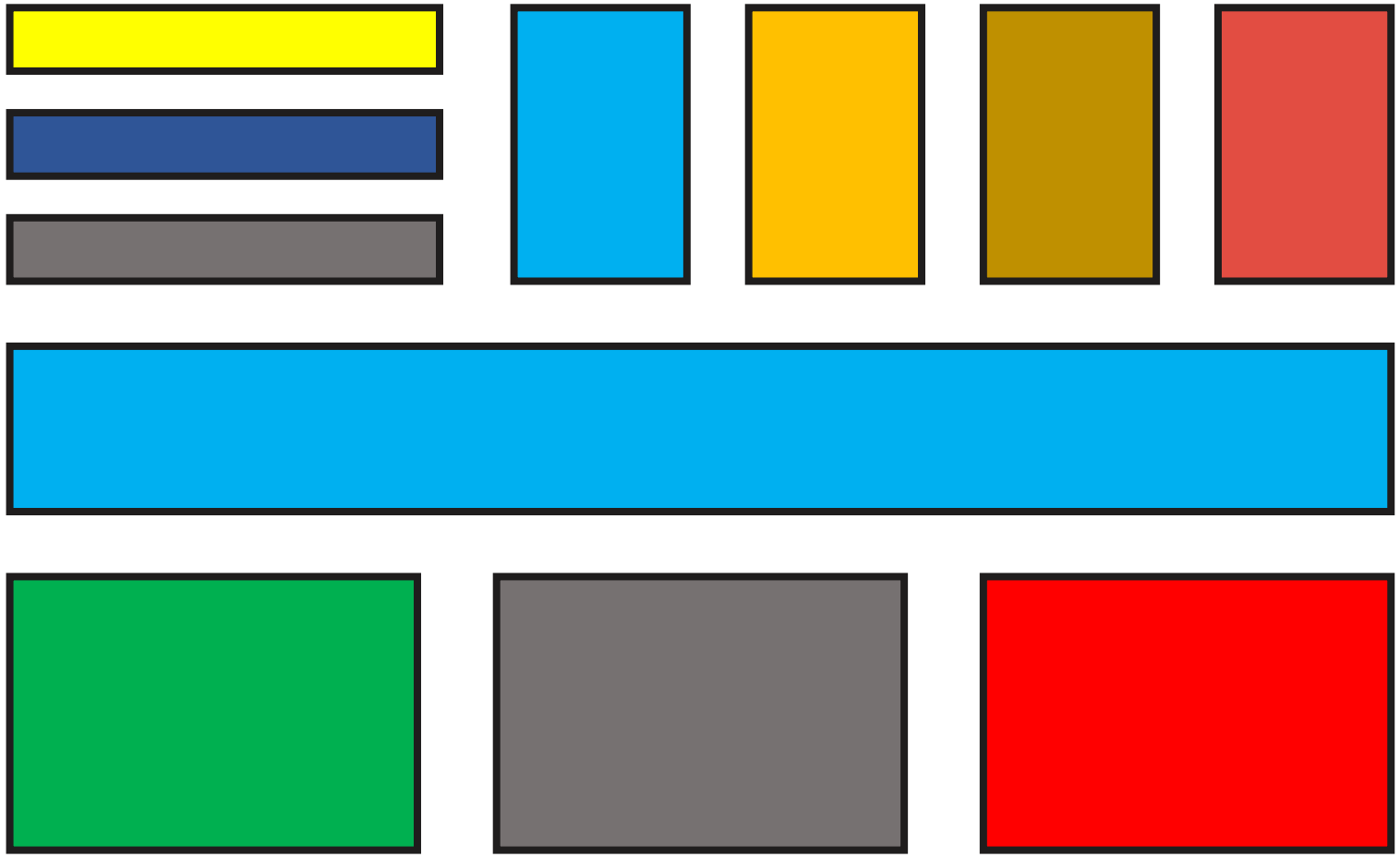

monolith


service
service
service
service
service
service
service
service
service
service
service

