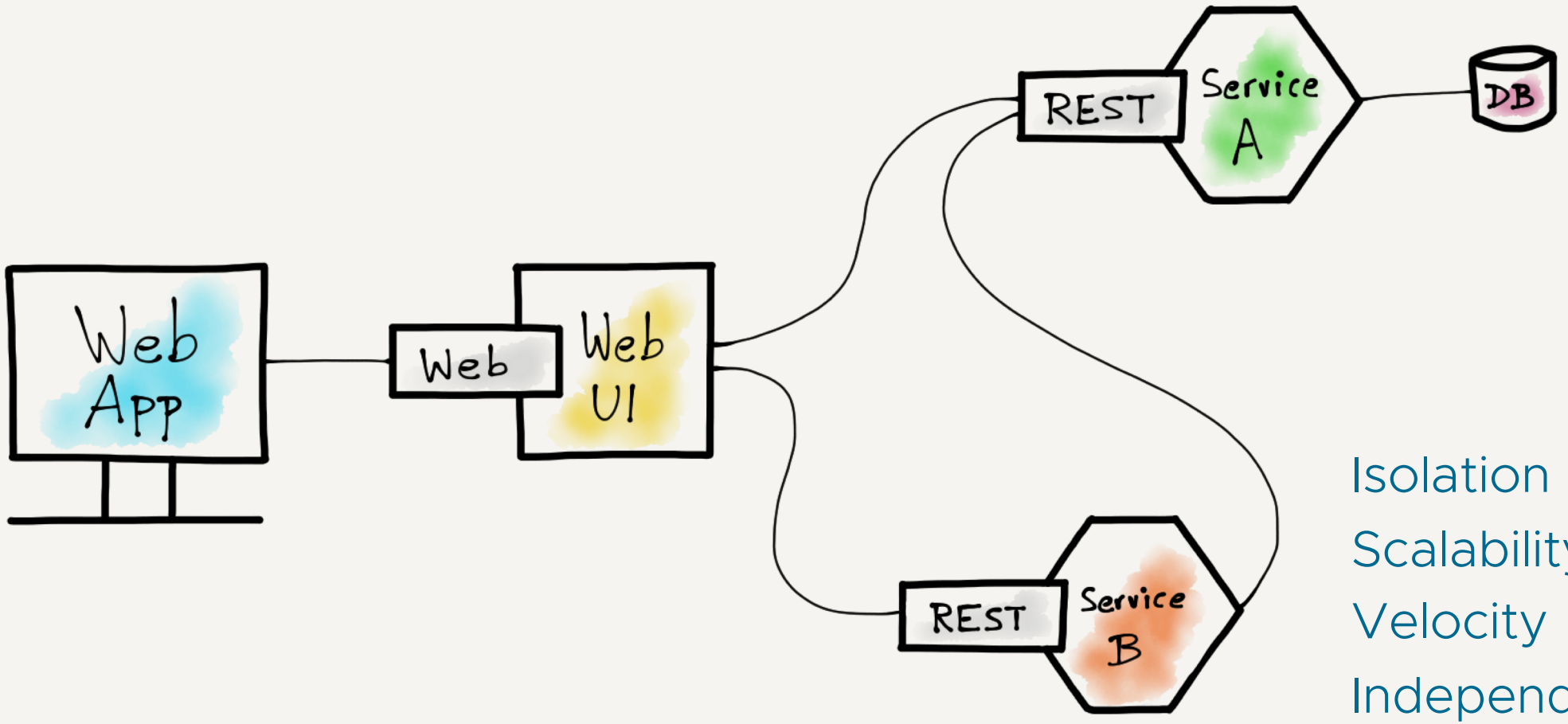
Seamless Cloud-Native Apps with gRPC-Web and Istio

gRPC Conf 2019

Venil Noronha

Open Source Technology Center, VMware

The REST APIs Approach



Isolation

Scalability

Velocity

Independence

Drawbacks of REST APIs

```
1  GET /api/users/123
2
3  {
4      "id": 123,
5      "name": "John Doe",
6      "weight": 180.56
7  }
```

Type Safety

Compatibility

Performance

Contract Definition

Protobufs and gRPC

```
1  service UserService {
2      rpc FindUser (FindUserRequest) returns (FindUserResponse);
3  }
4
5  message FindUserRequest {
6      uint64 id = 1;
7  }
8
9  message FindUserResponse {
10     uint64 id = 1;
11     string name = 2;
12     double weight = 3;
13 }
```

