



API Management & Istio

Integration and challenges

Alejandro Martinez Ruiz <amr@redhat.com>
Modern Integration and Application Development Day
Milano, April 3rd, 2019

API Management is “business”

Process of publishing web APIs, enforcing usage policies, controlling access, nurturing a subscriber community, collecting and analyzing usage statistics, and reporting on performance

Service Mesh is “infrastructure”

Infrastructure layer to make service to service communications safe, fast and reliable, providing service discovery, load balancing, authn and authz, secure communications, observability and others

API Management vs Service Mesh

Similarities and differences

API Management

Business and people centric

Resources are APIs and endpoints

Subjects are apps and users

Authentication & authorization

Controls access and tracks usage

Service Mesh

Infrastructure centric

Resources are services, routes, pods

Subjects are services and requests

Authentication & authorization

Controls access and tracks usage

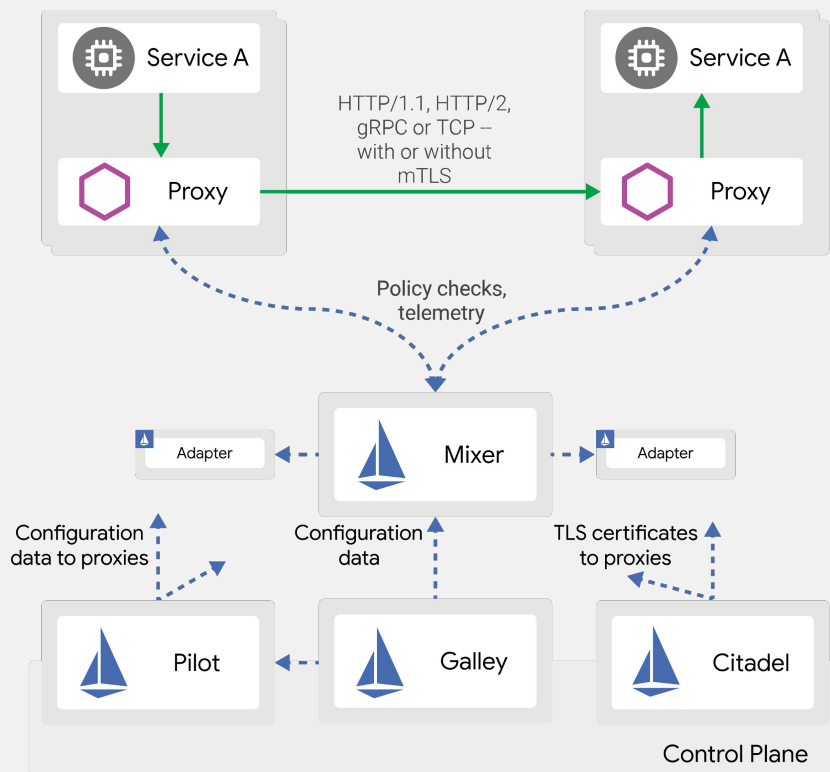
“API Management” & “Service Mesh”

Enforcing usage policies, controlling access, collecting and analyzing usage statistics, and reporting on performance all differ in scope but share the mechanisms, ie. authentication, rate limiting, ...

Istio is a “Service Mesh”