Booking.com



Booking.com



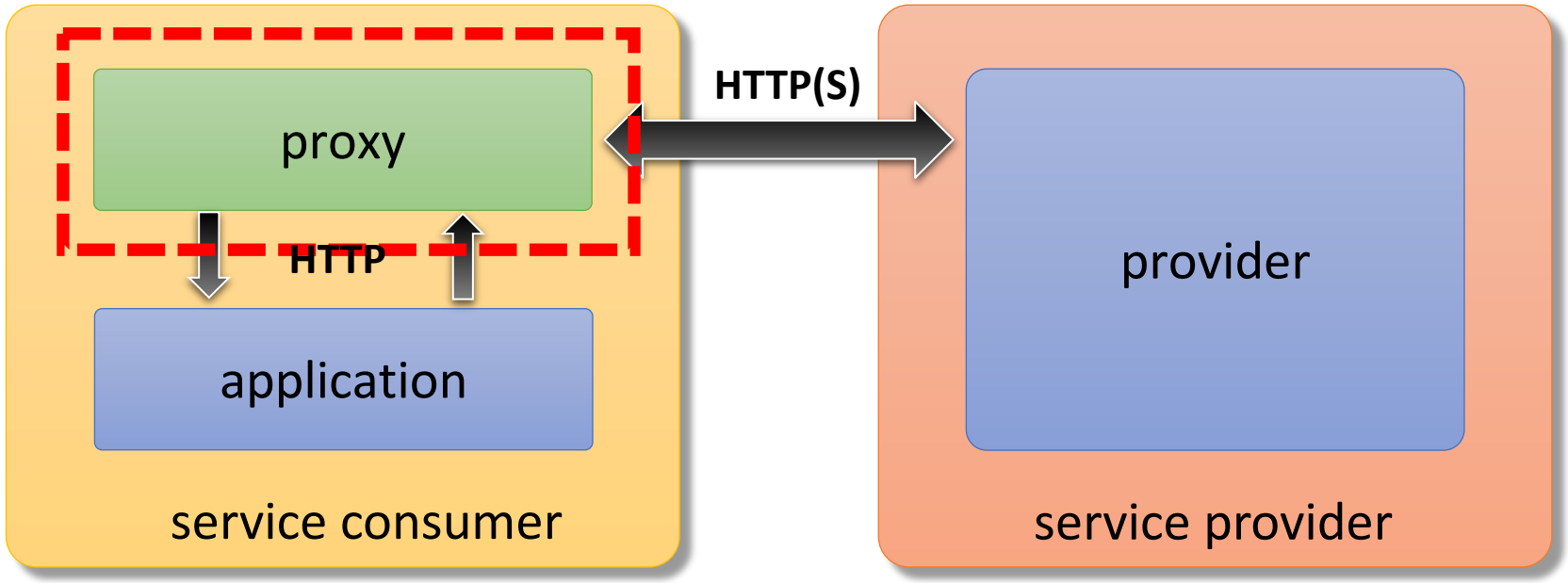
Booking.com

Consistency & Visibility

in communications

Our setup

data plane



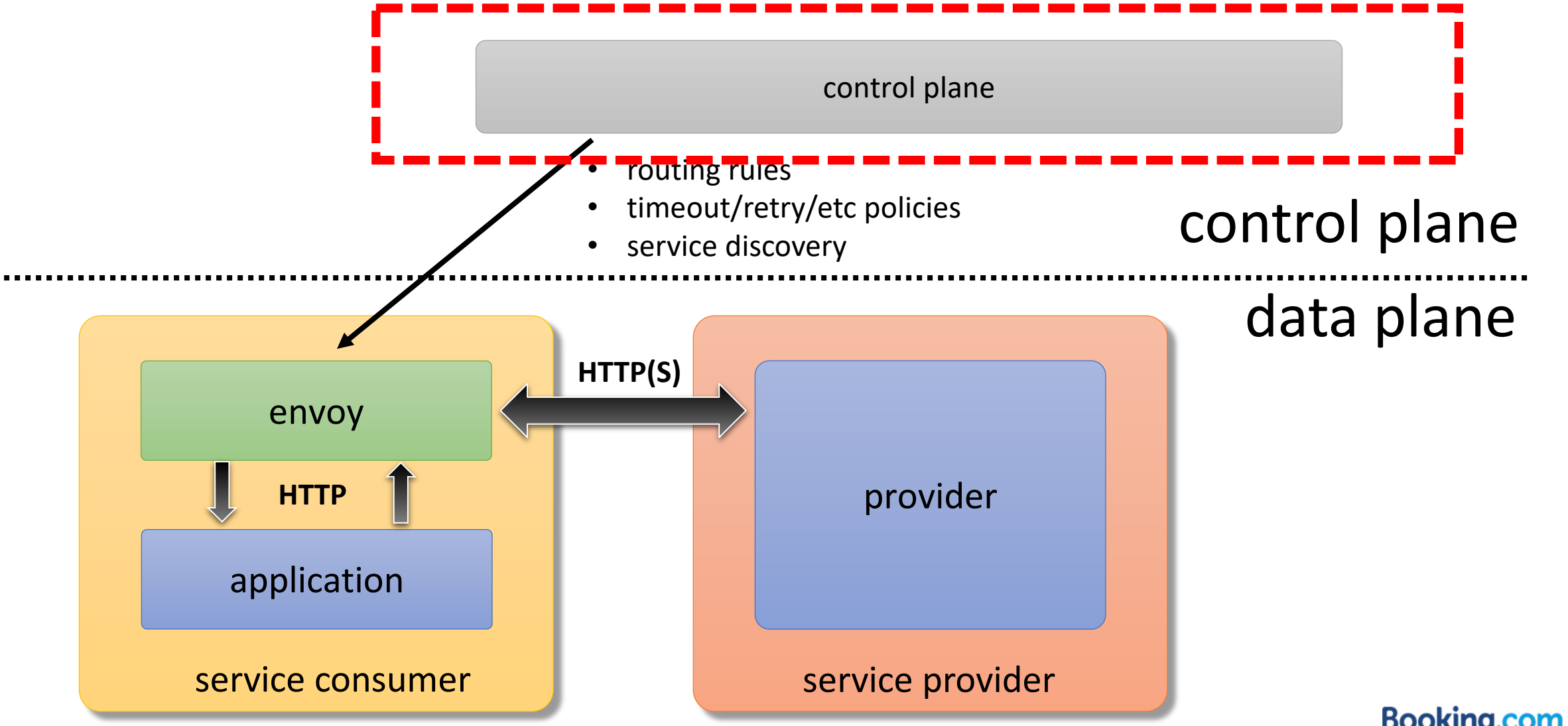
Booking.com

linkerd

envoy

- graceful restart
- TCP proxy

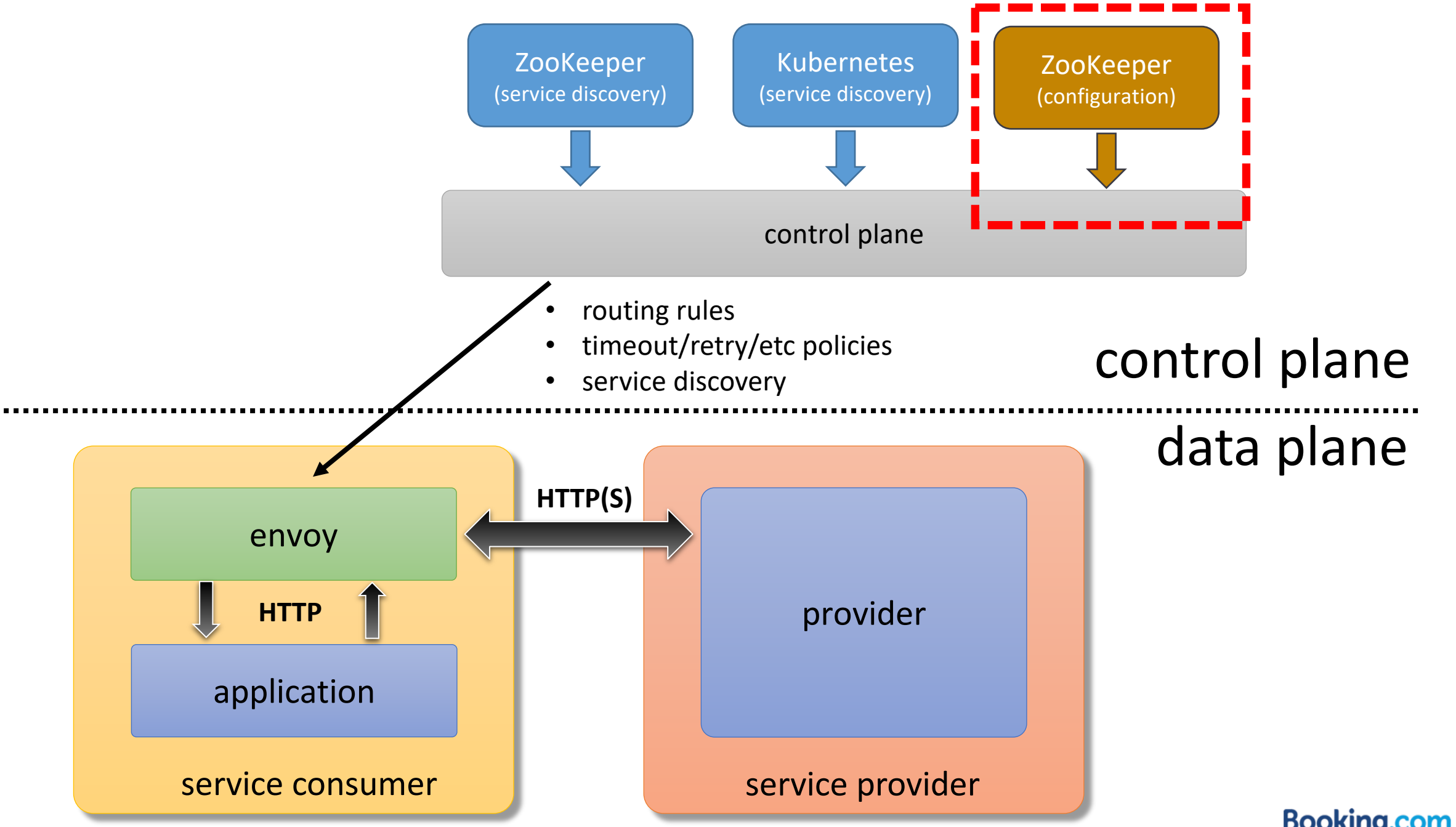
Booking.com


control plane
routing rules
timeout/retry/etc policies
service discovery
control plane
data plane
envoy
HTTP(S)
HTTP
application
provider
service consumer
service provider

