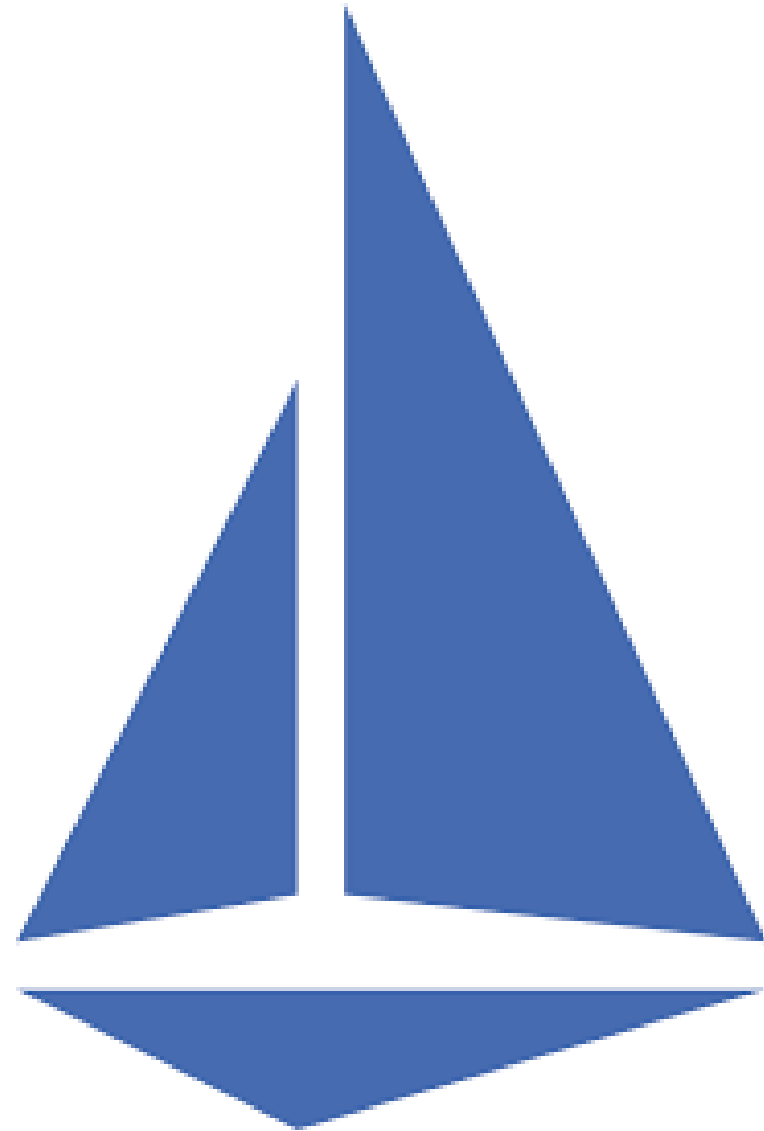


Les maillages de services avec Istio

Antoine Auffret
Emmanuel Sauvegrain



PRÉSENTATION DU PLAN



Introduction au contexte actuel



Présentation des maillages de services



Présentation de la solution Istio



Présentation du fonctionnement d'Istio



Mise en place de la solution



Introduction et contexte

Vision de plus en plus fragmentée des applications.

Augmentation du nombres d'architectures microservices.

Certaines applications d'envergure ne pourraient pas fonctionner sans une architecture microservices.

Les limites des architectures microservices

Les avantages

- Meilleure gestion en cas de panne. (Seul le service tombe).
- Développement simplifié et autonome.

Les limites

- Perte de contrôle de la solution au fil du temps.
- Difficulté à modifier et mettre en place des règles.
- Graves problèmes pour les applications d'envergures.

Présentation des maillages de services

- **Un maillage de service à quoi ça sert ?**
 - Permet de pallier aux problèmes des architectures microservices.
 - Apporte une couche applicative dédiée à la gestion de l'architecture.
- **Les objectifs des maillages de services ?**
 - Mise en place simple et rapide sur des réseaux imposants et complexes.
 - Installation de la solution sans toucher aux codes des différents services.

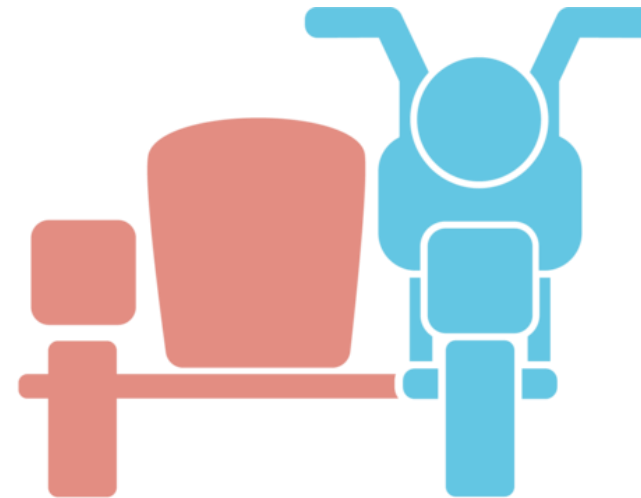
Présentation d'Istio

- Istio est la solution de maillage de service la plus réputée et la plus utilisée actuellement.
- Solution open-source développé avec le soutien d'IBM, Google cloud, Red-Hat et Lyft.
- Les 4 principes d'Istio:
 - La gestion du trafic
 - L'observabilité du réseau
 - La sécurité
 - La mise en place de politiques de contrôles