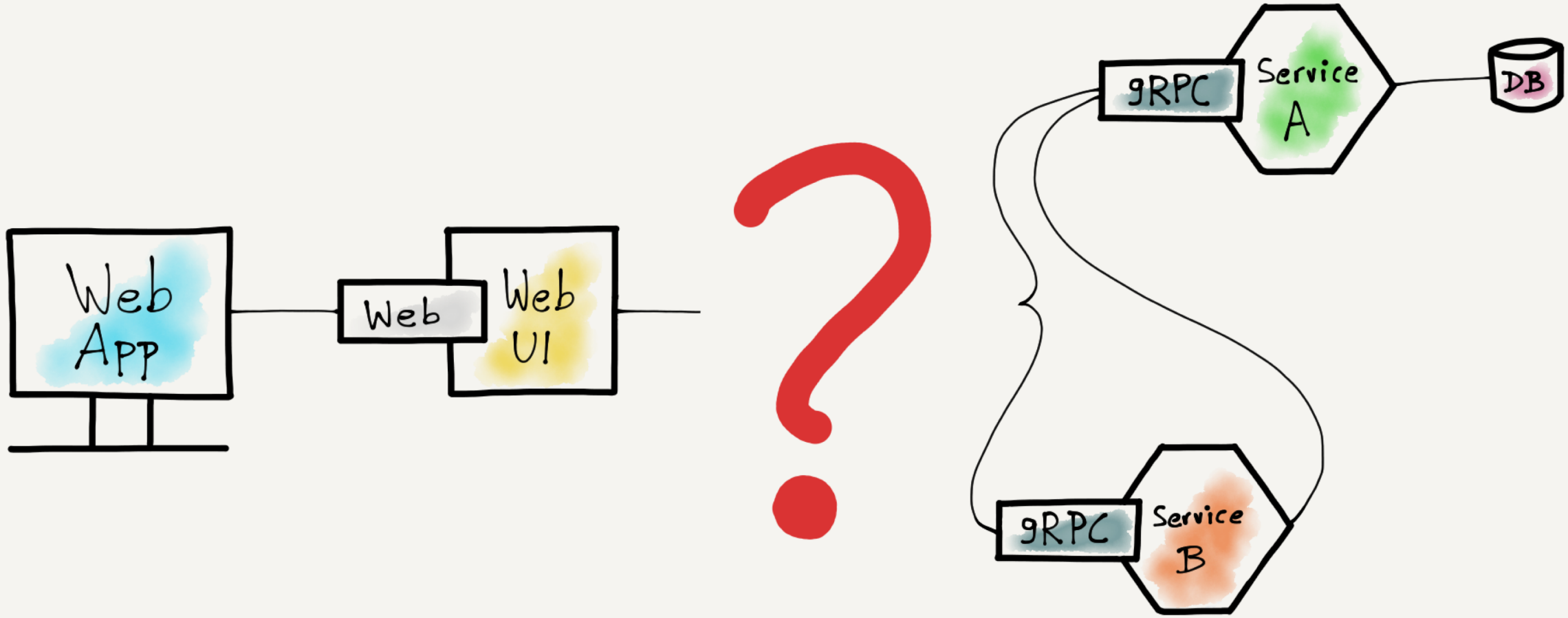
Type Safety

Compatibility

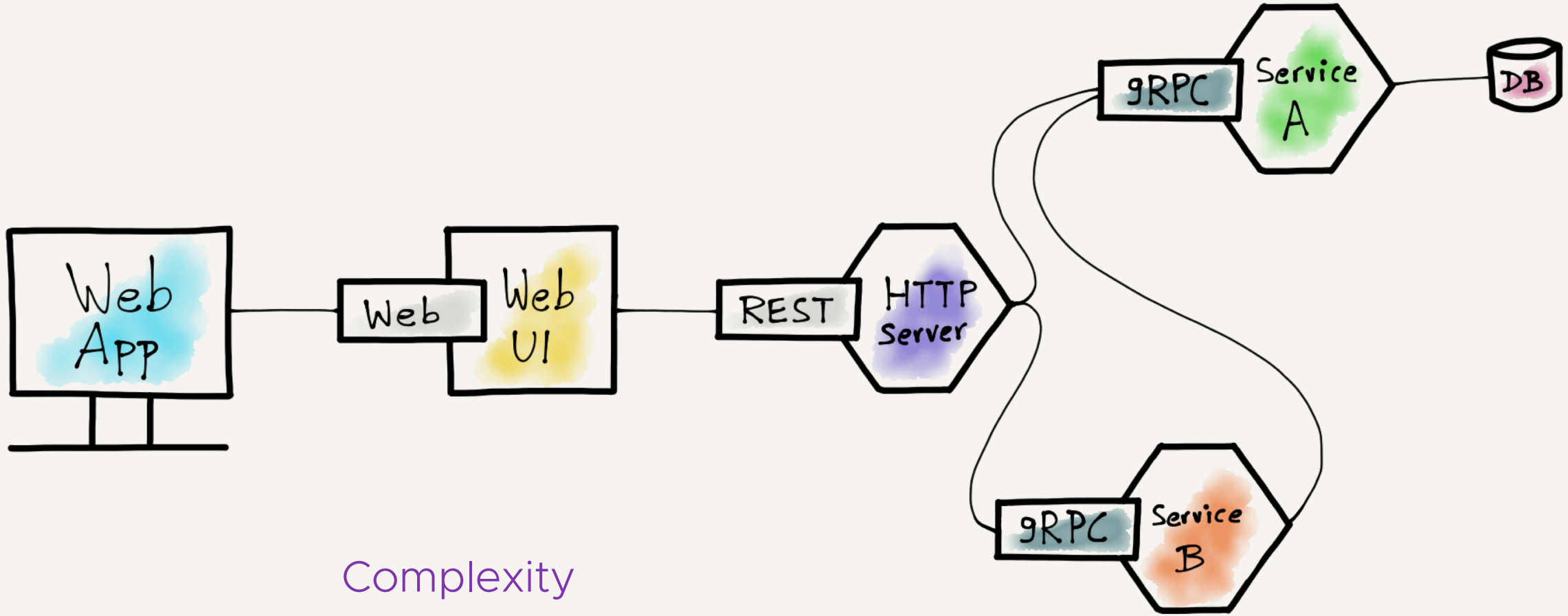
Performance

Contract Definition

Challenges with gRPC and the Web



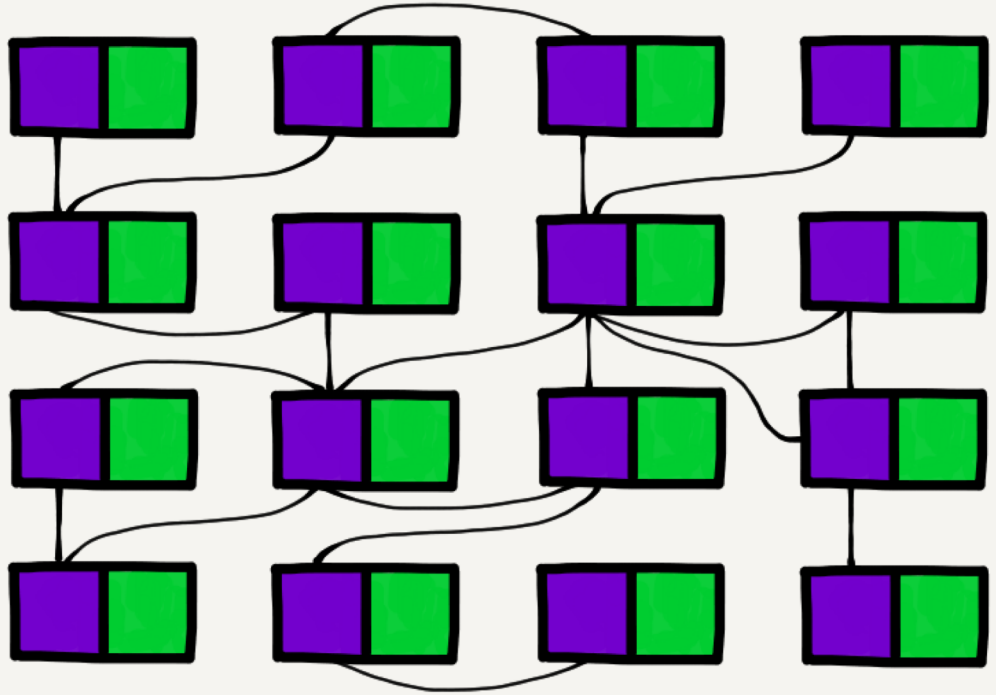
The gRPC + HTTP Server Pattern



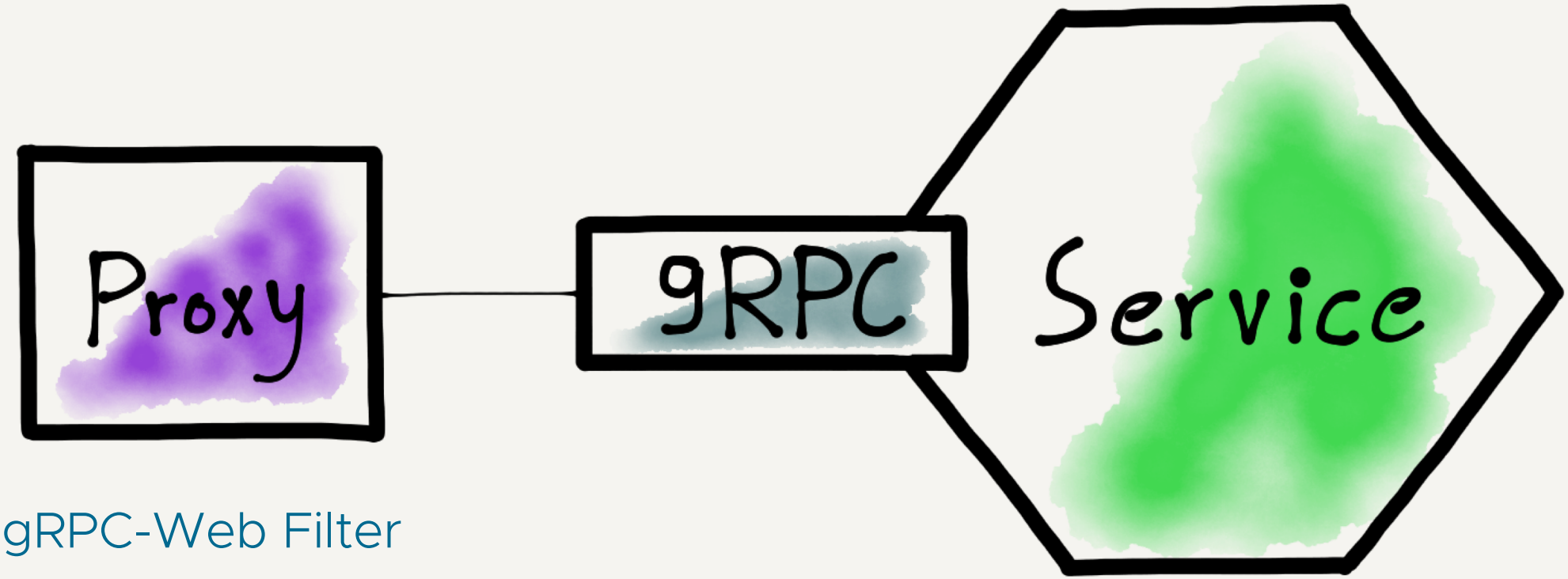
Complexity

Service Mesh Overview

- Dedicated infrastructure layer
- Handles service-to-service communication
- Manages complex topology of services
- Array of lightweight network proxies
- Deployed alongside application code
- Doesn’t need the application to be aware