Booking.com

control plane

- in-house (v1/v2 API)
  - started in August 2017; Istio is around 0.2
  - one complex system at a time
- start with bare-metal support
- minimal abstraction
  - yup, just write Envoy config (partially)
  - fine with Envoy's set of features
- self-service for service owners


ZooKeeper
(service discovery)
Kubernetes
(service discovery)
ZooKeeper
(configuration)
control plane
routing rules
timeout/retry/etc policies
service discovery
control plane
data plane
envoy
HTTP
application
HTTP(S)
provider
service consumer
service provider

configuration

- Envoy configuration is quite rich
- power vs. usability:
  - cluster specification
  - virtual host specification

```
kind: ClusterSpec
metadata:
  annotations:
    watcher_name: zookeeper.watcher
    paths: ["/pools/api/dc"]
spec:
  name: api.prod.cluster
  lb_policy: LEAST_REQUEST
  connect_timeout: 1s
  lb_subset_config:
    subset_selectors:
    - keys: ["DC"]
    - keys: ["IsLocal"]
    fallback_policy: DEFAULT_SUBSET
    default_subset:
      IsLocal: true
```

```
kind: VirtualHostSpec
spec:
  name: api.vhost
  domains: ["api.service"]
  routes:
  - match:
      prefix: /
    route:
      cluster: api.prod.cluster
      timeout: 10s
      retry_policy:
        retry_on: 5xx
        num_retries: 3
        per_try_timeout: 3s
```

```
kind: ClusterSpec
metadata:
  annotations:
    watcher_name: zookeeper.watcher
    paths: ["/pools/api/dc"]
spec:
  name: api.prod.cluster
  lb_policy: LEAST_REQUEST
  connect_timeout: 1s
  lb_subset_config:
    subset_selectors:
    - keys: ["DC"]
    - keys: ["IsLocal"]
    fallback_policy: DEFAULT_SUBSET
    default_subset:
      IsLocal: true
```

```
kind: VirtualHostSpec
spec:
  name: api.vhost
  domains: ["api.service"]
  routes:
  - match:
      prefix: /
    route:
      cluster: api.prod.cluster
      timeout: 10s
      retry_policy:
        retry_on: 5xx
        num_retries: 3
        per_try_timeout: 3s
```

```
kind: ClusterSpec
metadata:
  annotations:
    watcher_name: zookeeper.watcher
    paths: ["/pools/api/dc"]
spec:
  name: api.prod.cluster
  lb_policy: LEAST_REQUEST
  connect_timeout: 1s
  lb_subset_config:
    subset_selectors:
    - keys: ["DC"]
    - keys: ["IsLocal"]
    fallback_policy: DEFAULT_SUBSET
    default_subset:
      IsLocal: true
```

```
kind: VirtualHostSpec
spec:
  name: api.vhost
  domains: ["api.service"]
  routes:
  - match:
      prefix: /
    route:
      cluster: api.prod.cluster
      timeout: 10s
      retry_policy:
        retry_on: 5xx
        num_retries: 3
        per_try_timeout: 3s
```

```
kind: ClusterSpec
metadata:
  annotations:
    watcher_name: zookeeper.watcher
    paths: ["/pools/api/dc"]
spec:
  name: api.prod.cluster
  lb_policy: LEAST_REQUEST
  connect_timeout: 1s
  lb_subset_config:
    subset_selectors:
    - keys: ["DC"]
    - keys: ["IsLocal"]
    fallback_policy: DEFAULT_SUBSET
    default_subset:
      IsLocal: true
```

```
kind: VirtualHostSpec
spec:
  name: api.vhost
  domains: ["api.service"]
  routes:
  - match:
      prefix: /
    route:
      cluster: api.prod.cluster
      timeout: 10s
      retry_policy:
        retry_on: 5xx
        num_retries: 3
        per_try_timeout: 3s
```

```
kind: ClusterSpec
metadata:
  annotations:
    watcher_name: zookeeper.watcher
    paths: ["/pools/api/dc"]
spec:
  name: api.prod.cluster
  lb_policy: LEAST_REQUEST
  connect_timeout: 1s
  lb_subset_config:
    subset_selectors:
    - keys: ["DC"]
    - keys: ["IsLocal"]
    fallback_policy: DEFAULT_SUBSET
    default_subset:
      IsLocal: true
```

envoy cluster spec

```
kind: VirtualHostSpec
spec:
  name: api.vhost
  domains: ["api.service"]
  routes:
  - match:
      prefix: /
    route:
      cluster: api.prod.cluster
      timeout: 10s
      retry_policy:
        retry_on: 5xx
        num_retries: 3
        per_try_timeout: 3s
```

envoy virtual host spec

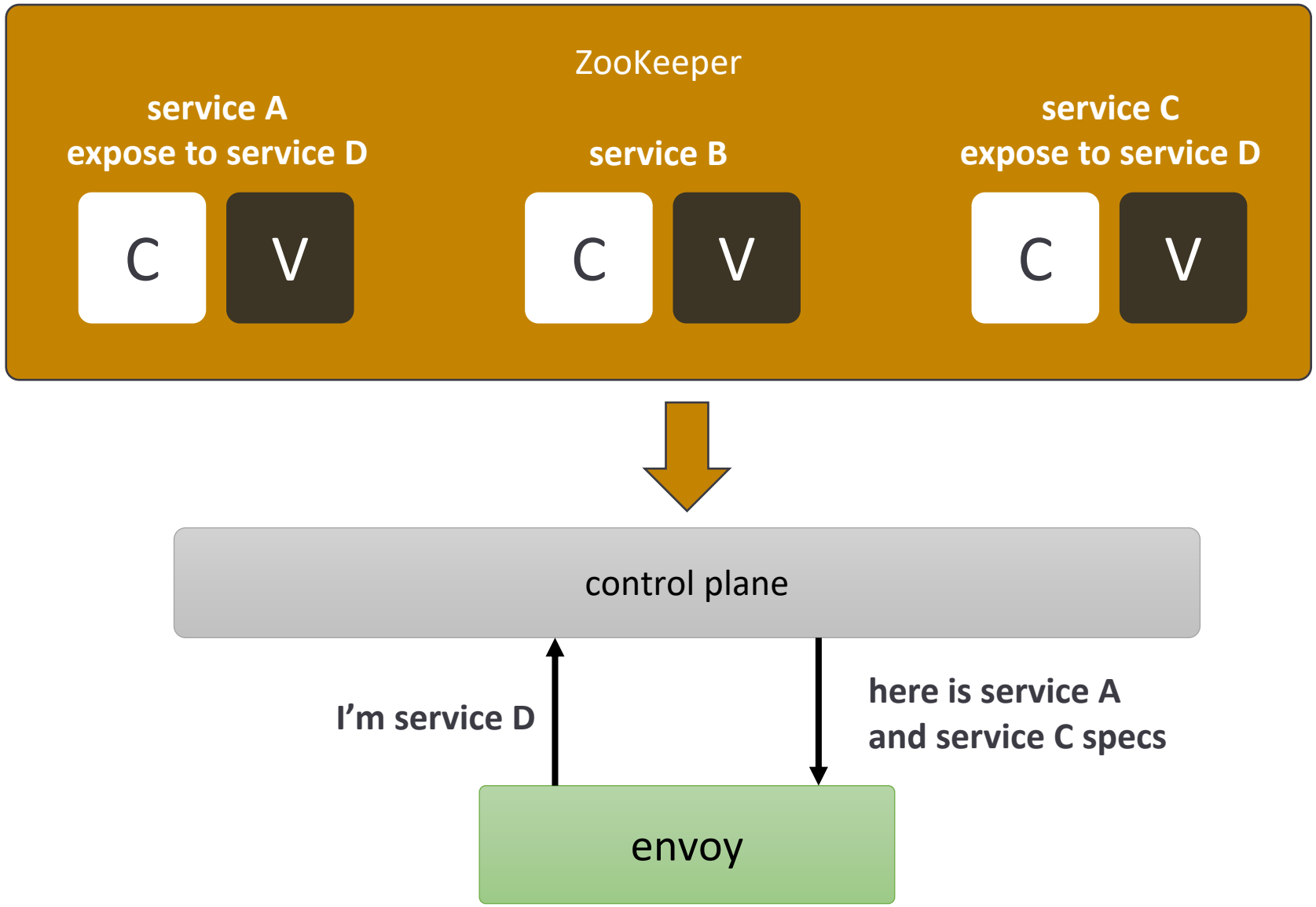
```
kind: ClusterSpec
metadata:
  annotations:
    watcher_name: zookeeper.watcher
    paths: ["/pools/api/dc"]
spec:
  name: api.pr
  lb_policy: I
  connect_time
  lb_subset_co
    subset_sel
    - keys: ["
    - keys: ["
    fallback_p
    default_subset:
      IsLocal: true
```