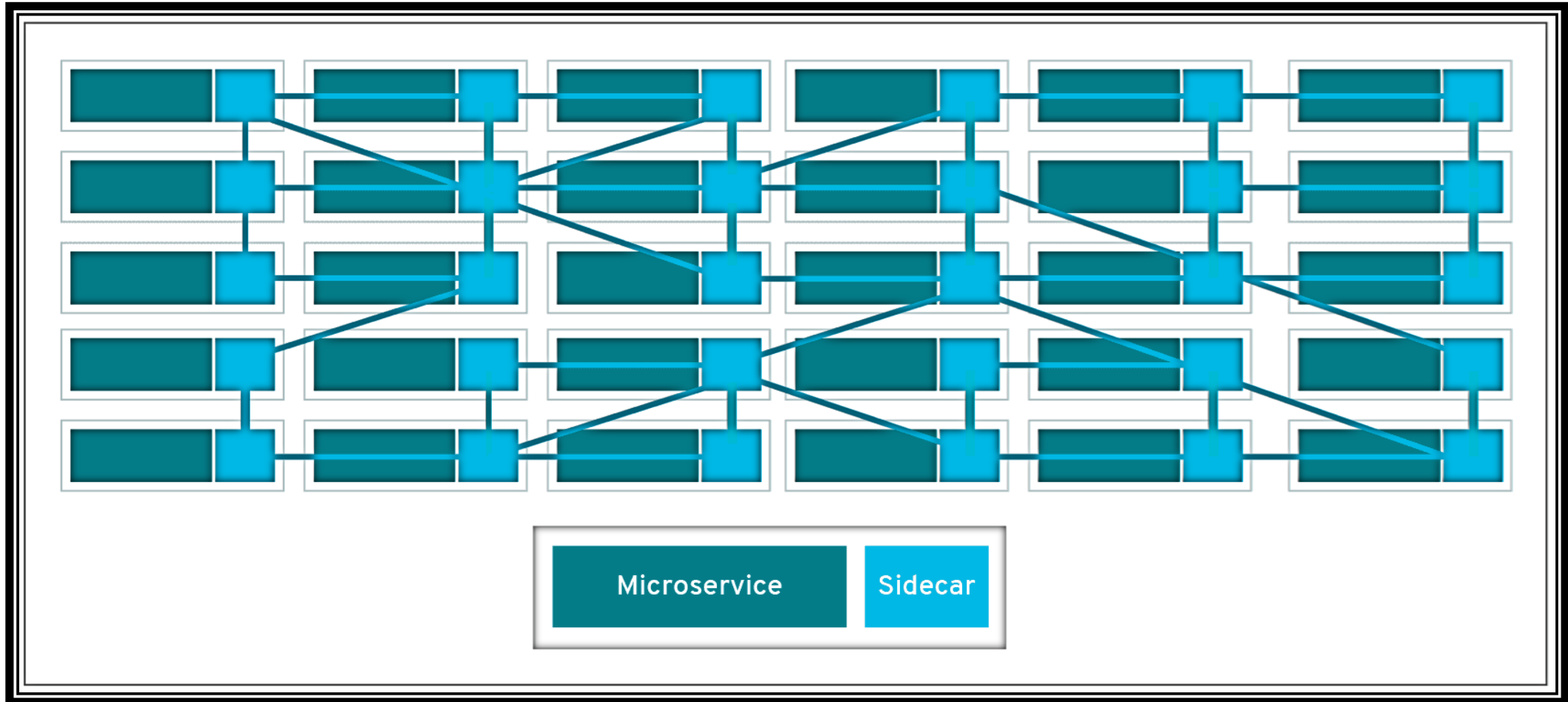


Comment l'information est-elle collectée ?

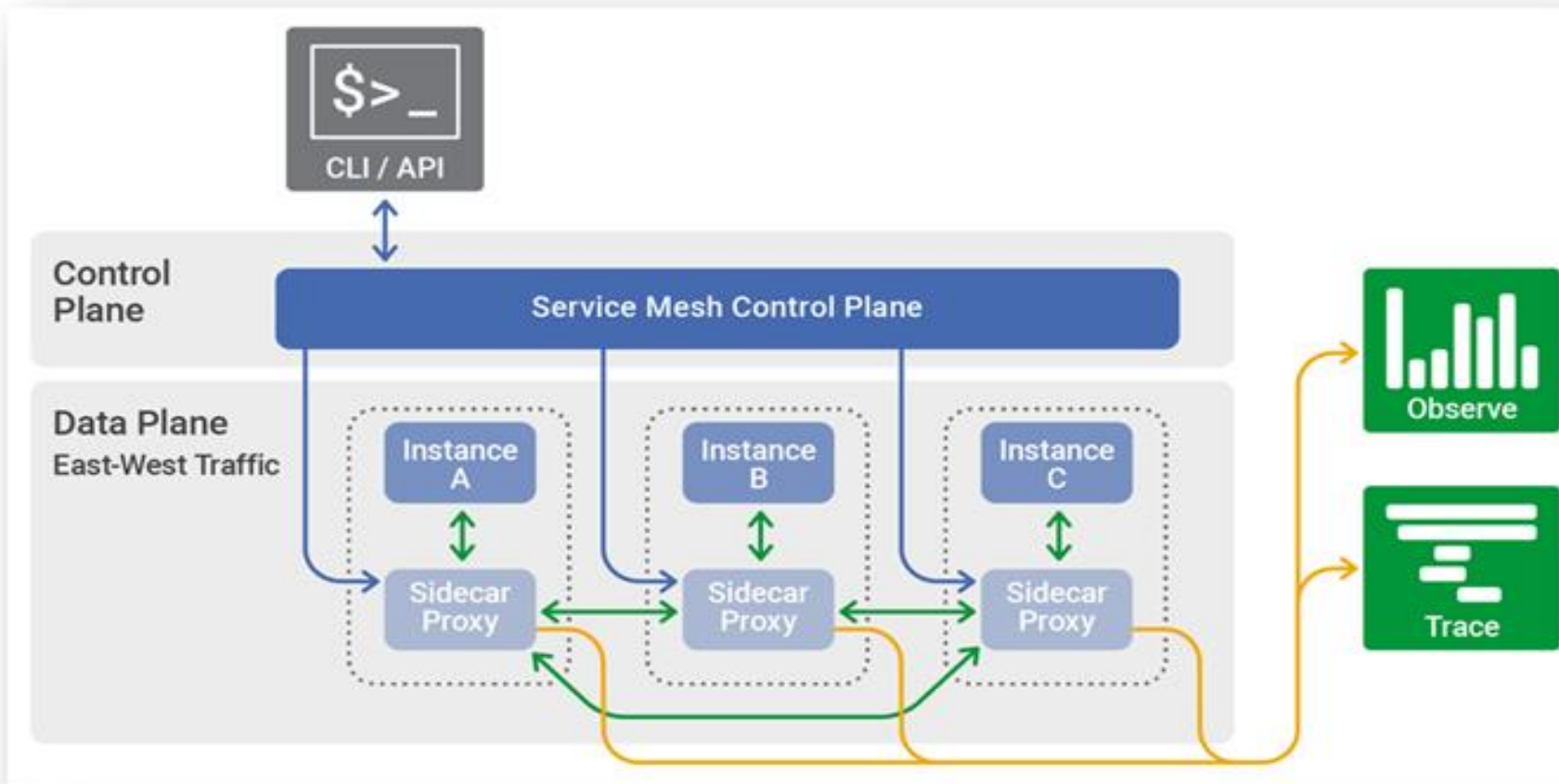
- Utilisation de proxy sidecar unique, pour chaque service.
- Implémentation en parallèle des services.
- Toutes les informations qui entrent et sortent du service passent par le proxy.
- Cela permet un accès total à l'information:
(Masse de données, temps de transmission, erreurs...)



Structure: Première approche



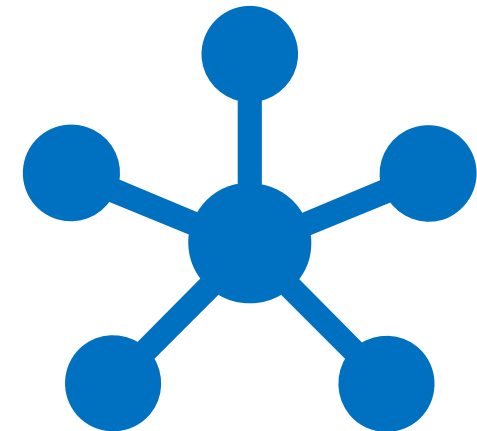
Structure seconde approche



- Un maillage de service est divisé en deux couches:
 - Couche de données
 - Couche de contrôle

Le plan de données

- Son rôle = Collecter des informations pour qu'elles soient traitées
- Ensemble de proxies Envoy
 - Optimiser les communications
 - Gestion dynamique des configurations
 - Gère la découverte de services
 - Un fonctionnement qu'importe le type et le langage
- Mixer :
 - Récupère l'ensemble des informations proxy.
 - Communique avec le centre de contrôle
 - Envoie les données aux solutions de monitoring