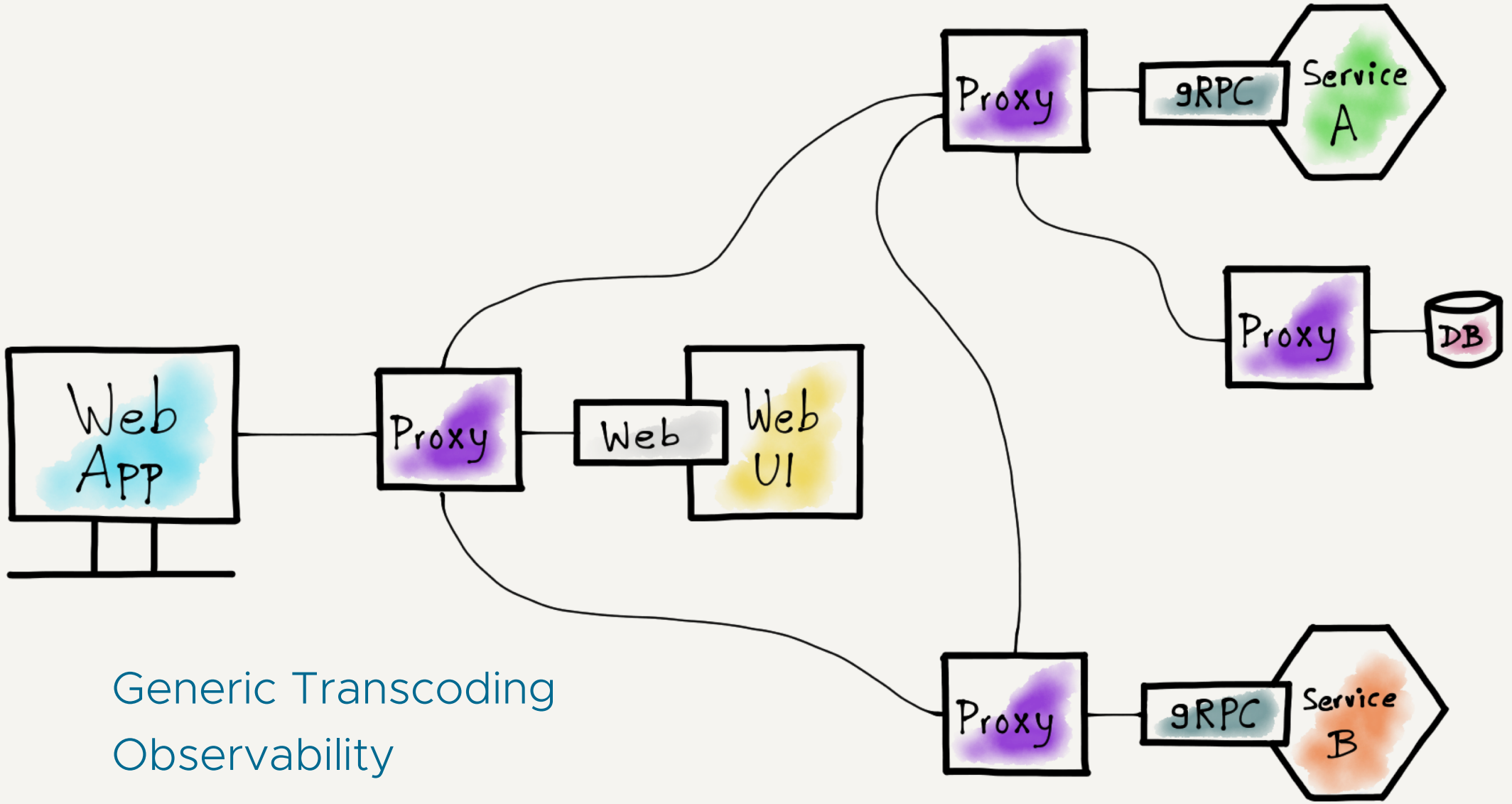


The Proxy Pattern



gRPC-Web Filter

gRPC-Web and Istio



Generic Transcoding

Observability

Benefits of a Service Mesh

Discoverable

Services can find each other

Resilient

Built-in robustness frameworks, load balancing & test infrastructure

Configurable

Configure services dynamically at runtime

Observable

Standardized metrics, monitoring & distributed tracing

Secure

Encrypt and protect service communication

gRPC-Web and Istio Demo

An Emoji Web App

https://github.com/venilnoronha/grpc-web-istio-demo

The API Definition

```
1  package proto;
2
3  service EmojiService {
4      rpc InsertEmojis (EmojiRequest) returns (EmojiResponse);
5  }
6
7  message EmojiRequest {
8      string input_text = 1;
9  }
10
11 message EmojiResponse {
12     string output_text = 1;
13 }
```

I like :pizza: and :sushi:!

I like 🍕 and 🍣!

Generate Definitions for Go and JavaScript

Parse and generate the Go file

```
protoc -I proto/ proto/emoji.proto \
       --go_out=plugins=grpc:proto
```

Parse and generate the JavaScript files

```
protoc -I proto/ proto/emoji.proto \
       --js_out=import_style=commonjs:proto \
       --grpc-web_out=import_style=commonjs, \
                      mode=grpcwebtext:proto
```

The Generated Go File

```
1  type EmojiServiceServer interface {
2      InsertEmojis(context.Context, *EmojiRequest) (*EmojiResponse, error)
3  }
4
5  type EmojiRequest struct {
6      InputText string `protobuf:"bytes,1,..."`
7  }
8
9  type EmojiResponse struct {
10     OutputText string `protobuf:"bytes,1,..."`
11 }
```

The Generated JavaScript Files

```
1  proto.EmojiRequest.prototype.setInputText = function(value) {
2      jspb.Message.setProto3StringField(this, 1, value);
3  };
4
5  proto.EmojiResponse.prototype.getOutputText = function() {
6      return jspb.Message.getFieldWithDefault(this, 1, "");
7  };
8
9  proto.EmojiServiceClient.prototype.insertEmojis =
10     function(request, metadata, callback) {
11         return this.client_.rpcCall('/proto.EmojiService/InsertEmojis',
12             request, metadata, callback);
13     };
```

The EmojiService Server

```
1  func (s *server) InsertEmojis(ctx context.Context,
2          req *proto.EmojiRequest) (*proto.EmojiResponse, error) {
3      outputText := emoji.Sprint(req.InputText)
4      return &proto.EmojiResponse{OutputText: outputText}, nil
5  }
6
7  func main() {
8      grpcServer := grpc.NewServer()
9      proto.RegisterEmojiServiceServer(grpcServer, &server{})
10
11     listener, err := net.Listen("tcp", ":9000")
12     grpcServer.Serve(listener)
13 }
```

```
$ go run cmd/server.go
2019/03/09 14:16:18 Listening on [::]:9000
```

The EmojiService Client

```
1  var server = flag.String("server", "localhost:9000", "Server address")
2  var text = flag.String("text", "Hello world!", "Input text")
3
4  func main() {
5      conn, err := grpc.Dial(*server, grpc.WithInsecure())
6      client := proto.NewEmojiServiceClient(conn)
7
8      req := &proto.EmojiRequest{InputText: *text}
9      res, err := client.InsertEmojis(context.Background(), req)
10     log.Printf("Server says: %s", res.OutputText)
11 }
```

```
$ go run cmd/client.go \
>         --text 'I like :pizza: and :sushi:!' \
>         --server 'localhost:9000'
2019/03/09 14:18:53 Request: I like :pizza: and :sushi:!
2019/03/09 14:18:53 Server says: I like 🍕 and 🍣!
$
```

The Server Container

```
1  FROM golang:1.12 as builder
2  WORKDIR /root/go/src/.../grpc-web-istio-demo/
3  COPY ./ .
4  RUN CGO_ENABLED=0 GOOS=linux \
5      go build -a -installsuffix cgo -v \
6      -o bin/server ./cmd/server.go
7
8  FROM scratch
9  WORKDIR /bin/
10 COPY --from=builder /root/.../grpc-web-istio-demo/bin/server .
11 ENTRYPOINT [ "/bin/server" ]
12 EXPOSE 9000
```

Build and Push the Server Image

Build the Docker image

```
1  docker build -f server.Dockerfile \
2               -t vnoronha/grpc-web-istio-demo:server .
```

Push the Docker image

```
1  docker push vnoronha/grpc-web-istio-demo:server
```

The Web UI HTML

```
<!DOCTYPE html>
<html>
    <body>
        <div id="editor" contentEditable="true"
                onkeyup="insertEmojis()">
        </div>
        <script src="dist/main.js"></script>
    </body>
</html>
```

The Web UI JavaScript

```
1  const {EmojiRequest, EmojiResponse} = require('./emoji_pb.js');
2  const {EmojiServiceClient} = require('./emoji_grpc_web_pb.js');
3
4  var client = new EmojiServiceClient('http://' + window.location.host);
5  var editor = document.getElementById('editor');
6
7  function insertEmojis() {
8      var req = new EmojiRequest();
9      req.setInputText(editor.innerText);
10     client.insertEmojis(req, {}, (err, res) => {
11         editor.innerText = res.getOutputText();
12     });
13 };
```

The Web UI Container

```
1  FROM node:8.15 as builder
2  WORKDIR /web-ui/
3  COPY ./ .
4  RUN npm install
5  RUN npx webpack app.js
6
7  FROM python:2.7
8  WORKDIR /web-ui/
9  COPY --from=builder /web-ui/ .
10 ENTRYPOINT [ "python" ]
11 CMD [ "-m", "SimpleHTTPServer", "9001" ]
12 EXPOSE 9001
```