Backed by Google, IBM, Lyft and Red Hat

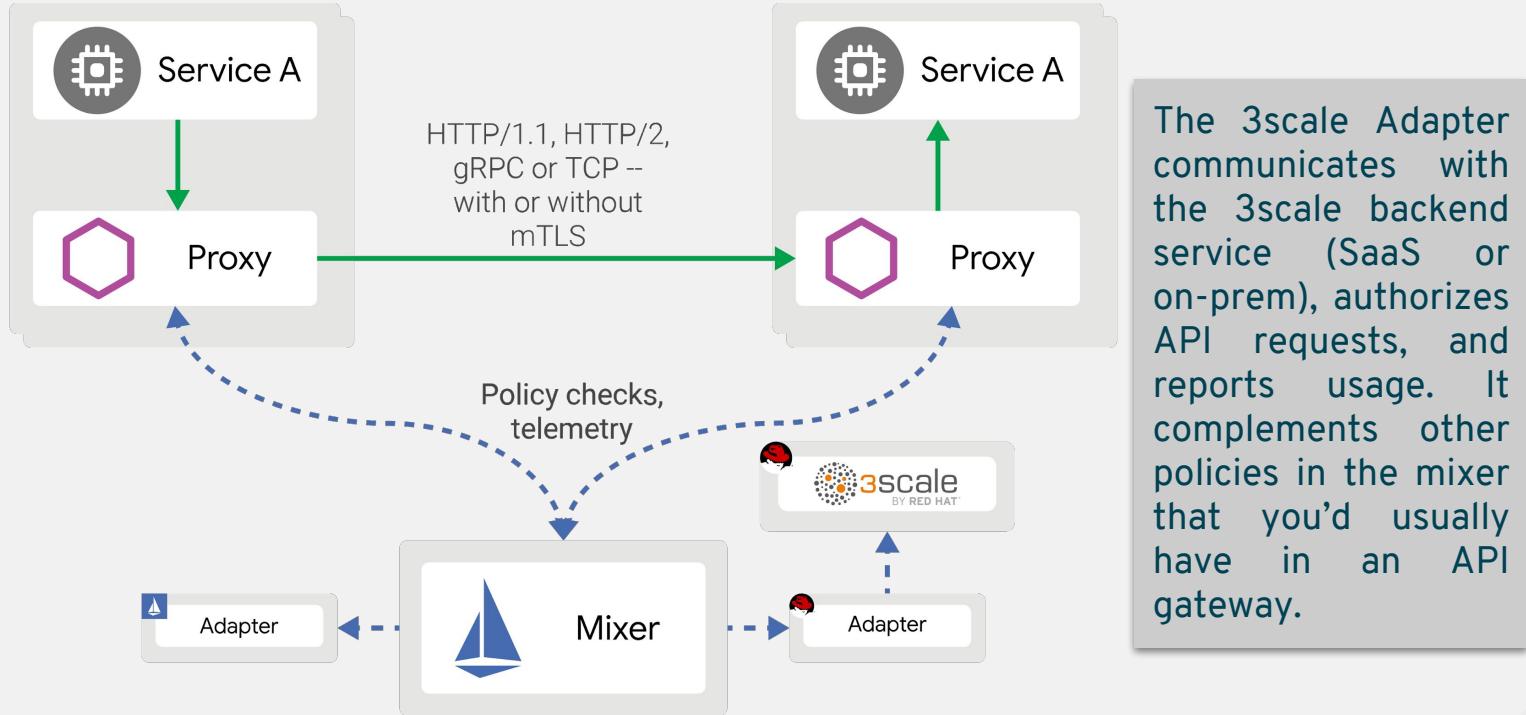
Istio Architecture and Features



- Traffic Management
 - ◆ Service Discovery
 - ◆ Intelligent Routing
 - ◆ Resiliency
- Security
 - ◆ Authentication
 - ◆ Authorization
 - ◆ Encryption
- Observability
 - ◆ Tracing
 - ◆ Monitoring
 - ◆ Logging