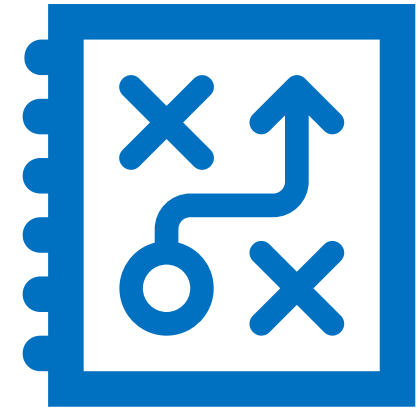


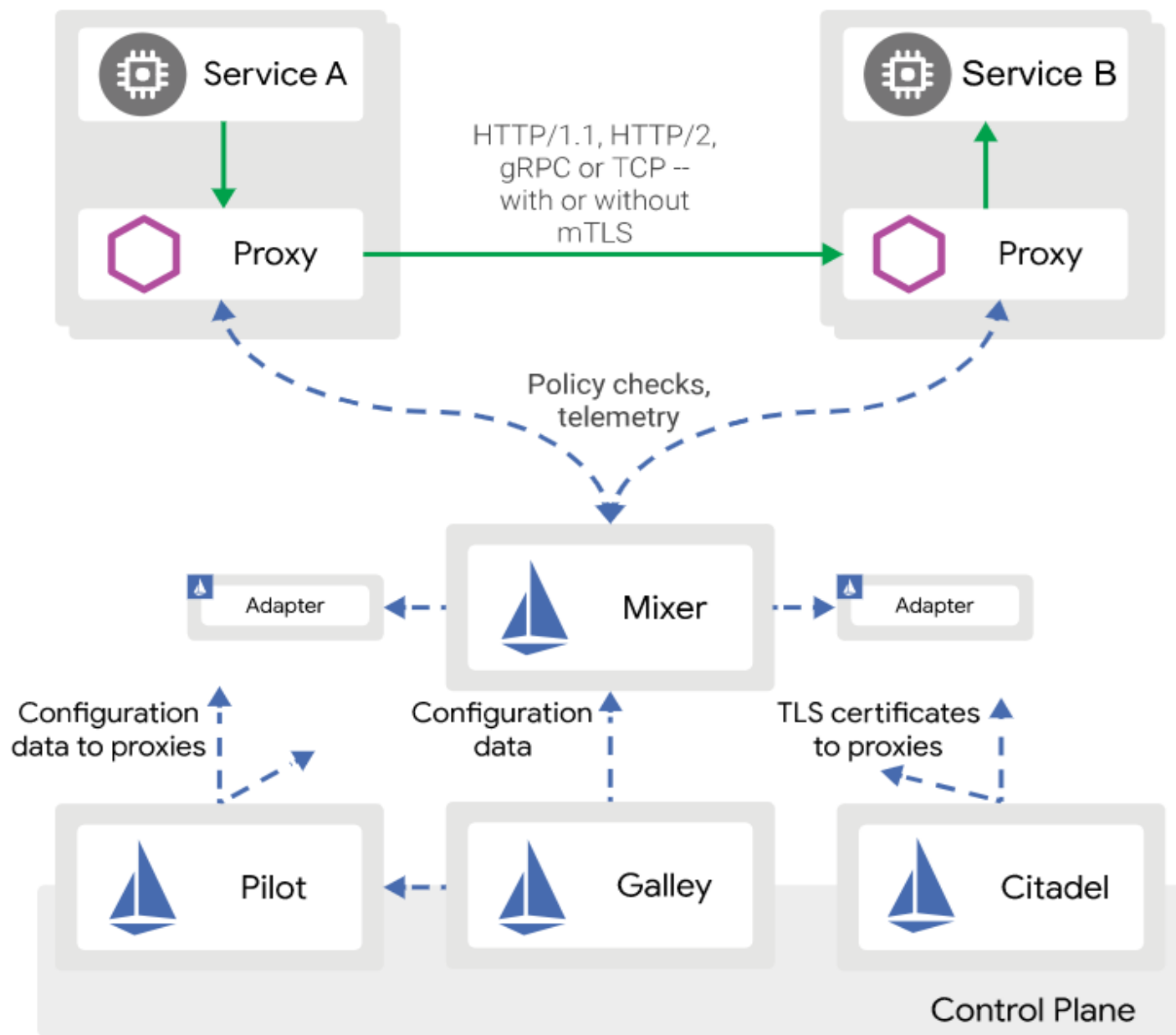
Le plan de contrôle

- Son rôle : Définir les règles appliquées au proxys qui gère la totalité des communications du réseau.

Les différents composants d'Istio:

- Pilot = Permet la découverte de service
- Citadel = Aspect sécurité d'Istio
- Galley = Distribution des configuration proxies

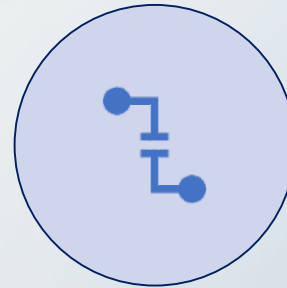




Synthèse de la partie présentation



Les maillages de services permettent de faciliter la gestion d'un réseau de microservices.



Les services mesh sont divisés en deux couches : Couches de données et couches de contrôles



Istio la solution la plus réputée, se veut très simple et rapide à mettre en place.



Les maillages de services sont amenés à évoluer dans les années à venir.

Démonstration



kubernetes



ISTIO





Objectifs de la démonstration



**Architecture
utilisée**



**Installation et
utilisation d'Istio**



**Déployer une
application**



Gestion du trafic

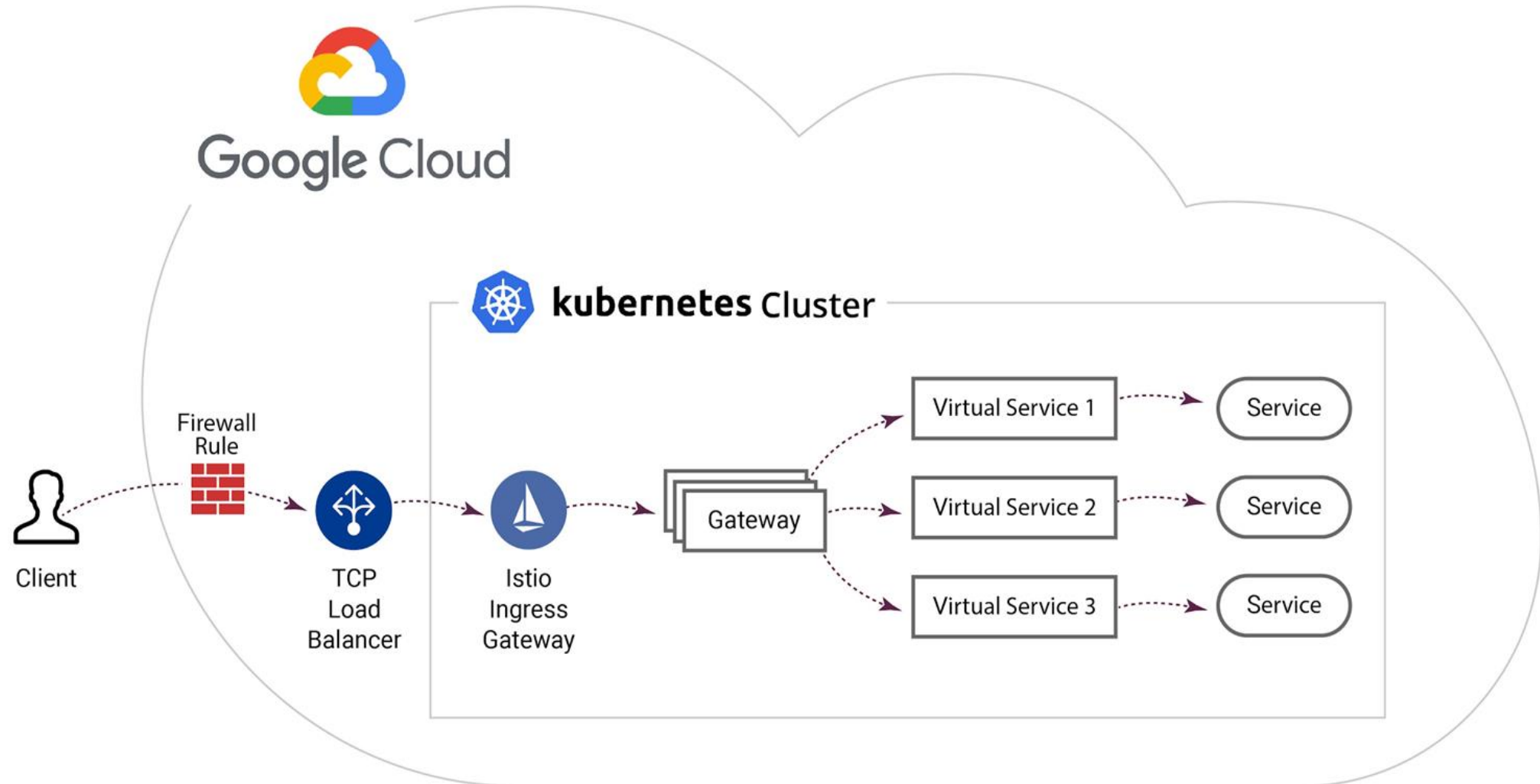
Routage intelligent
Injection de fautes



Observabilité


Surveillance
Traçabilité

Architecture utilisée



GKE (Google Kubernetes Engine)