Build and Push the Web UI Image

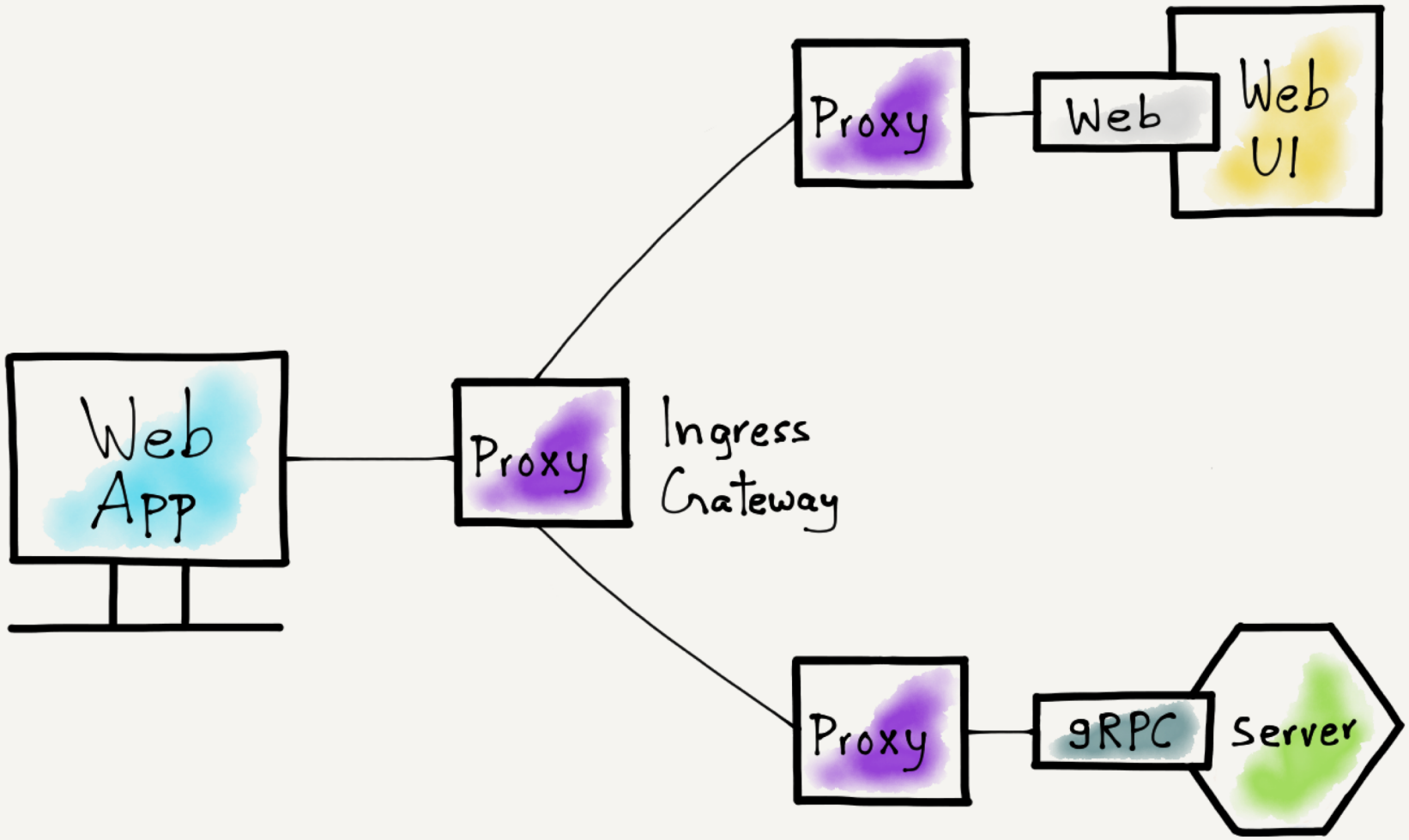
Build the Docker image

```
1  docker build -f docker/web-ui.Dockerfile
2               -t vnoronha/grpc-web-istio-demo:web-ui .
```

Push the Docker image

```
1  docker push vnoronha/grpc-web-istio-demo:web-ui
```

The Emoji Web App Deployment



The Kubernetes Configuration for the Server

```yaml
apiVersion: v1
kind: Service
metadata:
  name: server
  labels:
    app: server
spec:
  ports:
  - name: grpc-web
    port: 9000
  selector:
    app: server
```

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: server
spec:
  replicas: 1
  template:
    spec:
      containers:
      - name: server
        image: grpc-web-istio-demo:server
        ports:
        - containerPort: 9000
```

The Kubernetes Configuration for the Web UI

```yaml
apiVersion: v1
kind: Service
metadata:
  name: web-ui
  labels:
    app: web-ui
spec:
  ports:
  - name: http
    port: 9001
  selector:
    app: web-ui
```

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: web-ui
spec:
  replicas: 1
  template:
    spec:
      containers:
      - name: web-ui
        image: grpc-web-istio-demo:web-ui
        ports:
        - containerPort: 9001
```