envoy cluster spec

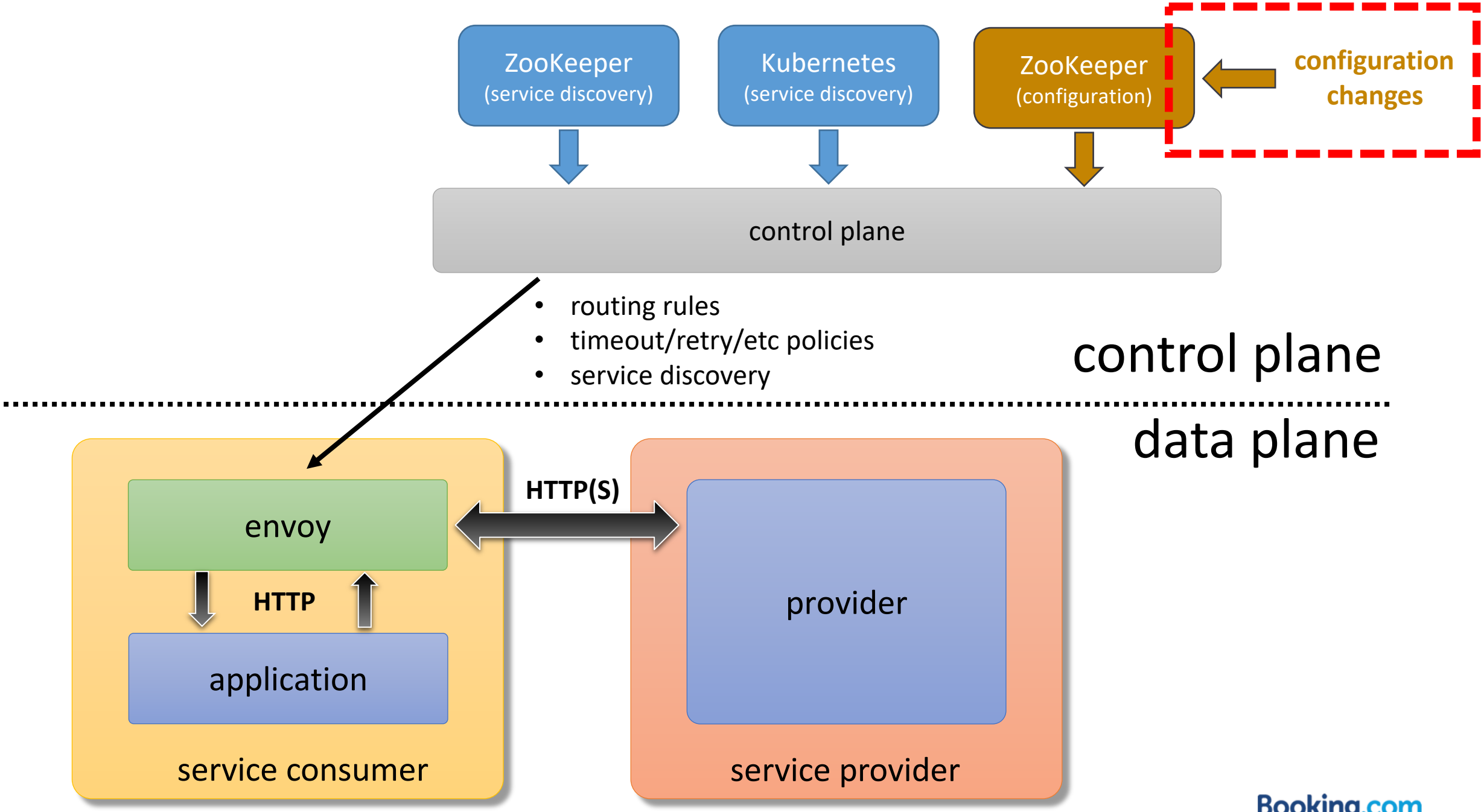
```
kind: VirtualHostSpec
spec:
  name: api.vhost
  domains: ["api.service"]
  routes:
                                rod.cluster
                                x
                                  3
                  eout: 3s
```

envoy virtual host spec

bootstrap wizard


ZooKeeper
service A
expose to service D
C
V
service B
C
V
service C
expose to service D
C
V
control plane
I'm service D
here is service A
and service C specs
envoy