- <https://cloud.google.com>
- Utilisation d'un cluster Kubernetes
 - 4 nœuds (1 à 4 autoscaling)
 - Kubernetes 1.15 (compatible Istio)
 - 1 vCPU et 3.75 Go RAM / nœud

<input type="checkbox"/> Nom ^	Zone	Taille du cluster	Nombre total de cœurs	Mémoire totale	Notifications	Libellés
<input type="checkbox"/>  istio-demo	us-central1-a	4	4 processeurs virtuels	15,00 Go		<div>Se connecter  </div>

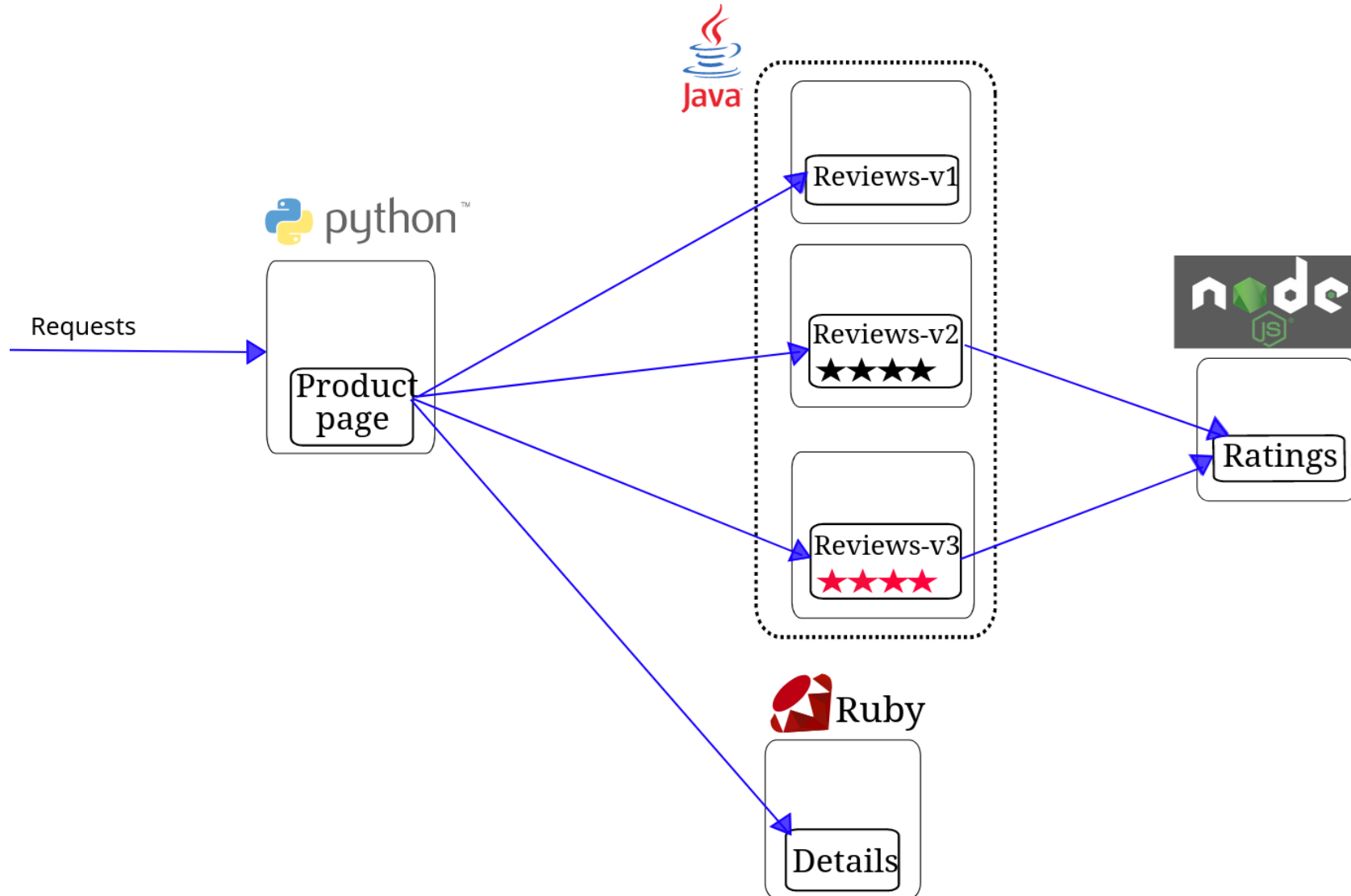
Très simple à installer et à prendre en main

- Télécharger Istio
 - `$ curl -L https://istio.io/downloadIstio | sh -`
- Vérifier la compatibilité avec le cluster
 - `$ istioctl verify-install`
- Installer un profil
 - `$ istioctl manifest apply --set profile=demo`