Istio Architecture with API Management

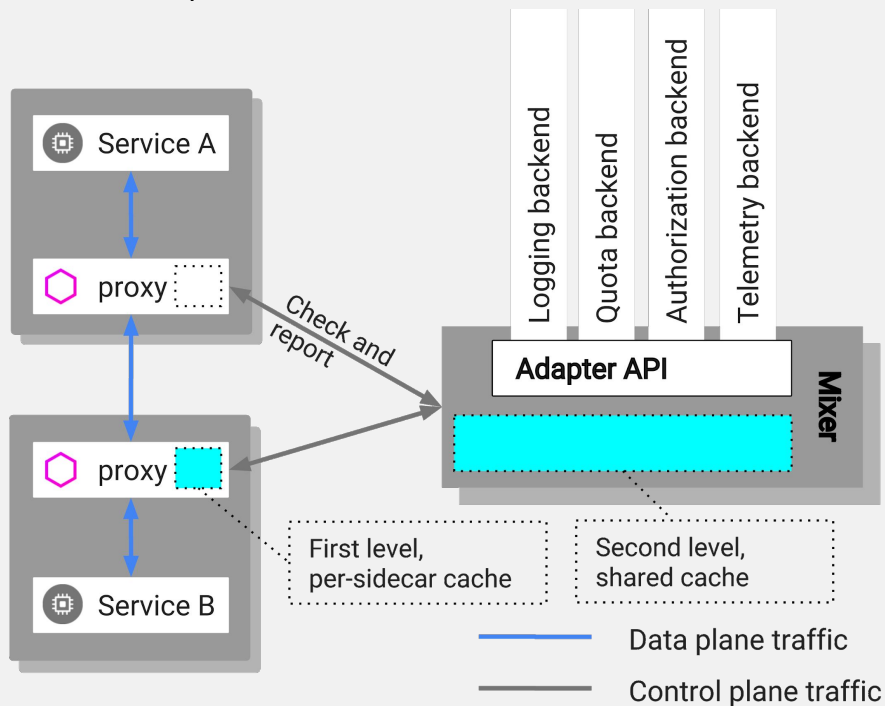
Integration with Mixer



Mixer Adapter Model

Features & Topology

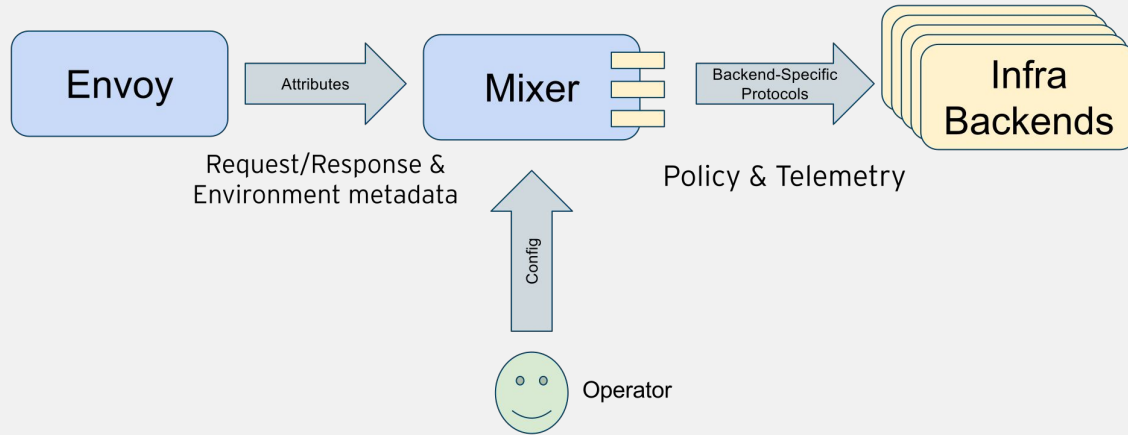
Mixer Adapter Model



- Mixer mediates communication to and from services
- Collects request or environment attributes from proxies
- Enforces access control and usage policies
- Sends telemetry data
- 1st and 2nd level caches to reduce latencies
- Plugin model “adapter” for custom policies

Mixer is an Attribute Processing Machine

Mixer Adapter Model



1. Envoy generates attributes from request, responses and the environment
2. Mixer processes attributes and routes them to adapters according to config
3. Adapters receive them and integrate infrastructure backends (ie. Prometheus, 3scale)

Mixer Adapter Model

Templates & Instances

Adapters & Handlers

Rules

Templates & Instances are Schema & Input

Mixer Adapter Model

Templates describe bundles of data that an adapter needs at request time. One adapter can support several templates.

A few templates ship out of the box, like Quota, Authorization or Trace Span.

Instances specify how the fields should be filled with strongly-typed attribute expressions.

```
20 # instance for template authorization-
19 apiVersion: "config.istio.io/v1alpha2"-
18 kind: instance-
17 metadata:-
16   name: threescale-authorization-istiodevel-admin.3scale.net-123456-
15   namespace: istio-system-
14   labels:-
13     • "service-mesh.3scale.net/host": "istiodevel-admin.3scale.net"-
12     • "service-mesh.3scale.net/service-id": "123456"-
11 spec:-
10   template: threescale-authorization-
9   params:-
8     • subject:-
7       • user: request.query_params["user_key"] | request.headers["User-Key"] | ""-
6       • properties:-
5         • app_id: request.query_params["app_id"] | request.headers["App-Id"] | ""-
4         • app_key: request.query_params["app_key"] | request.headers["App-Key"] | ""-
3     • action:-
2       • path: request.url_path-
1       • method: request.method | "get"-
```

Adapter kind declares input & config schema

Mixer Adapter Model

Adapter kind specifies whether it's session based (ie. stores once its configuration), the data input schema (templates) it will be handling, and the configuration schema (ie. we want to know where the 3scale instance is located).