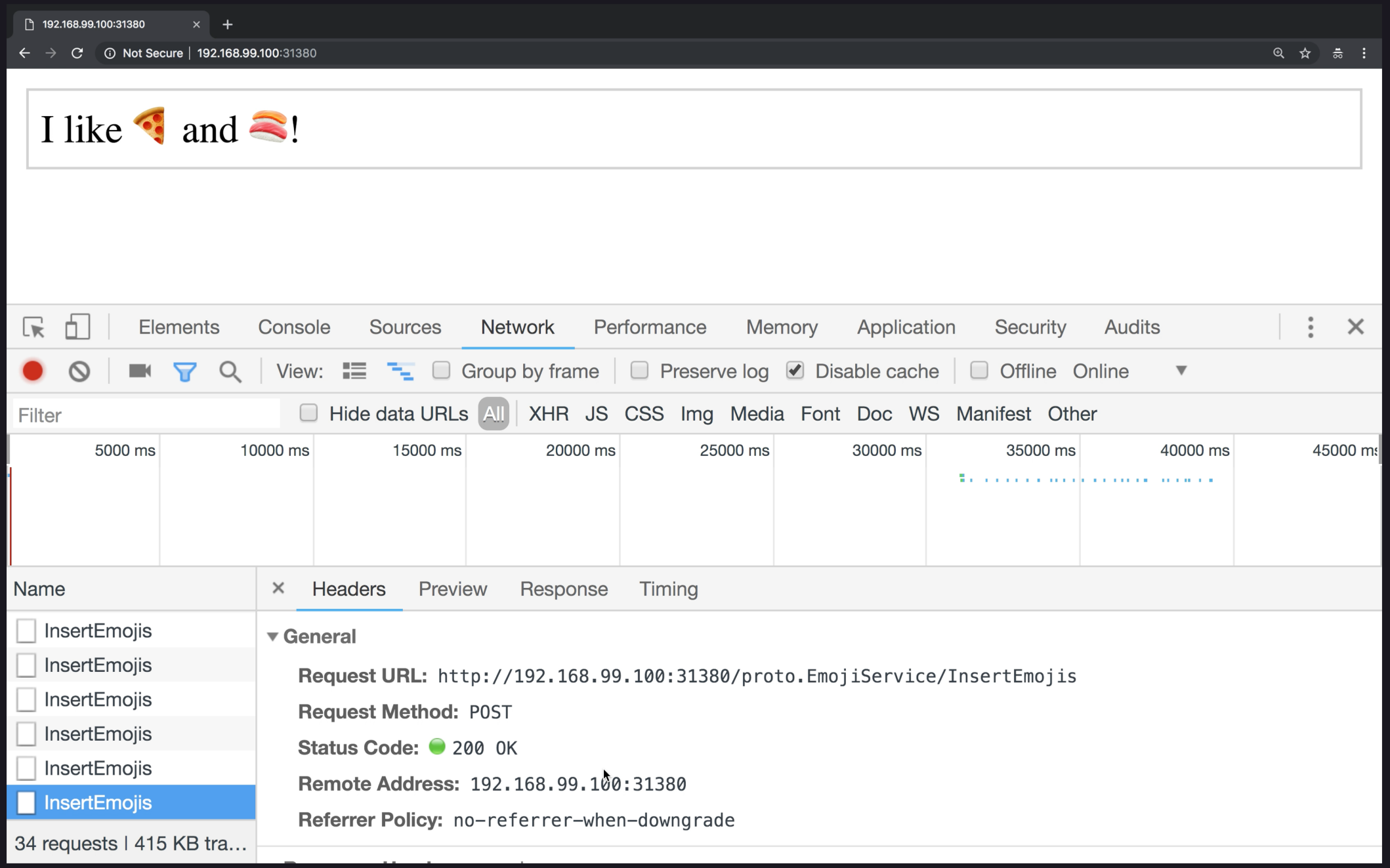
The Istio Gateway Configuration

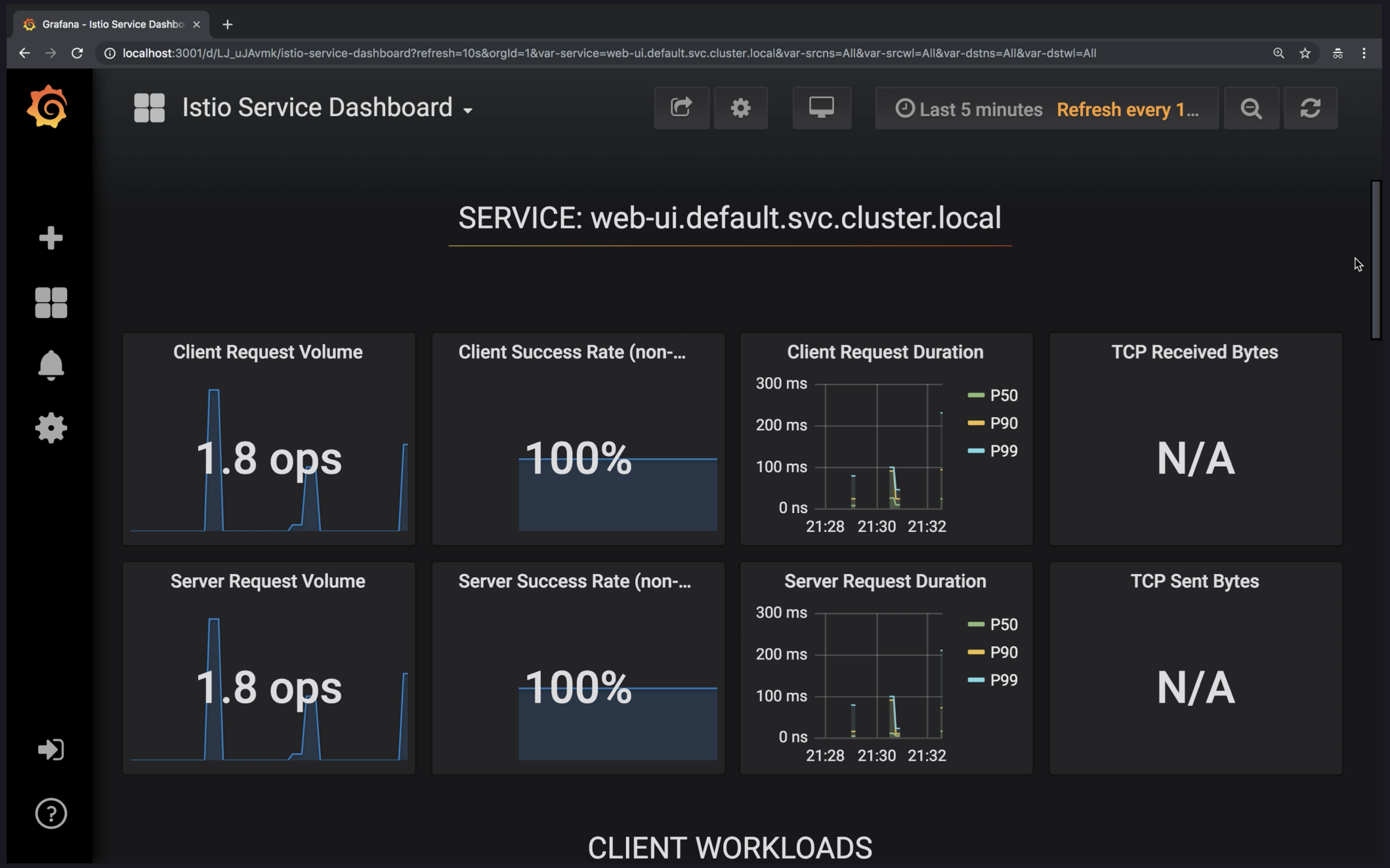
```yaml
apiVersion: istio.io/v1alpha3
kind: Gateway
metadata:
  name: gateway
spec:
  selector:
    istio: ingressgateway
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
```

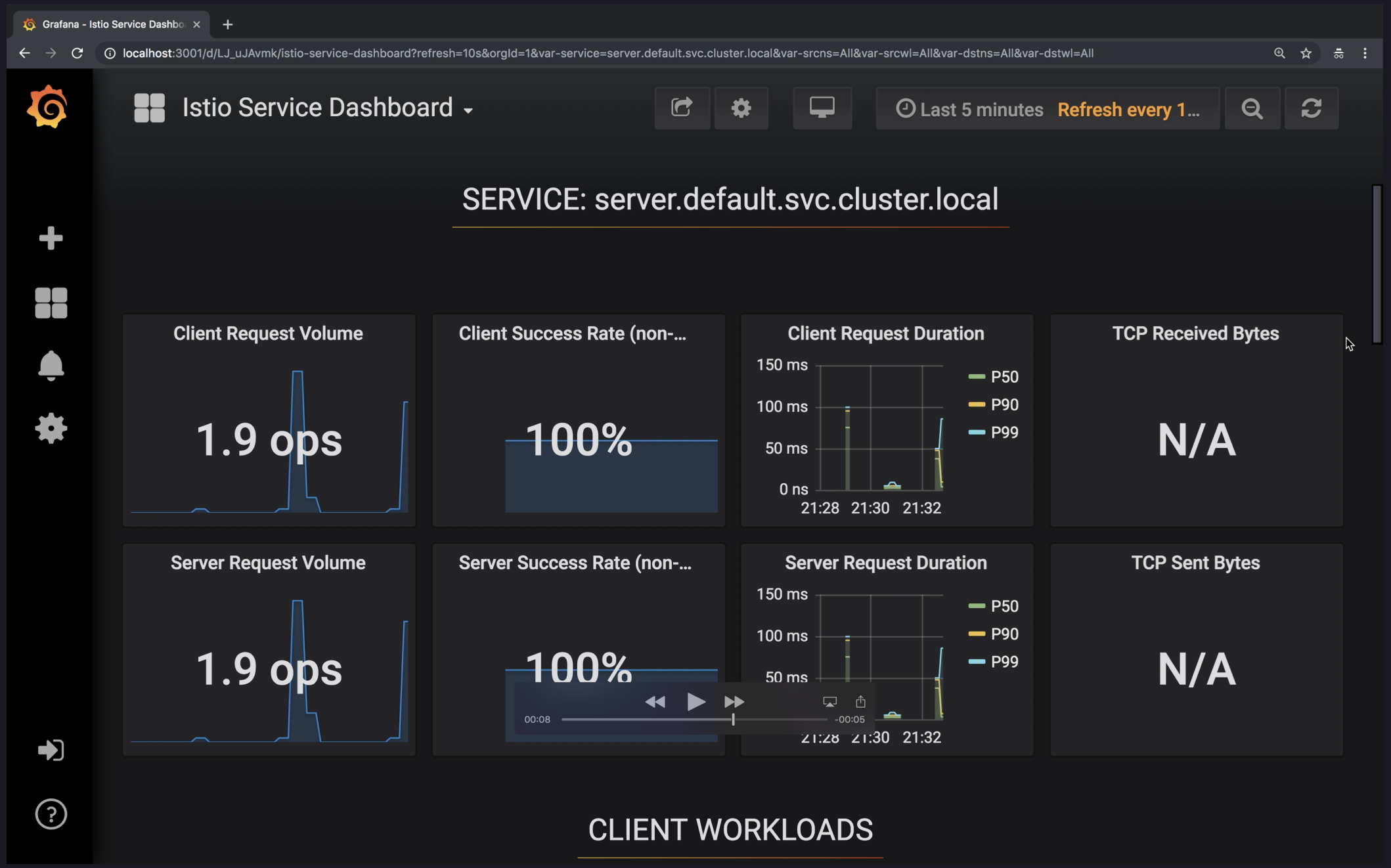
```yaml
apiVersion: istio.io/v1alpha3
kind: VirtualService
spec:
  gateways:
  - gateway
  http:
  - match:
    - uri:
        prefix: /proto.EmojiService
    route:
    - destination:
        host: server
  - route:
    - destination:
        host: web-ui
```