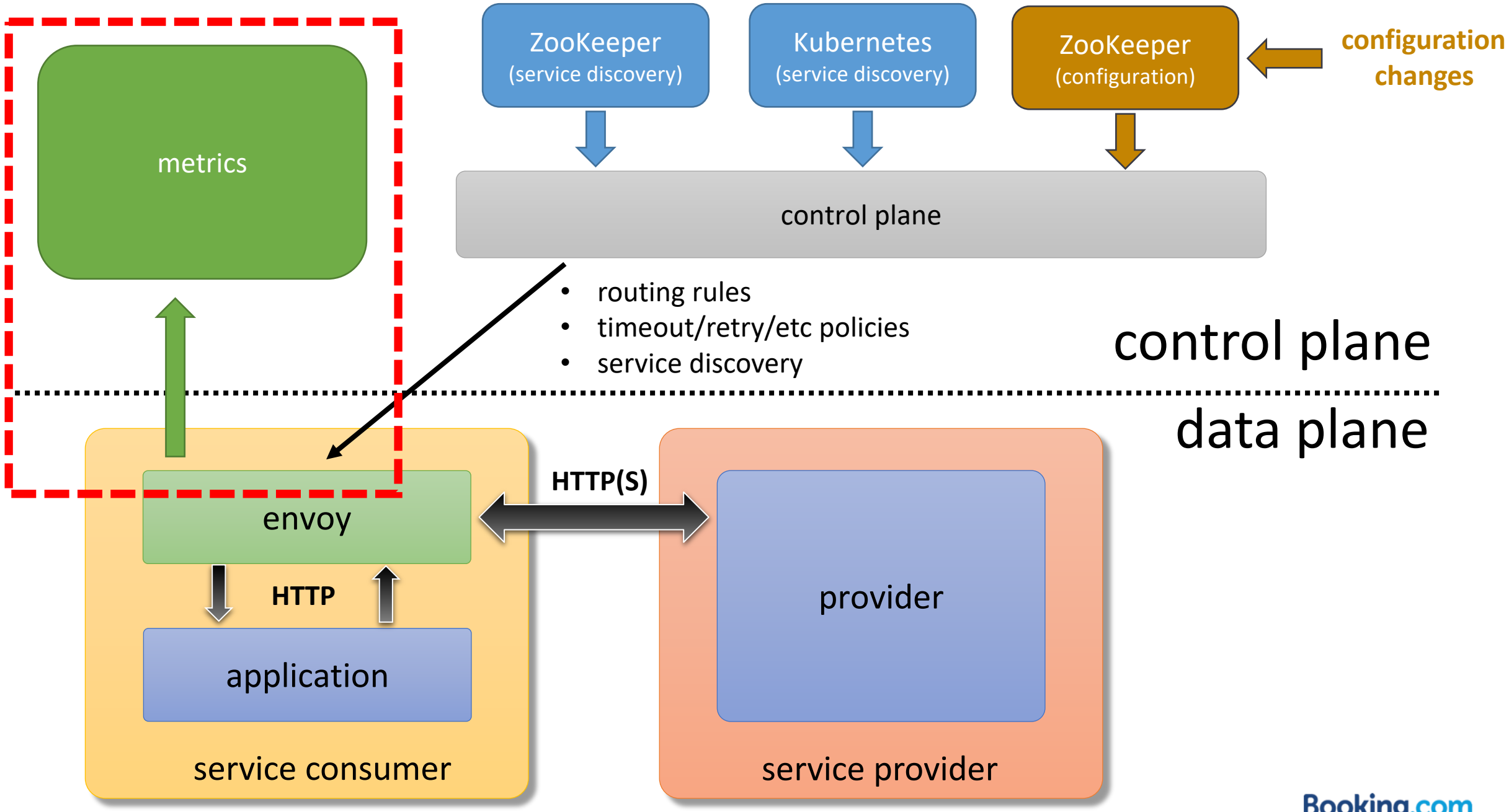
Booking.com


ZooKeeper
(service discovery)
Kubernetes
(service discovery)
ZooKeeper
(configuration)
configuration changes
control plane
routing rules
timeout/retry/etc policies
service discovery
control plane
data plane
envoy
HTTP
application
HTTP(S)
provider
service consumer
service provider

Booking.com

Deploying changes

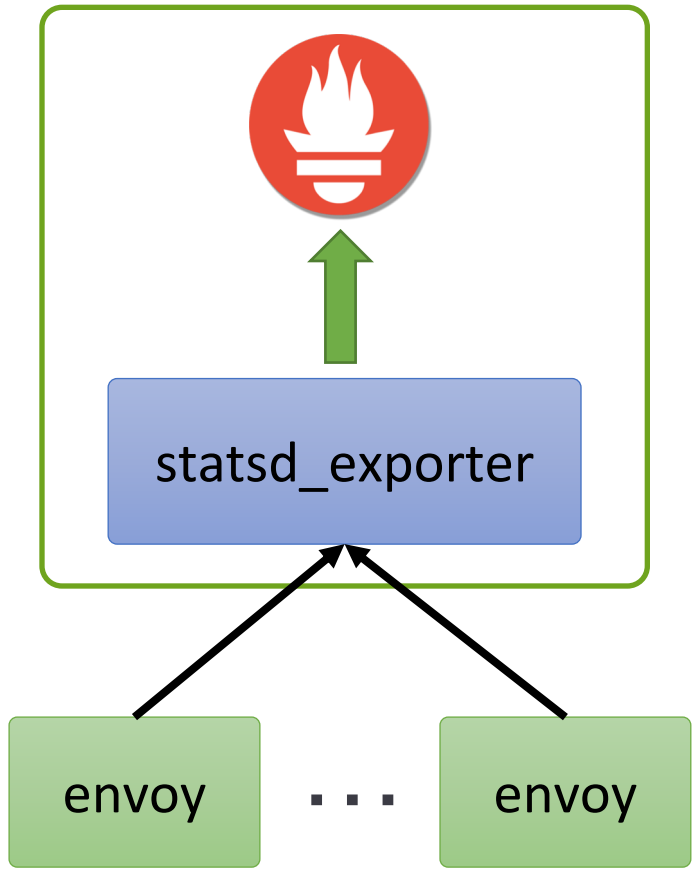
- integration with existent infrastructure
  - git-deploy
- vanguard (canary) deployment
- syntax and semantic validation
- fast rollback


ZooKeeper
(service discovery)
Kubernetes
(service discovery)
ZooKeeper
(configuration)
configuration changes
metrics
control plane
routing rules
timeout/retry/etc policies
service discovery
control plane
data plane
envoy
HTTP(S)
HTTP
application
provider
service consumer
service provider