The Istio tooling autogenerates the “config” protobufs descriptor.

```
# this config is created through command-  
# mixgen adapter -c $GOPATH/src/istio.io/istio/mixer/adapter/3scale-istio-a  
-pter/config -s=false -n threescale -t threescale-authorization-  
apiVersion: "config.istio.io/v1alpha2"-  
kind: adapter-  
metadata:-  
  name: threescale-  
  namespace: istio-system-  
spec:-  
  description:-  
  session_based: false-  
  templates:-  
  - threescale-authorization-  
  config: CvD6AgogZ29vZ2xlL3Byb3RvYnVmL2Rlc2NyaXB0b3IucHJvdG8SD2dvv2dsZS5w
```

Handlers provide Adapter configuration

Mixer Adapter Model

Handlers instantiate the configuration for the adapter and specify its gRPC service address. There could be many handlers for a single adapter.

The 3scale Istio adapter obtains credentials from the handler for a specific 3scale instance and service, since access tokens work per-service.

```
14 # handler for adapter threescale-
13 apiVersion: "config.istio.io/v1alpha2"-
12 kind: handler-
11 metadata:-
10   name: threescale-istiodevel-admin.3scale.net-123456.handler.istio-system-
9   namespace: istio-system-
8 spec:-
7   adapter: threescale-
6   params:-
5     • service_id: "123456"-
4     • system_url: "https://istiodevel-admin.3scale.net"-
3     • access_token: "secret-token"-
2   connection:-
1   address: "threescale-istio-adapter:3333"-
0 -
~
~
```

Rules

Mixer Adapter Model

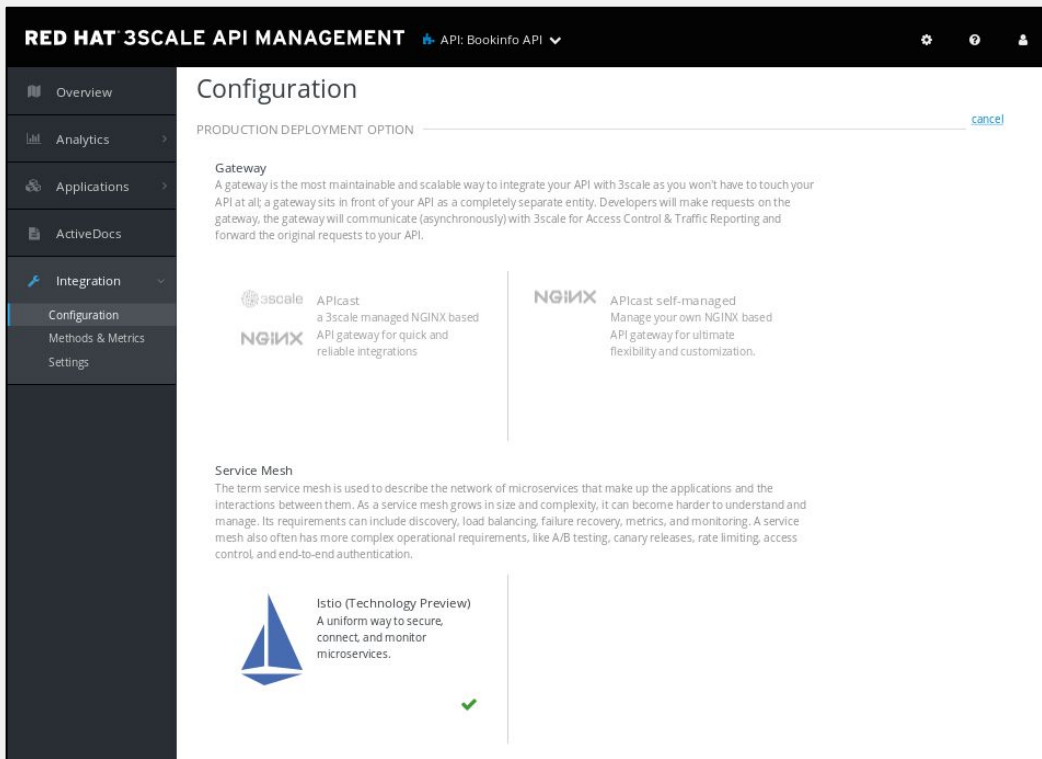
Rules specify a handler and a set of instances and a matching predicate, which is an attribute expression returning a boolean.