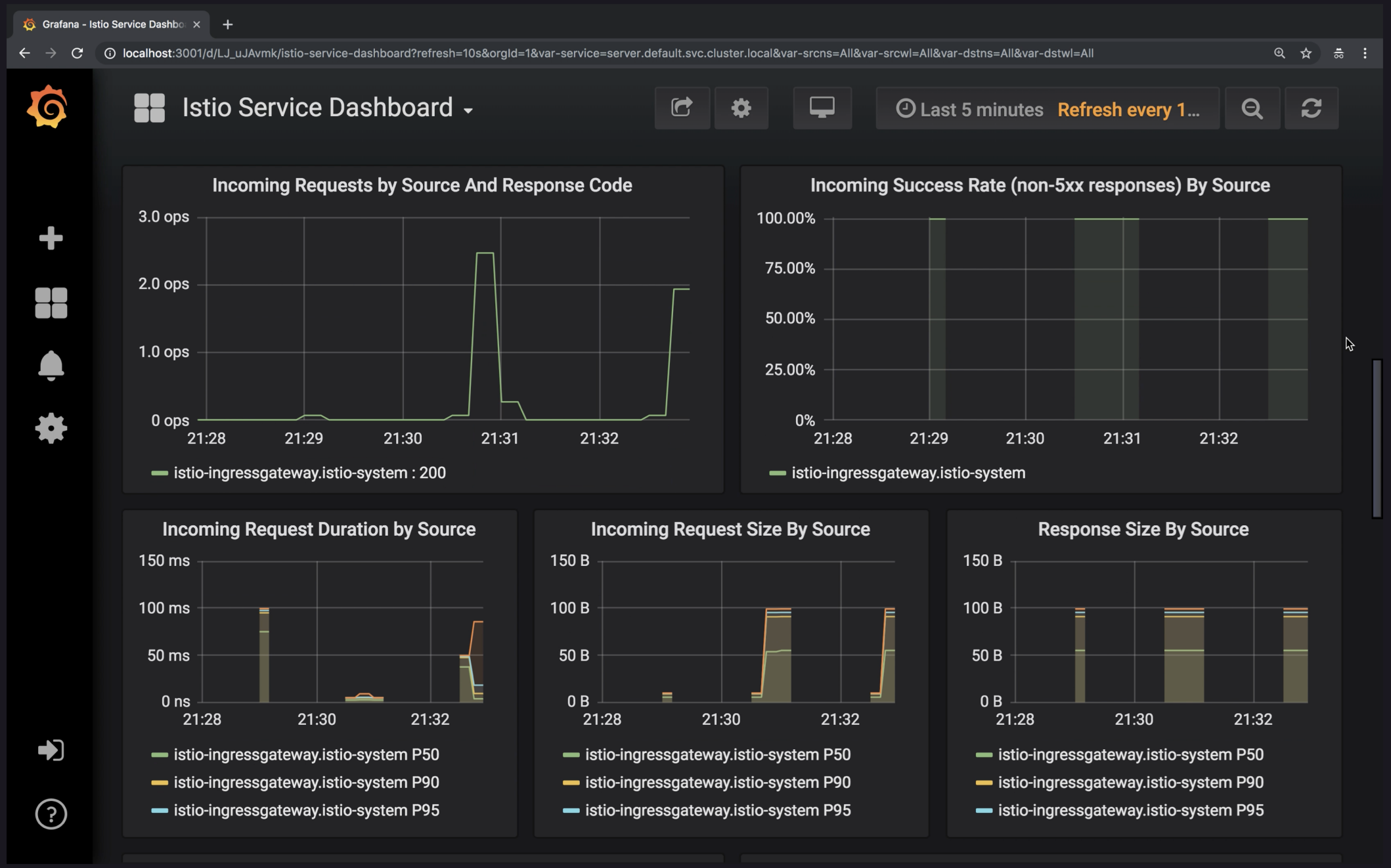
```
$ kubectl apply -f <(istioctl kube-inject -f server.yaml)
service/server created
deployment.extensions/server created
$
$ kubectl apply -f <(istioctl kube-inject -f web-ui.yaml)
service/web-ui created
deployment.extensions/web-ui created
$
$ kubectl apply -f gateway.yaml
gateway.networking.istio.io/gateway created
virtualservice.networking.istio.io/virtual-service created
$
```

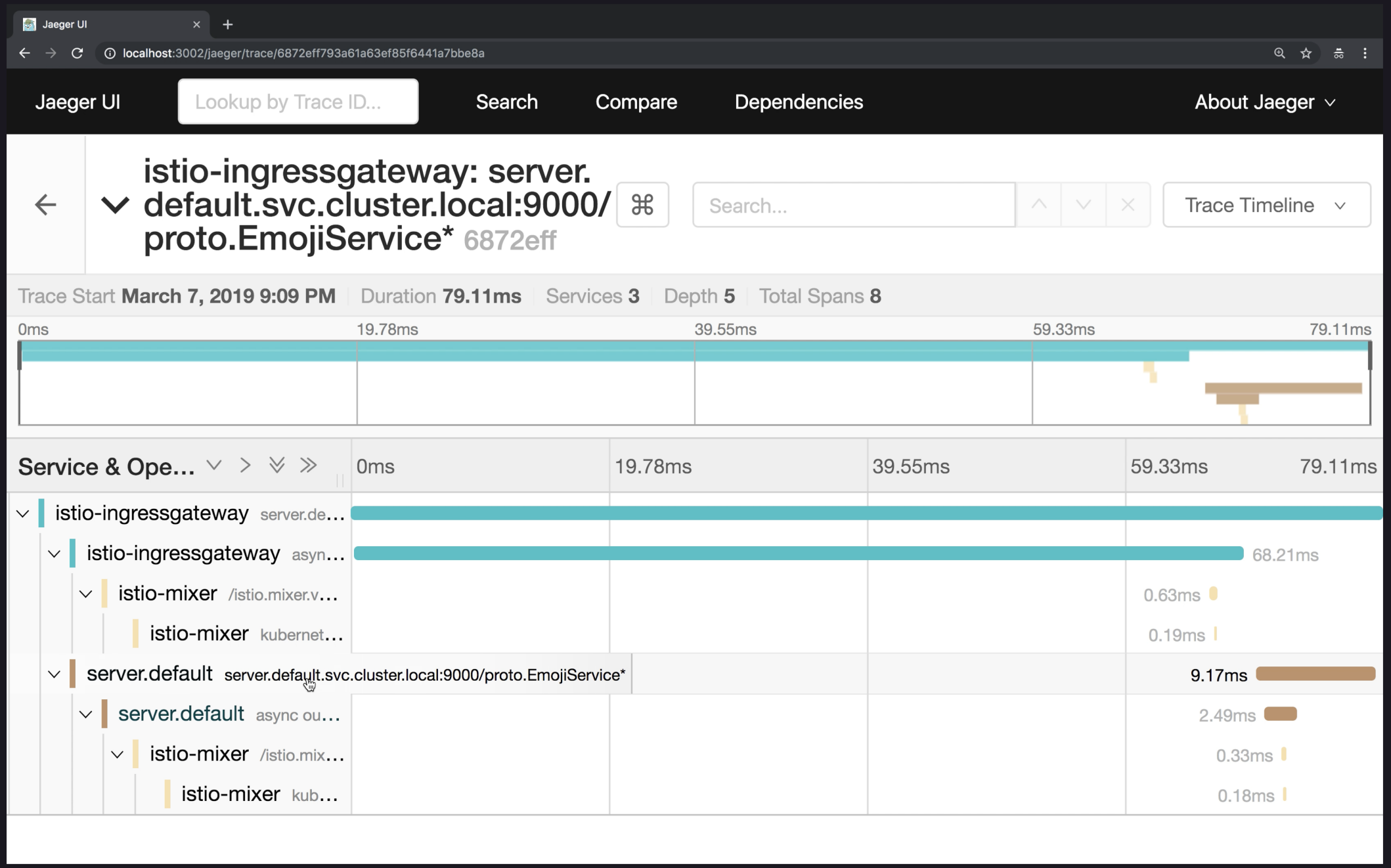
```
$ kubectl get pods
NAME                          READY     STATUS    RESTARTS   AGE
server-6985ccb646-x82pv       2/2       Running   0          26s
web-ui-dd6ddcbbc-7jwq4        2/2       Running   0          13s
$
```