Booking.com

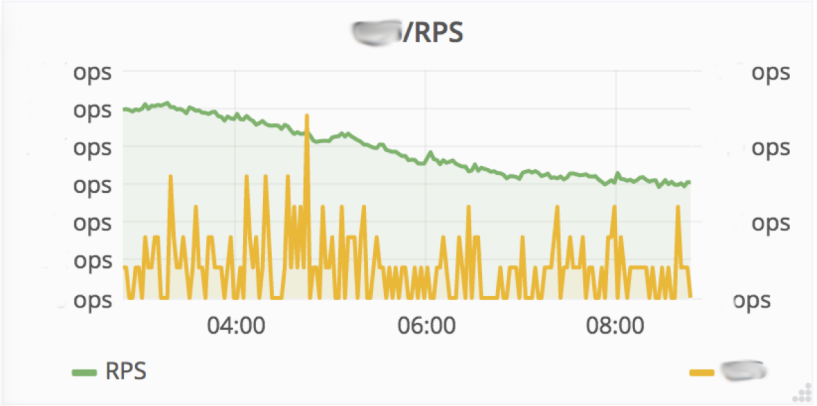
Monitoring

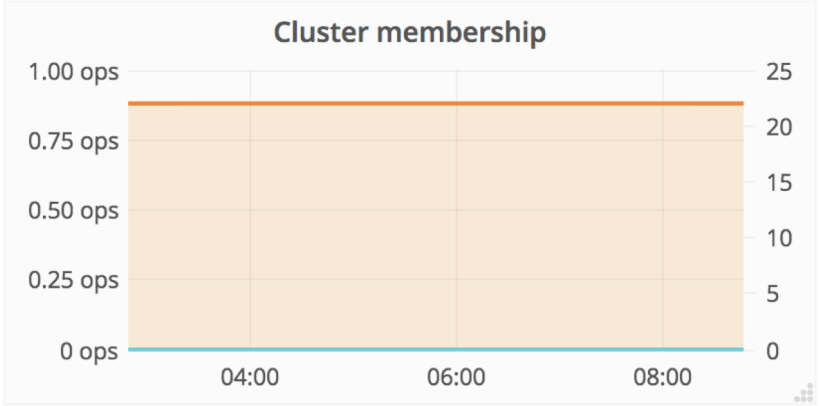
- approach #1 – graphite
  - excellent in-house facilities
  - aggregations is too heavy at scale
- approach #2 – prometheus
  - statsd support
  - statsd_exporter
  - still iterating
- standard dashboard

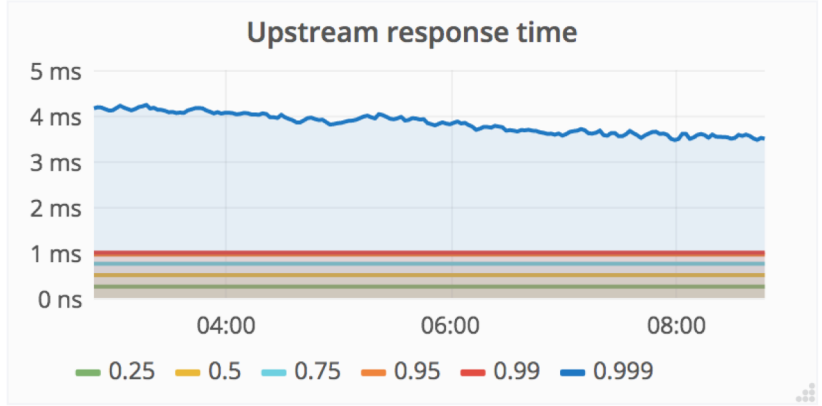


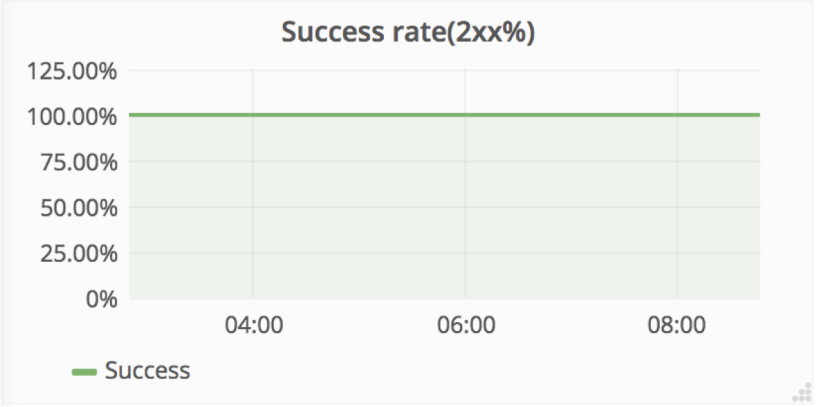
DataSource [redacted] To Service [redacted].prod.cluster