If the result of the predicate is true, Mixer fills in the instances and sends the data to the adapter described by the handler.

```
16 apiVersion: "config.istio.io/v1alpha2"~
15 kind: rule~
14 metadata:~
13   name: threescale-replace-me.3scale.net-443-example-service-id~
12   namespace: istio-system~
11   labels:~
10     • "service-mesh.3scale.net/host": "istiodevel-admin.3scale.net"~
9     • "service-mesh.3scale.net/service-id": "123456"~
8 spec:~
7   match: destination.labels["service-mesh.3scale.net"] == "true" &&~
6     • destination.labels["service-mesh.3scale.net/uid"] ==~
5     • ..... "istiodevel-admin.3scale.net-123456"~
4   actions:~
3     - handler: threescale-istiodevel-admin.3scale.net-123456.handler.istio-system~
2     • instances:~
1     • - threescale-authorization-istiodevel-admin.3scale.net-123456~
0 ~
```


Configuring an Istio API in 3scale

Configuring an Istio API in 3scale (I)





RED HAT 3SCALE API MANAGEMENT API: Bookinfo API


Configuration

PRODUCTION DEPLOYMENT OPTION [cancel](#)

Gateway
A gateway is the most maintainable and scalable way to integrate your API with 3scale as you won't have to touch your API at all; a gateway sits in front of your API as a completely separate entity. Developers will make requests on the gateway, the gateway will communicate (asynchronously) with 3scale for Access Control & Traffic Reporting and forward the original requests to your API.

Logo	Provider	Description
	3scale	APIcast, a 3scale managed NGINX based API gateway for quick and reliable integrations
	NGINX	APIcast self-managed. Manage your own NGINX based API gateway for ultimate flexibility and customization.

Service Mesh
The term service mesh is used to describe the network of microservices that make up the applications and the interactions between them. As a service mesh grows in size and complexity, it can become harder to understand and manage. Its requirements can include discovery, load balancing, failure recovery, metrics, and monitoring. A service mesh also often has more complex operational requirements, like A/B testing, canary releases, rate limiting, access control, and end-to-end authentication.

	Istio (Technology Preview)	A uniform way to secure, connect, and monitor microservices.
---	----------------------------	--

A new “Service Mesh” option can be selected when configuring the integration’s production deployment option.

Configuring an Istio API in 3scale (II)

The screenshot displays the 'RED HAT 3SCALE API MANAGEMENT' interface for the 'API: Bookinfo API'. The left sidebar contains navigation options: Overview, Analytics, Applications, ActiveDocs, Integration, Configuration, Methods & Metrics, and Settings. The main content area is titled 'AUTHENTICATION' and includes a sub-section 'Authentication' with the text: 'Authentication is essential to provide Access Control. The chosen authentication mode dictates how your customers will authenticate with your API.'

Three authentication methods are listed:

- API Key (user_key)**: The application is identified & authenticated via a single string. This option is currently selected, indicated by a green checkmark.
- App_ID and App_Key Pair**: The application is identified via the App_ID and authenticated via the App_Key.
- OpenID Connect**: Use OpenID Connect for any OAuth 2.0 flow.

An 'Update Service' button is located at the bottom right of the configuration area.

Currently supported authentication methods are API Key and App ID + App Key, with Open ID Connect coming up soon

Configuring an Istio API in 3scale (III)

The screenshot shows the 'Integration' configuration page for the 'Bookinfo API' in the Red Hat 3Scale API Management console. The page features a sidebar with navigation options: Overview, Analytics, Applications, ActiveDocs, Integration (selected), Configuration, Methods & Metrics, and Settings. The main content area is titled 'Integration' and contains a section for 'MAPPING RULES'. This section includes a table with the following data:

Verb	Pattern		Metric or Method (Define)	Last?
GET	/	1	hits	<input type="checkbox"/>
GET	/productpage	1	products	<input type="checkbox"/>

Below the table, there is a green '+ Add Mapping Rule' button. At the bottom right of the configuration area, there is a blue 'Update' button and a link to 'Back to Integration & Configuration'. The footer of the page includes 'Privacy Refunds Support' and 'Powered by 3scale'.

We can add mapping rules for methods (endpoints) we want to manage, as well as limits (ie. 4 hits per minute)

Configuring an Istio API in 3scale (IV)

Configuration, Methods and Settings

Follow the instructions on the [Integration page](#) to download and install the Istio Service Mesh into your codebase.