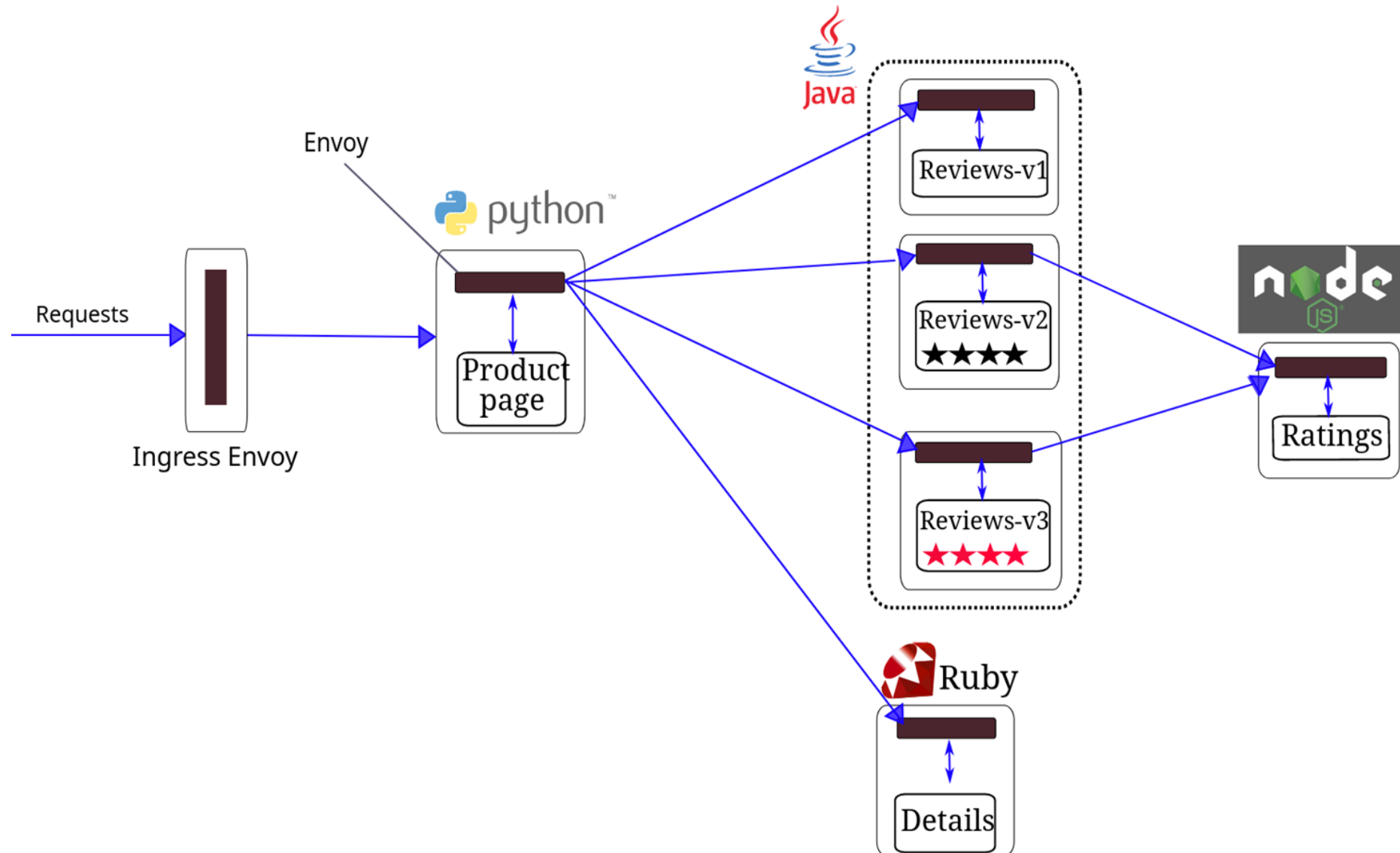
Installation d'Istio

	default	demo	minimal	sds	remote
Core components					
<code>istio-citadel</code>	X	X		X	X
<code>istio-egressgateway</code>		X			
<code>istio-galley</code>	X	X		X	
<code>istio-ingressgateway</code>	X	X		X	
<code>istio-nodeagent</code>				X	
<code>istio-pilot</code>	X	X	X	X	
<code>istio-policy</code>	X	X		X	
<code>istio-sidecar-injector</code>	X	X		X	X
<code>istio-telemetry</code>	X	X		X	
Addons					
<code>grafana</code>		X			
<code>istio-tracing</code>		X			
<code>kiali</code>		X			
<code>prometheus</code>	X	X		X	

Application : Bookinfo sans Istio



Application : Bookinfo avec Istio





Configurer la passerelle

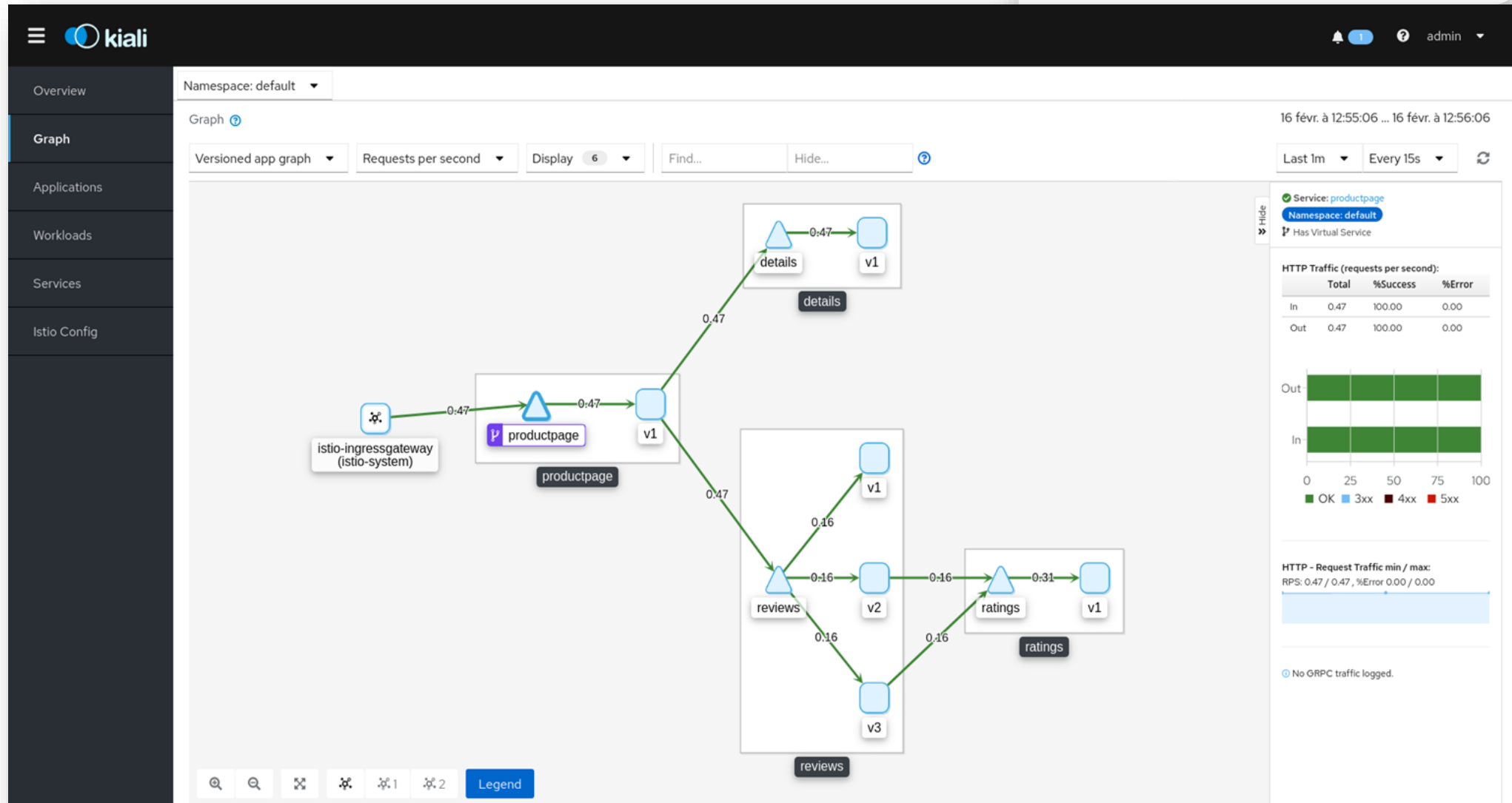
```
$ kubectl apply -f samples/bookinfo/networking/bookinfo-gateway.yaml
```

```
$ kubectl apply -f samples/bookinfo/networking/destination-rule-all.yaml
```

```
$ kubectl get service istio-ingressgateway -n istio-system
```