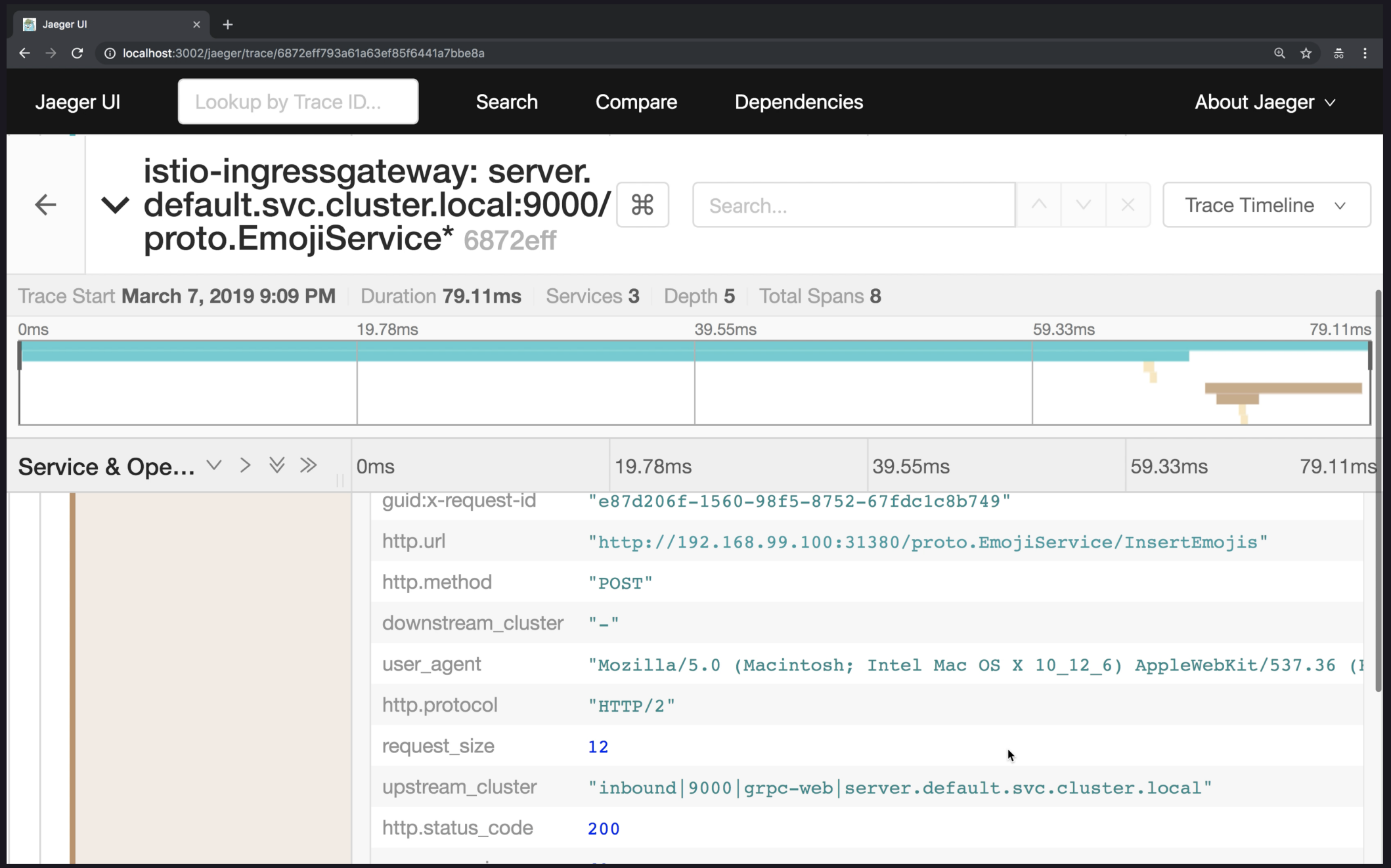

192.168.99.100:31380
Not Secure | 192.168.99.100:31380
I like 🍕 and 🍣!
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Audits
View:
Group by frame
Preserve log
Disable cache
Offline
Online
Filter
Hide data URLs
All
XHR
JS
CSS
Img
Media
Font
Doc
WS
Manifest
Other
5000 ms
10000 ms
15000 ms
20000 ms
25000 ms
30000 ms
35000 ms
40000 ms
45000 ms
Name
Headers
Preview
Response
Timing
InsertEmojis
InsertEmojis
InsertEmojis
InsertEmojis
InsertEmojis
InsertEmojis
34 requests | 415 KB tra...
General
Request URL: http://192.168.99.100:31380/proto.EmojiService/InsertEmojis
Request Method: POST
Status Code: 200 OK
Remote Address: 192.168.99.100:31380
Referrer Policy: no-referrer-when-downgrade

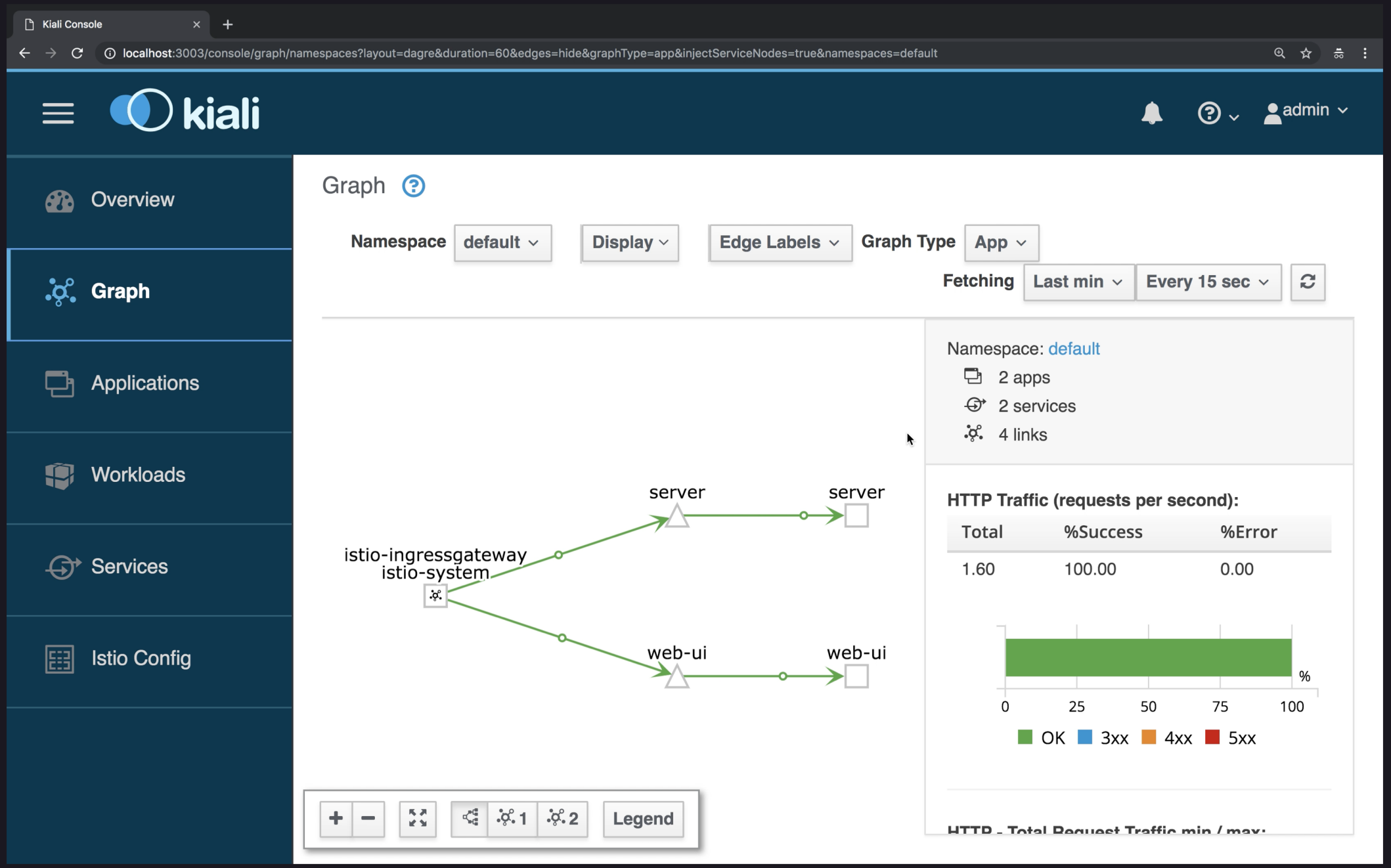

Grafana - Istio Service Dashbo
localhost:3001/d/LJ_uJAvmk/istio-service-dashboard?refresh=10s&orgId=1&var-service=web-ui.default.svc.cluster.local&var-srcns=All&var-srcwl=All&var-dstns=All&var-dstwl=All
Istio Service Dashboard
Last 5 minutes
Refresh every 1...
SERVICE: web-ui.default.svc.cluster.local
Client Request Volume
1.8 ops
Client Success Rate (non-...
100%
Client Request Duration
300 ms
200 ms
100 ms
0 ns
21:28
21:30
21:32
P50
P90
P99
TCP Received Bytes
N/A
Server Request Volume
1.8 ops
Server Success Rate (non-...
100%
Server Request Duration
300 ms
200 ms
100 ms
0 ns
21:28
21:30
21:32
P50
P90
P99
TCP Sent Bytes
N/A
CLIENT WORKLOADS