 [Get started with the Istio Service Mesh](#)

Authenticated by **API key**

ID for API calls is **2555417783508** and system name is **bookinfo**

Users **can** manage application keys

Users **can** manage applications

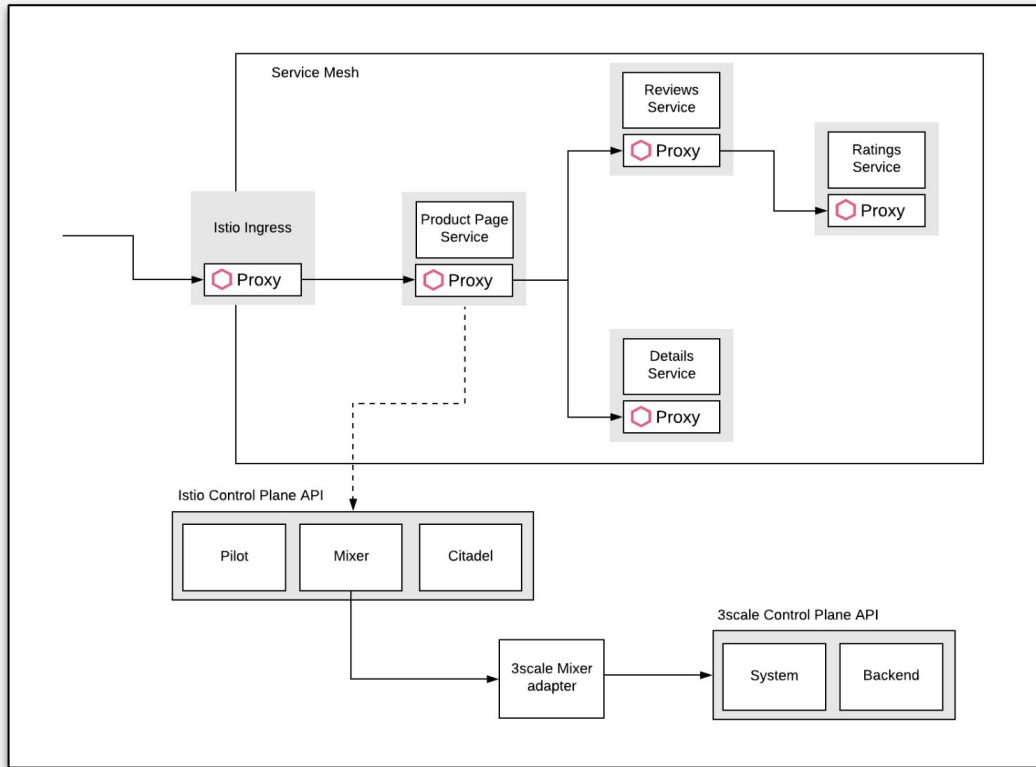
Users can **request plan change**

Users **cannot select a plan** when creating an application

You will need to gather the service id for your API and create an access token so the adapter can read the configuration and use the Service Management API to perform authorizations and reports

Demo

Demo | Bookinfo | Protect the service



Protect the Bookinfo service with an API key authentication pattern using the 3scale adapter.

1. Deploy Istio
2. Deploy Bookinfo app
3. Integrate in 3scale
4. Deploy the adapter
5. Test integration

Challenges

Istio Challenges

Problems

Istio SM

Mixer & RTT Latencies
Ease of configuration

3scale Adapter

3scale Latencies
Ease of configuration

Istio Challenges

Solutions

Istio SM

Mixer & RTT Latencies

Mixer v2 (aka Envoy All-in)

Ease of configuration

Kiali?