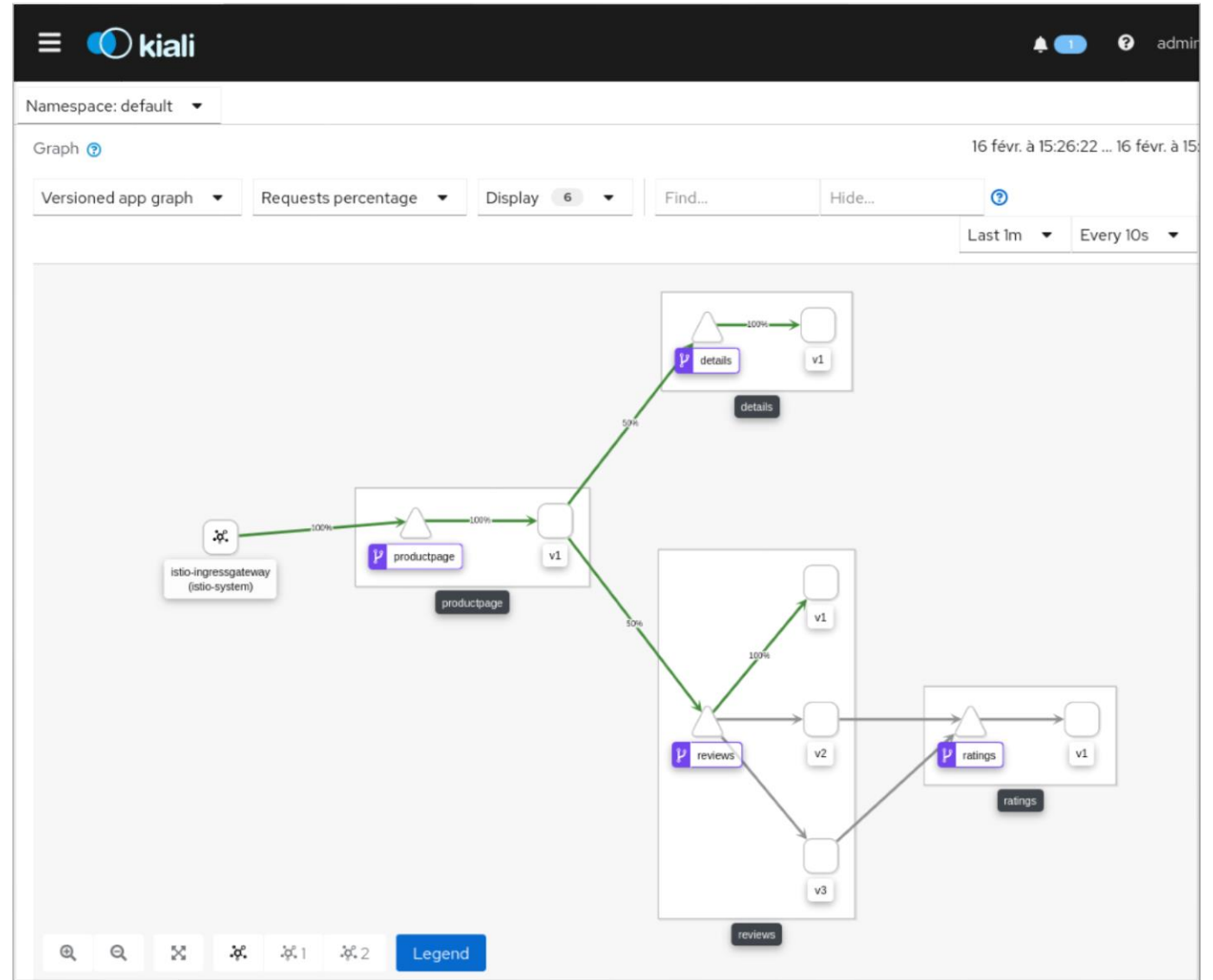
<http://34.67.4.59/productpage>

Visualiser le maillage



Modification du routage

```
$ kubectl apply -f  
samples/bookinfo/networki  
ng/virtual-service-all-  
v1.yaml
```



Injection de fautes

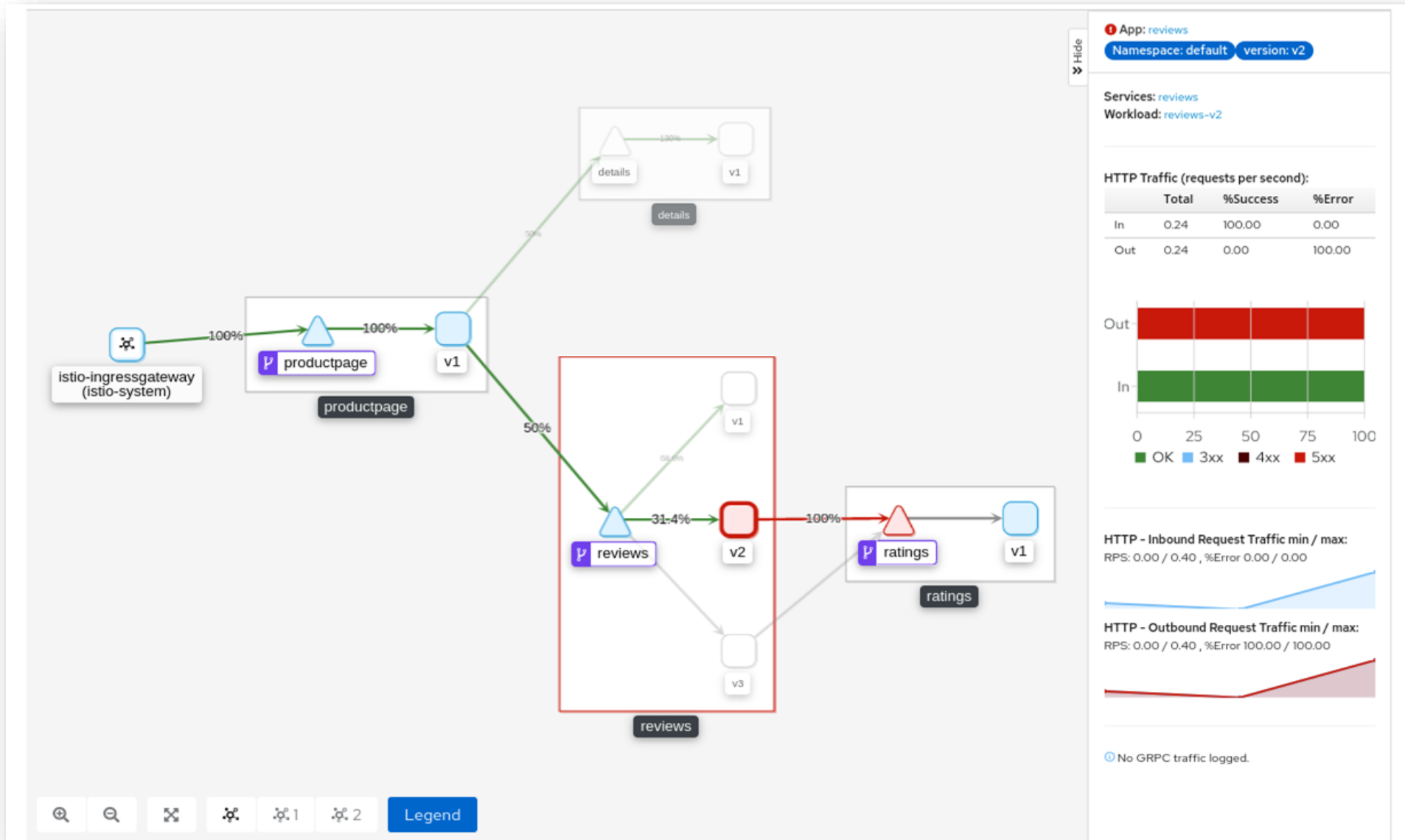
Redirection du trafic vers reviews-v2 si user “jason”

```
$ kubectl apply -f  
samples/bookinfo/networking/virtual-  
service-reviews-test-v2.yaml
```