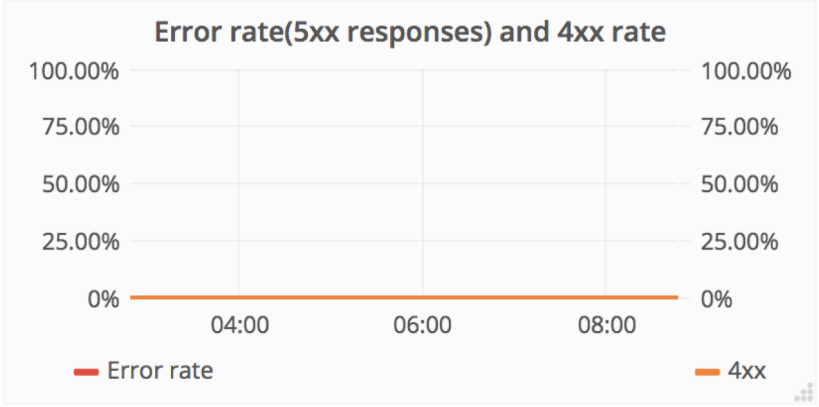
Egress to [redacted]prod.cluster

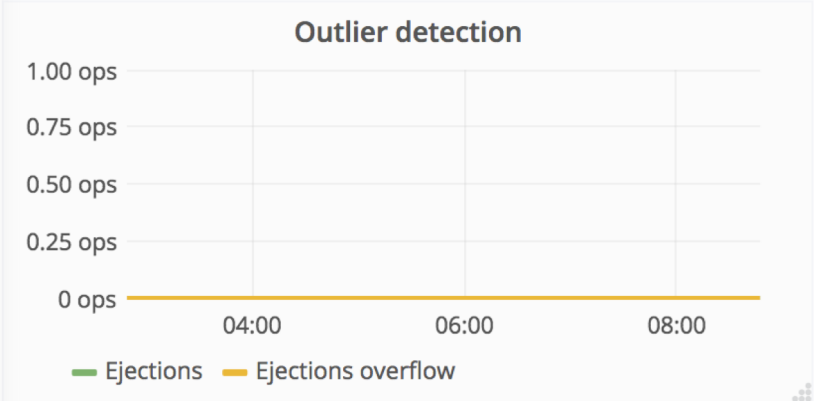


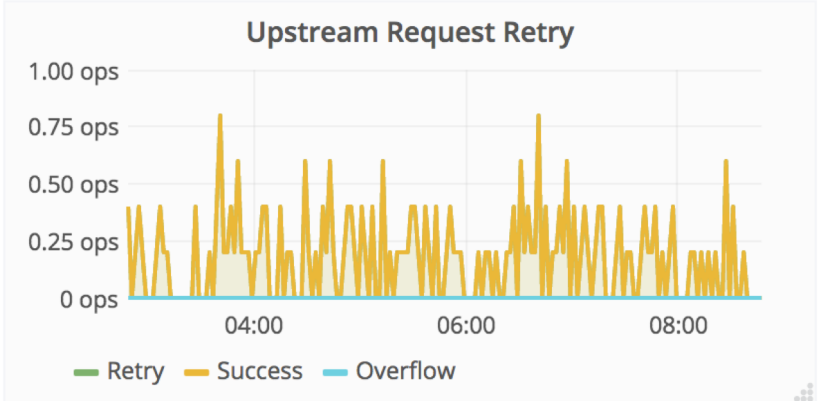


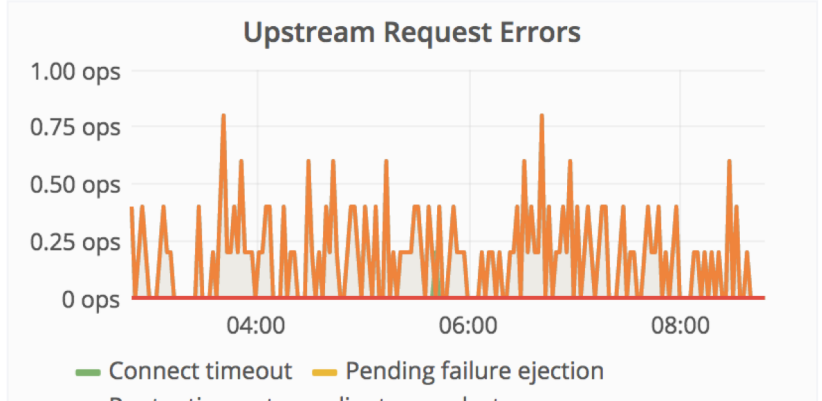


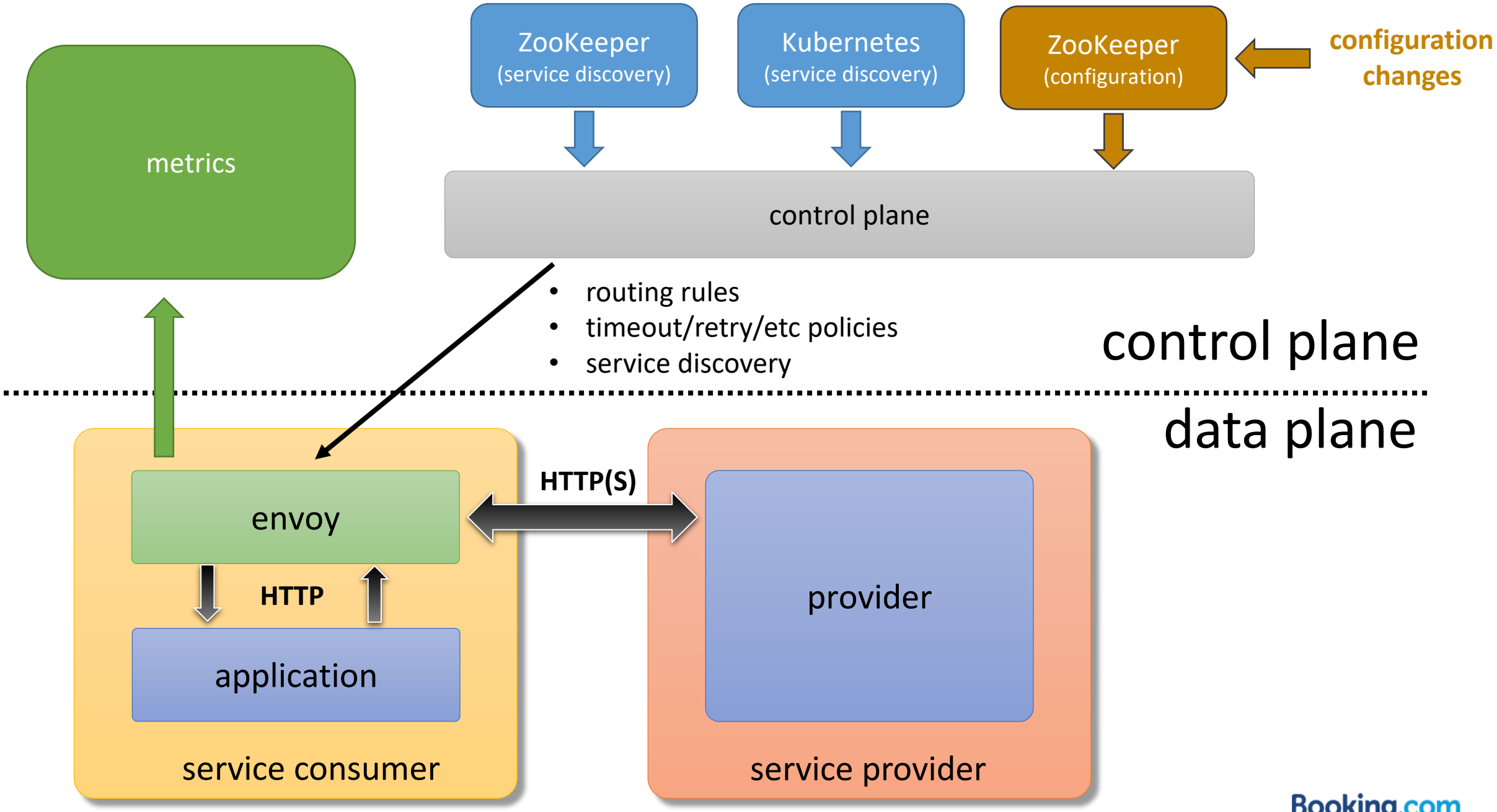










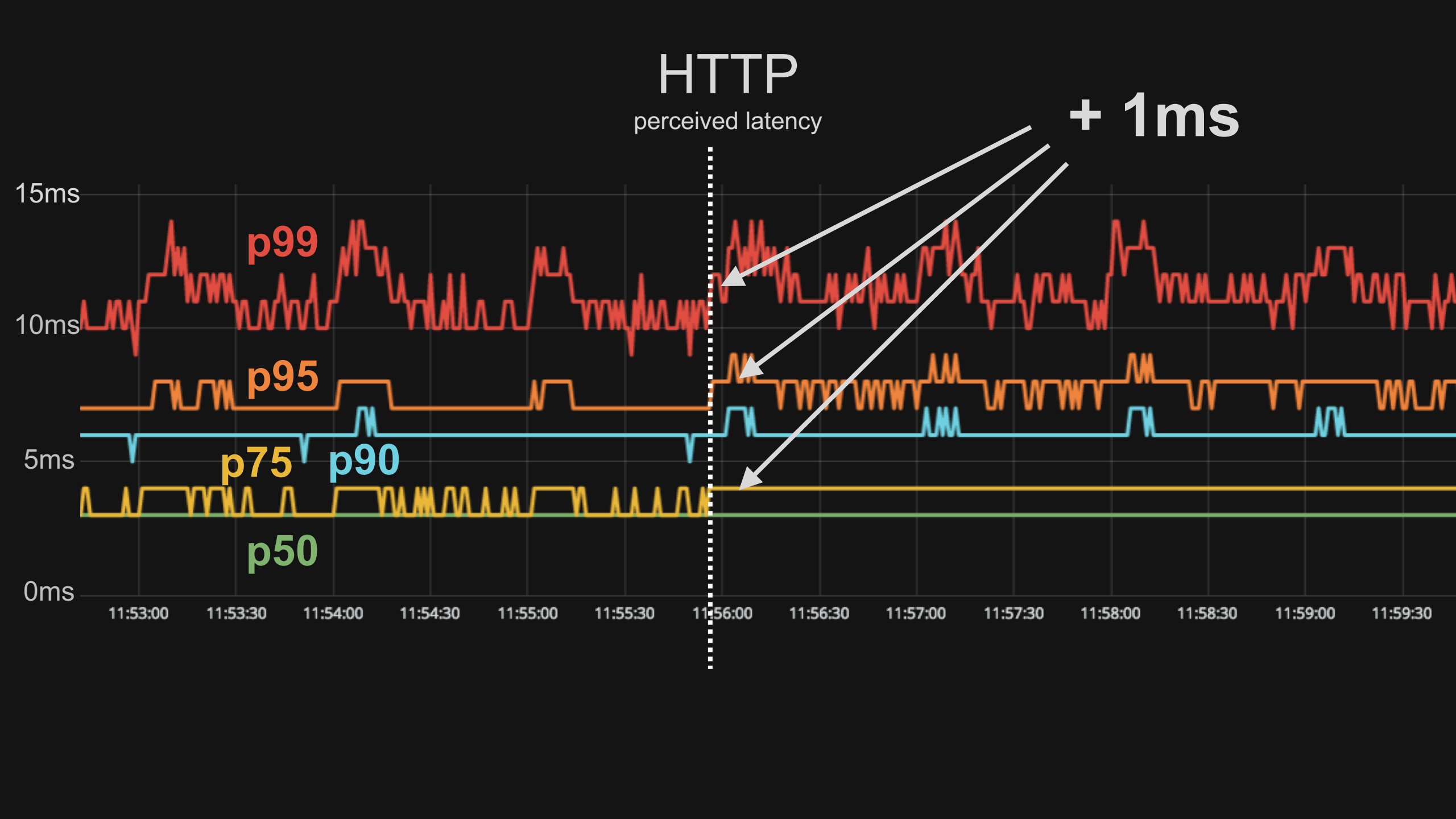

ZooKeeper
(service discovery)
Kubernetes
(service discovery)
ZooKeeper
(configuration)
configuration changes
metrics
control plane
routing rules
timeout/retry/etc policies
service discovery
control plane
data plane
envoy
HTTP(S)
HTTP
application
provider
service consumer
service provider

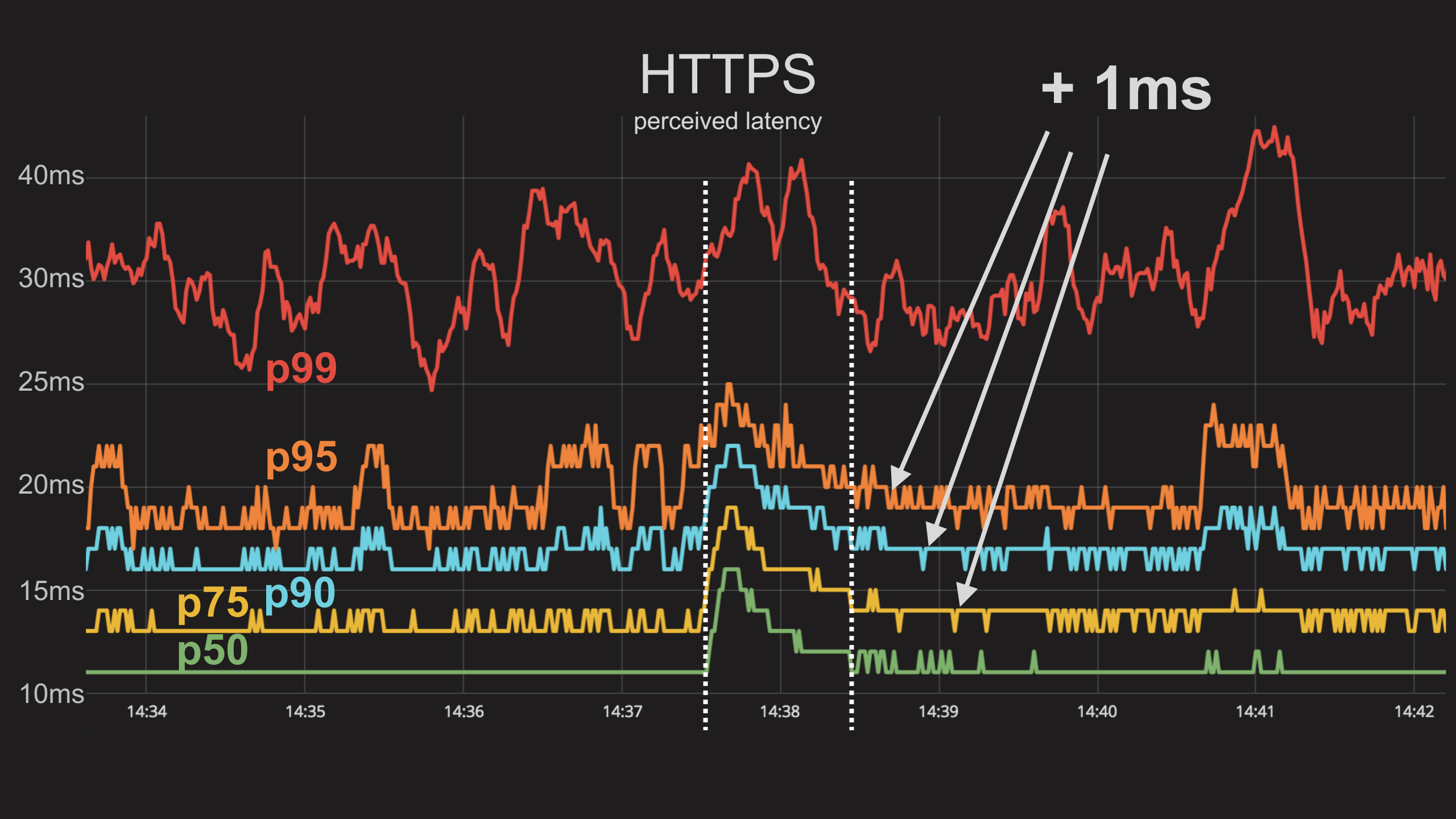
Production

Booking.com

Some numbers

- ~ 6 months in production
- ~ 30 projects
- ~ 10K servers
- hundreds of thousands RPS
- overheads…


HTTP
perceived latency
+ 1ms
15ms
10ms
5ms
0ms
p99
p95
p75
p90
p50
11:53:00
11:53:30
11:54:00
11:54:30
11:55:00
11:55:30
11:56:00
11:56:30
11:57:00
11:57:30
11:58:00
11:58:30
11:59:00
11:59:30


HTTPS
perceived latency
+ 1ms
40ms
30ms
25ms
20ms
15ms
10ms
p99
p95
p90
p75
p50
14:34
14:35
14:36
14:37
14:38
14:39
14:40
14:41
14:42

Some findings

- graceful restart works, but not always!
  - (major) Envoy upgrades may not be graceful
- ”x-envoy-retry-on = connect-failure” – too few
  “x-envoy-retry-on = 5xx” – too much
  - gateway-error – HTTP 502, 503, 504
- absence of TCP keepalive => stale connections
  - coming in 1.7
  - idle_timeout in TCP proxy in 1.6
- cluster_name (1.5) -> cluster_names (1.6)

Conclusion

Booking.com

service mesh* is
a building block on the path to SOA
technically & organizationally

* the way I understand it

Booking.com

Andrei Vereha
Envoy production stories
Friday, May 4th 14:00
Envoy Deep Dive

Booking.com


Thank you!
Ivan Kruglov
ivan.kruglov@booking.com
Booking.com