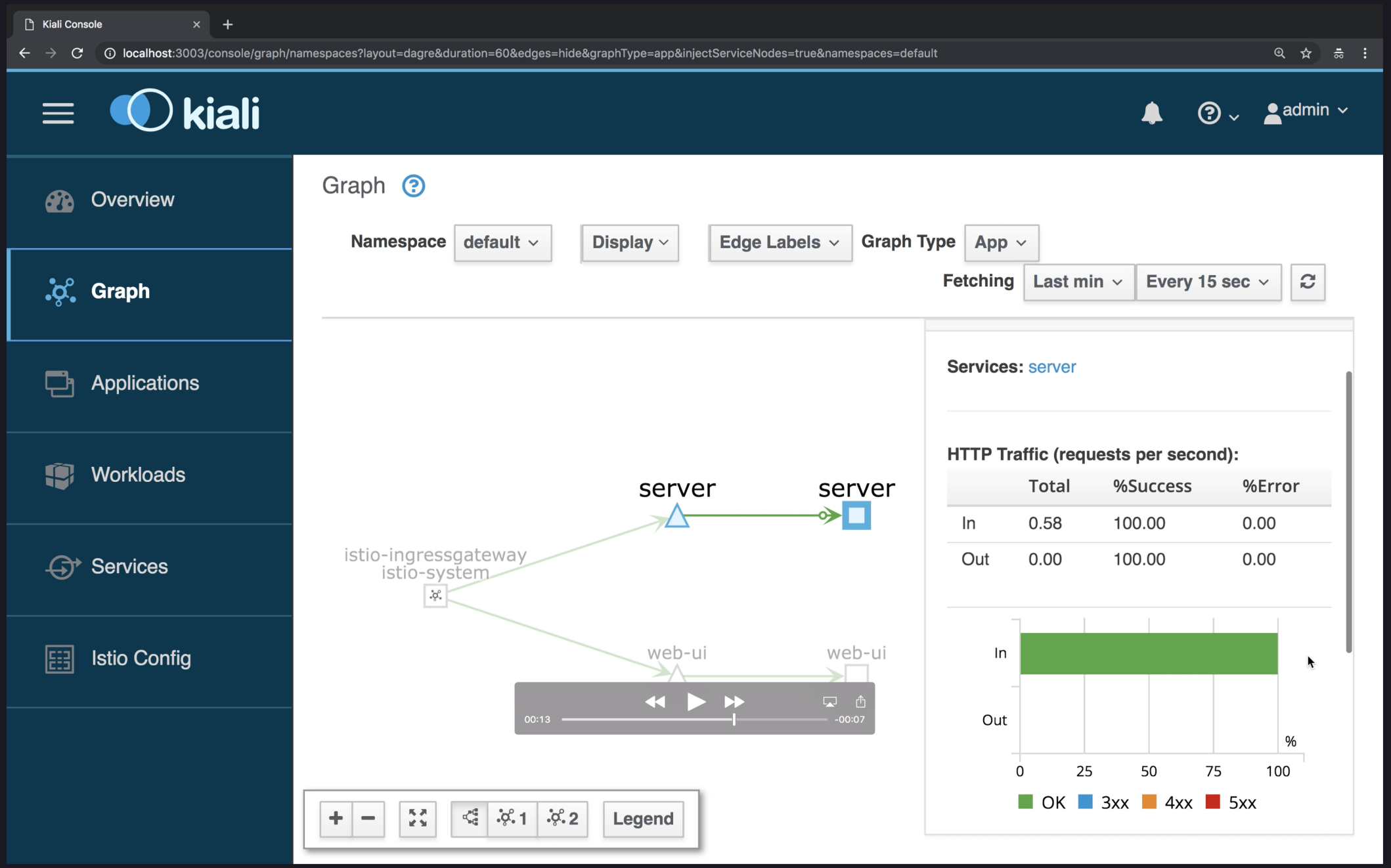

Grafana - Istio Service Dashbo
localhost:3001/d/LJ_uJAvmk/istio-service-dashboard?refresh=10s&orgId=1&var-service=server.default.svc.cluster.local&var-srcns=All&var-srcwl=All&var-dstns=All&var-dstwl=All
Istio Service Dashboard
Last 5 minutes
Refresh every 1...
SERVICE: server.default.svc.cluster.local
Client Request Volume
1.9 ops
Client Success Rate (non-...
100%
Client Request Duration
150 ms
100 ms
50 ms
0 ns
21:28
21:30
21:32
P50
P90
P99
TCP Received Bytes
N/A
Server Request Volume
1.9 ops
Server Success Rate (non-...
100%
00:08
-00:05
Server Request Duration
150 ms
100 ms
50 ms
21:28
21:30
21:32
P50
P90
P99
TCP Sent Bytes
N/A
CLIENT WORKLOADS


Grafana - Istio Service Dashbo
localhost:3001/d/LJ_uJAvmk/istio-service-dashboard?refresh=10s&orgId=1&var-service=server.default.svc.cluster.local&var-srcns=All&var-srcwl=All&var-dstns=All&var-dstwl=All
Istio Service Dashboard
Last 5 minutes
Refresh every 1...
Incoming Requests by Source And Response Code
3.0 ops
2.0 ops
1.0 ops
0 ops
21:28
21:29
21:30
21:31
21:32
istio-ingressgateway.istio-system : 200
Incoming Success Rate (non-5xx responses) By Source
100.00%
75.00%
50.00%
25.00%
0%
21:28
21:29
21:30
21:31
21:32
istio-ingressgateway.istio-system
Incoming Request Duration by Source
150 ms
100 ms
50 ms
0 ns
21:28
21:30
21:32
istio-ingressgateway.istio-system P50
istio-ingressgateway.istio-system P90
istio-ingressgateway.istio-system P95
Incoming Request Size By Source
150 B
100 B
50 B
0 B
21:28
21:30
21:32
istio-ingressgateway.istio-system P50
istio-ingressgateway.istio-system P90
istio-ingressgateway.istio-system P95
Response Size By Source
150 B
100 B
50 B
0 B
21:28
21:30
21:32
istio-ingressgateway.istio-system P50
istio-ingressgateway.istio-system P90
istio-ingressgateway.istio-system P95


Jaeger UI
localhost:3002/jaeger/trace/6872eff793a61a63ef85f6441a7bbe8a
Jaeger UI
Lookup by Trace ID...
Search
Compare
Dependencies
About Jaeger
istio-ingressgateway: server.
default.svc.cluster.local:9000/
proto.EmojiService* 6872eff
Search...
Trace Timeline
Trace Start March 7, 2019 9:09 PM
Duration 79.11ms
Services 3
Depth 5
Total Spans 8
0ms
19.78ms
39.55ms
59.33ms
79.11ms
Service & Ope...
istio-ingressgateway server.de...
istio-ingressgateway asyn...
68.21ms
istio-mixer /istio.mixer.v...
0.63ms
istio-mixer kubernet...
0.19ms
server.default server.default.svc.cluster.local:9000/proto.EmojiService*
9.17ms
server.default async ou...
2.49ms
istio-mixer /istio.mix...
0.33ms
istio-mixer kub...
0.18ms


Jaeger UI
localhost:3002/jaeger/trace/6872eff793a61a63ef85f6441a7bbe8a
Jaeger UI
Lookup by Trace ID...
Search
Compare
Dependencies
About Jaeger
istio-ingressgateway: server.
default.svc.cluster.local:9000/
proto.EmojiService* 6872eff
Search...
Trace Timeline
Trace Start March 7, 2019 9:09 PM
Duration 79.11ms
Services 3
Depth 5
Total Spans 8
0ms
19.78ms
39.55ms
59.33ms
79.11ms
Service & Ope...
guid:x-request-id "e87d206f-1560-98f5-8752-67fdc1c8b749"
http.url "http://192.168.99.100:31380/proto.EmojiService/InsertEmojis"
http.method "POST"
downstream_cluster "-"
user_agent "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (
http.protocol "HTTP/2"
request_size 12
upstream_cluster "inbound|9000|grpc-web|server.default.svc.cluster.local"
http.status_code 200


Kiali Console
localhost:3003/console/graph/namespaces?layout=dagre&duration=60&edges=hide&graphType=app&injectServiceNodes=true&namespaces=default
kiali
admin
Overview
Graph
Applications
Workloads
Services
Istio Config
Graph
Namespace default
Display
Edge Labels
Graph Type App
Fetching Last min Every 15 sec
server
server
istio-ingressgateway
istio-system
web-ui
web-ui
Legend
Namespace: default
2 apps
2 services
4 links
HTTP Traffic (requests per second):
Total %Success %Error
1.60 100.00 0.00
%
0 25 50 75 100
OK 3xx 4xx 5xx


Kiali Console
localhost:3003/console/graph/namespaces?layout=dagre&duration=60&edges=hide&graphType=app&injectServiceNodes=true&namespaces=default
kiali
admin
Overview
Graph
Applications
Workloads
Services
Istio Config
Graph
Namespace
default
Display
Edge Labels
Graph Type
App
Fetching
Last min
Every 15 sec
server
server
istio-ingressgateway
istio-system
web-ui
web-ui
00:13
-00:07
Services: server
HTTP Traffic (requests per second):
Total
%Success
%Error
In
0.58
100.00
0.00
Out
0.00
100.00
0.00
In
Out
%
0
25
50
75
100
OK
3xx
4xx
5xx
1
2
Legend

Conclusion

- Protobufs – API Contracts, Data Models, Compatibility
- gRPC – Based on HTTP/2, Client Stubs, Performance
- gRPC-Web – Protobufs + gRPC
- Envoy – Built-in HTTP-gRPC Transcoder
- Istio – Envoy, Metrics, Tracing, Service Graph

Hack with gRPC-Web and Istio!

Thank You

venilnoronha.io

@venilnoronha

@venilnoronha