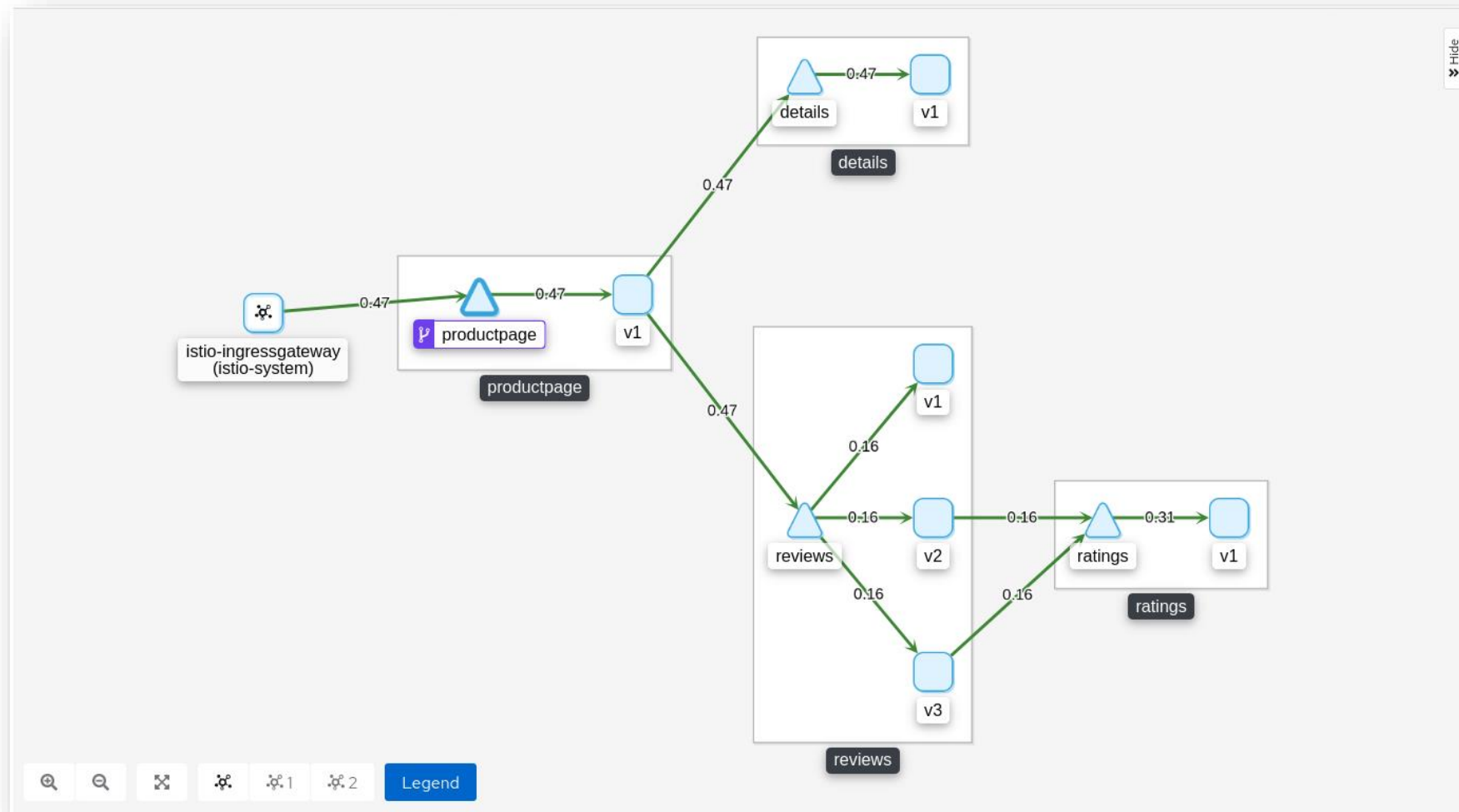
Faute (HTTP 500) pour le user “jason”

```
$ kubectl apply -f  
samples/bookinfo/networking/virtual-  
service-ratings-test-abort.yaml
```

Visualisation des fautes



Revenir à
l'état initial

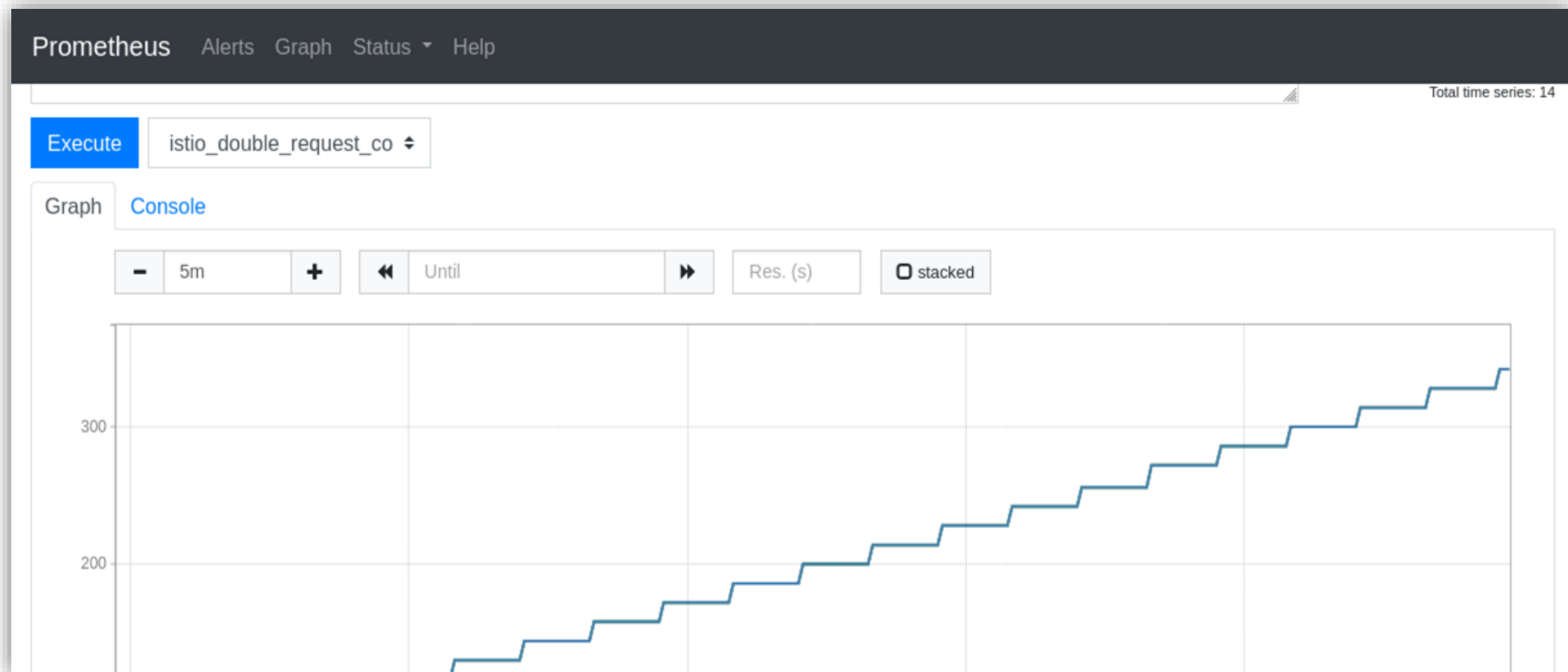


```
$ kubectl delete -f
samples/bookinfo/networking/virtual-service-
all-v1.yaml
```