3scale Adapter

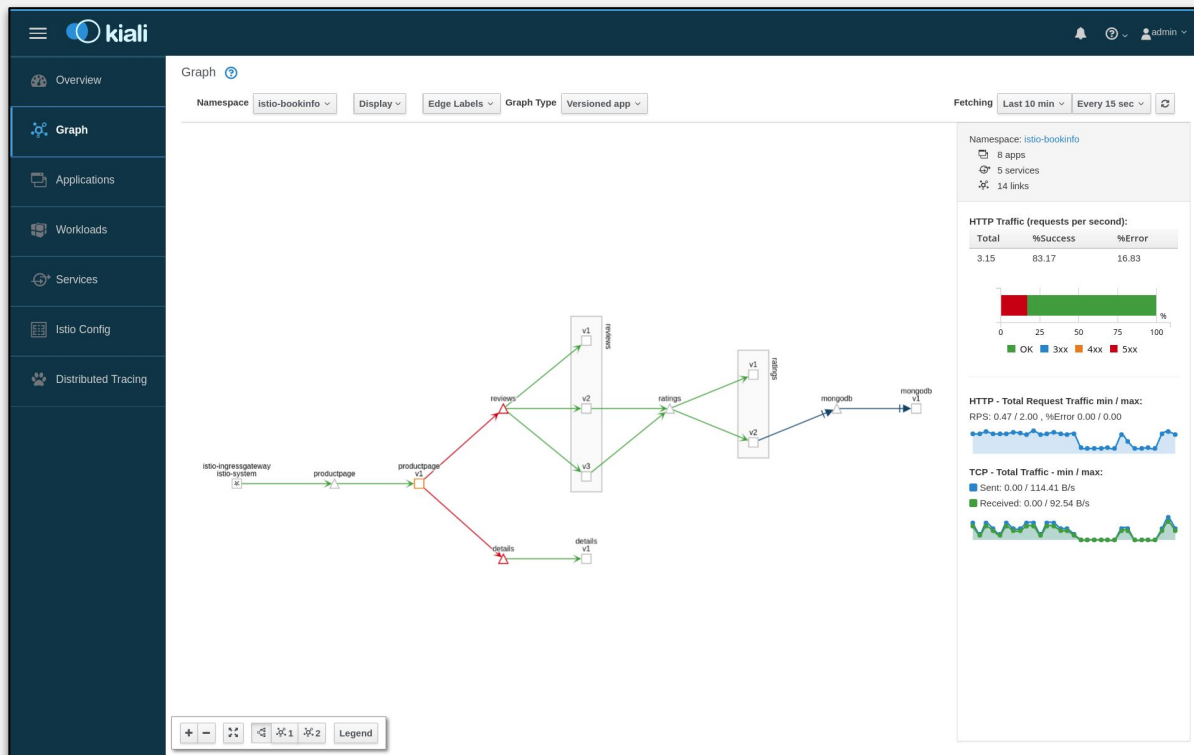
3scale Latencies

**Distributed Authorization +
Report Cache**

Ease of configuration

CLI tool + Kiali

Kiali - Observability & Integration



Maistra

Istio distribution by Red Hat

Kiali + Prometheus + Grafana + Jaeger + Service Catalog + 3scale Adapter

<https://maistra.io>

Install OpenShift & Maistra on CentOS/RHEL 7

```
curl -sSfL https://bit.ly/gist-get | bash -s -- https://bit.ly/maistra-install-sh
```

3scale Istio Adapter

<https://github.com/3scale/3scale-istio-adapter>

3scale Istio Adapter

3scale / 3scale-istio-adapter

Watch 3 Unstar 17 Fork 5

Code Issues 6 Pull requests 1 Projects 0 Wiki Insights Settings

Red Hat 3scale Istio Mixer Adapter - Add 3scale's API Management to the Service Mesh [Edit](#)

Manage topics

203 commits 7 branches 5 releases 3 contributors Apache-2.0

Branch: master New pull request Create new file Upload files Find File Clone or download -

PhilipGough Merge pull request #80 from 3scale/release-process Latest commit scdcefd 2 days ago

.circleci	circleci: correct references to the old directory structure	3 months ago
cmd	docs: cli - documents the version flag	2 days ago
config	config: Re-generate config against istio 1.1.0 tag	13 days ago
deploy	release: prep for v0.5.0	7 days ago
istio	config: Copy adapter config to istio and testdata directory	13 days ago
pkg	test: templating/unit - Adds unit tests to k8 validations public API	7 days ago
scripts	scripts: Add go script to generate yaml Deployment definition	5 days ago
testdata	config: Copy adapter config to istio and testdata directory	13 days ago
.gitignore	Fix paths in .gitignore	2 months ago
DEVEL.md	docs: Document the release target in 'make release'	2 days ago
Dockerfile	Makefile,Dockerfile: Support building a release via Make target	2 days ago
Dockerfile.dev	Update paths in Dockerfiles, Makefile, generate script and doc refs	2 months ago



THE END



plus.google.com/+RedHat



facebook.com/redhatinc



linkedin.com/company/red-hat



twitter.com/RedHat



youtube.com/user/RedHatVideos