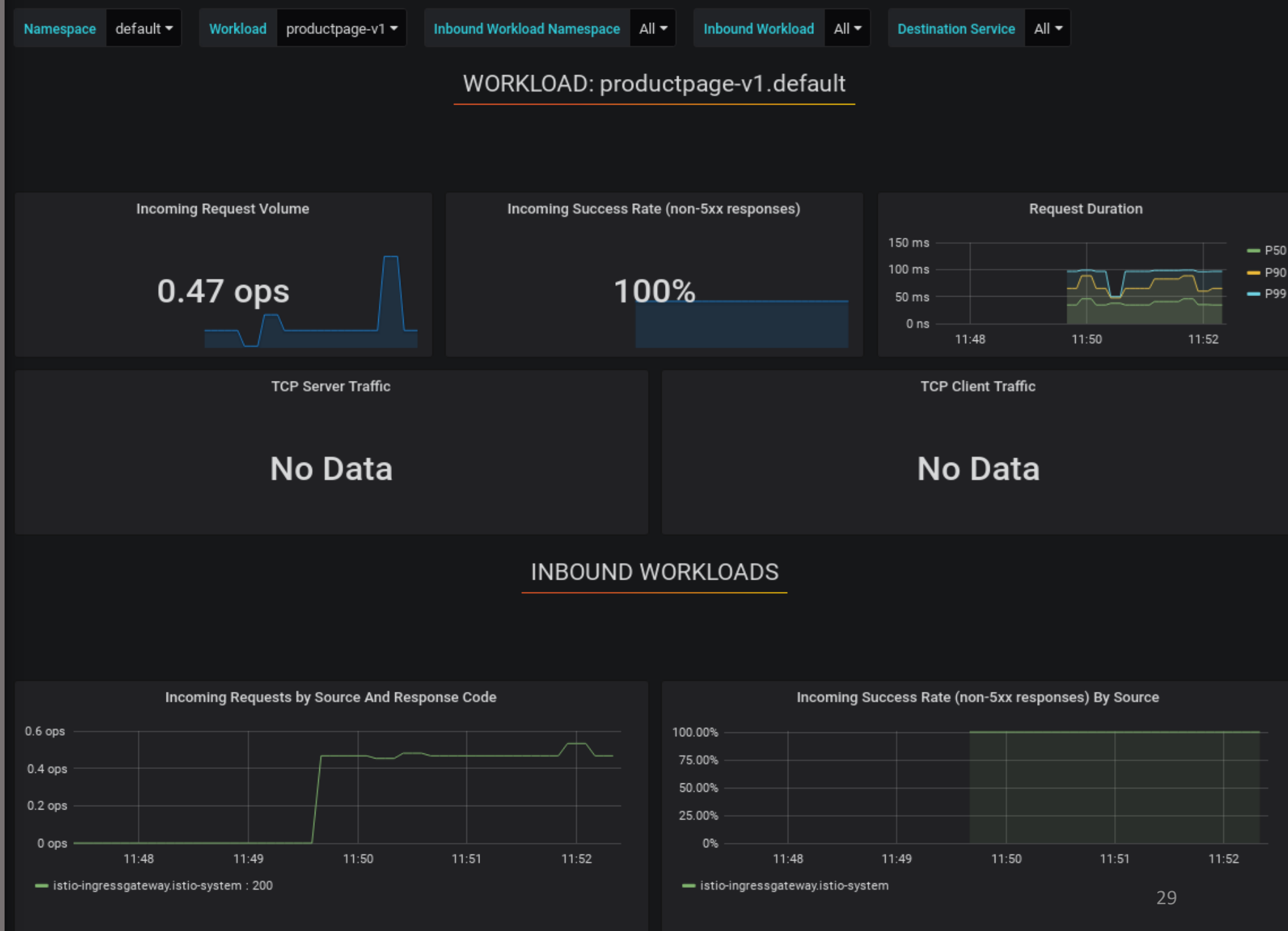
Surveillance (Prometheus)



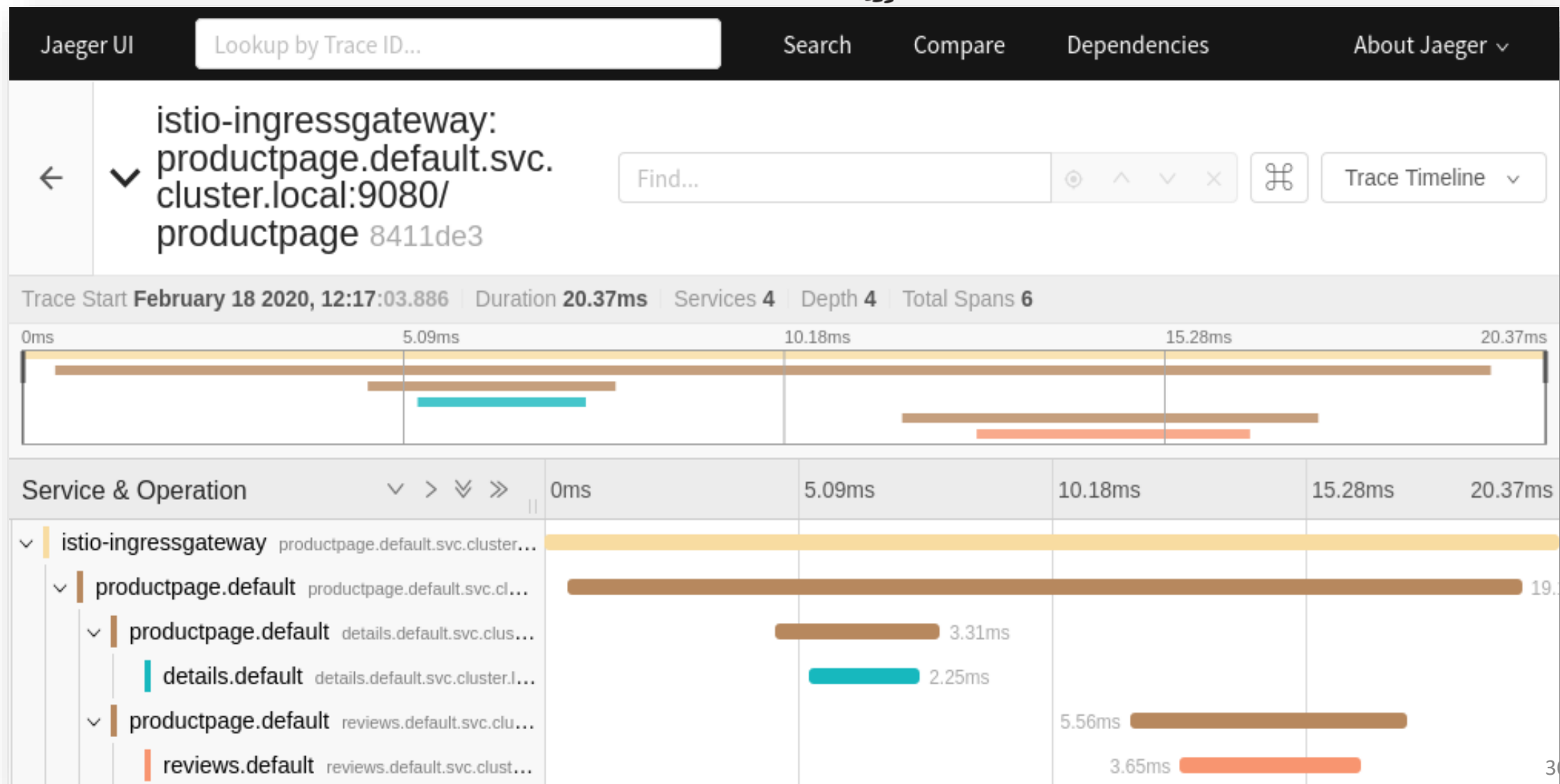
```
$ kubectl apply -f  
samples/bookinfo/telemetry/metrics.yaml
```



Surveillance (Grafana)



Traçabilidade (Jaeger)





Conclusion

- Répondre aux exigences des microservices :
 - **Gestion du trafic** (routage, fautes, disjoncteurs, délais)
 - **Observabilité** (métriques, logs, traces)
 - **Sécurité** (authentifications mTLS, autorisations)
 - **Politiques** (contrôles, refus, listes blanches/noires)
- Qui utilise Istio en production ?

Pour aller
plus loin



Tutoriel :
<https://github.com/Antoine-Auffret/service-mesh-istio-tutorial>



Documentation Istio :
<https://istio.io/docs>



Google Cloud Istio :
<https://cloud.google.com/istio>

Merci de
votre
attention

Des questions ?

