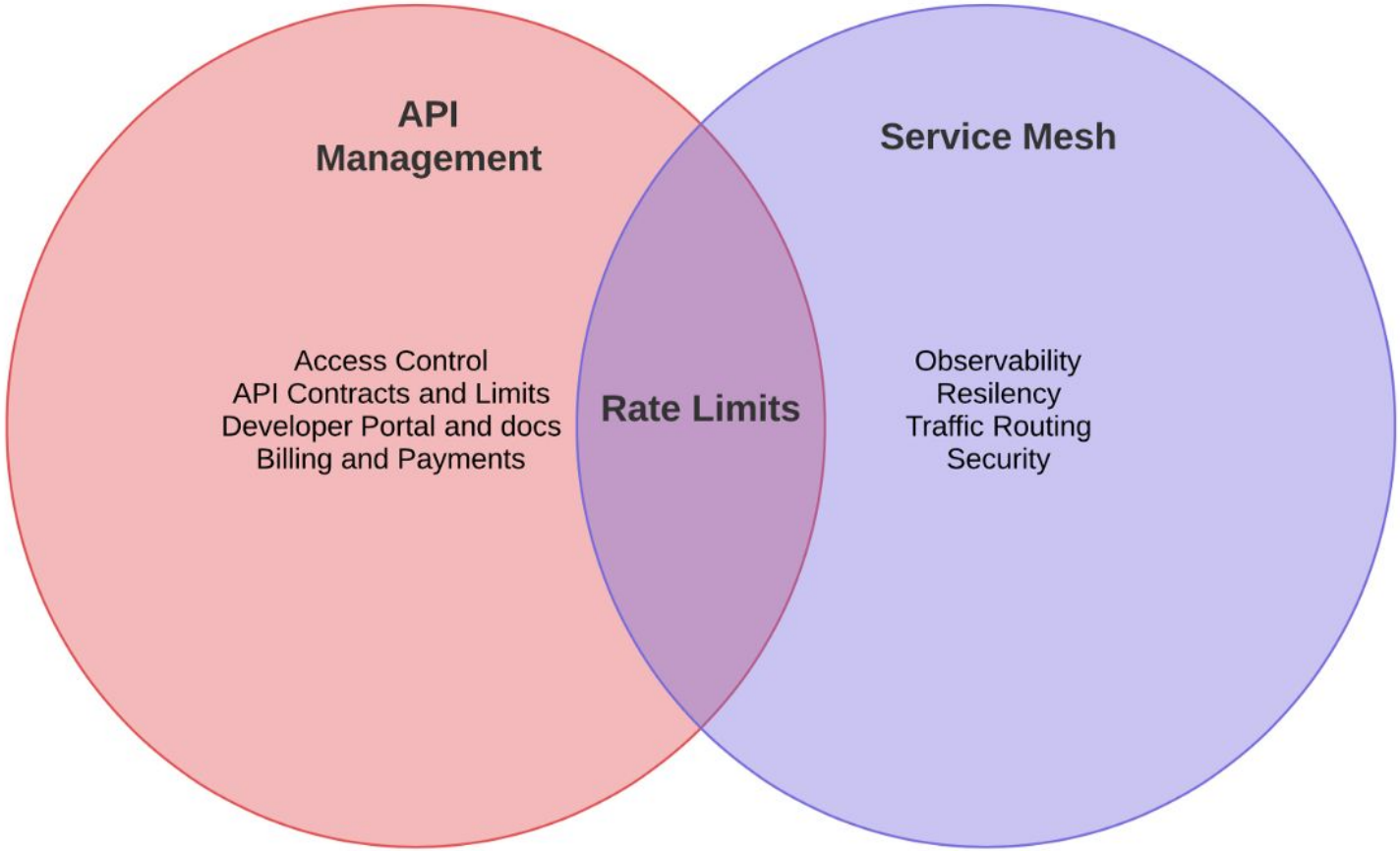
ISTIO & KIALI

..:: Du Microservice au Service Mesh ::..

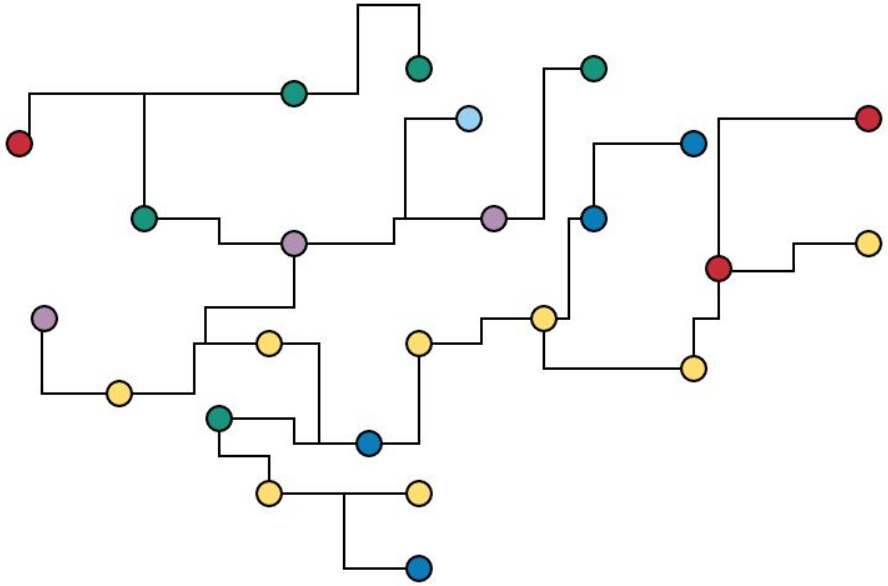
ISTIO

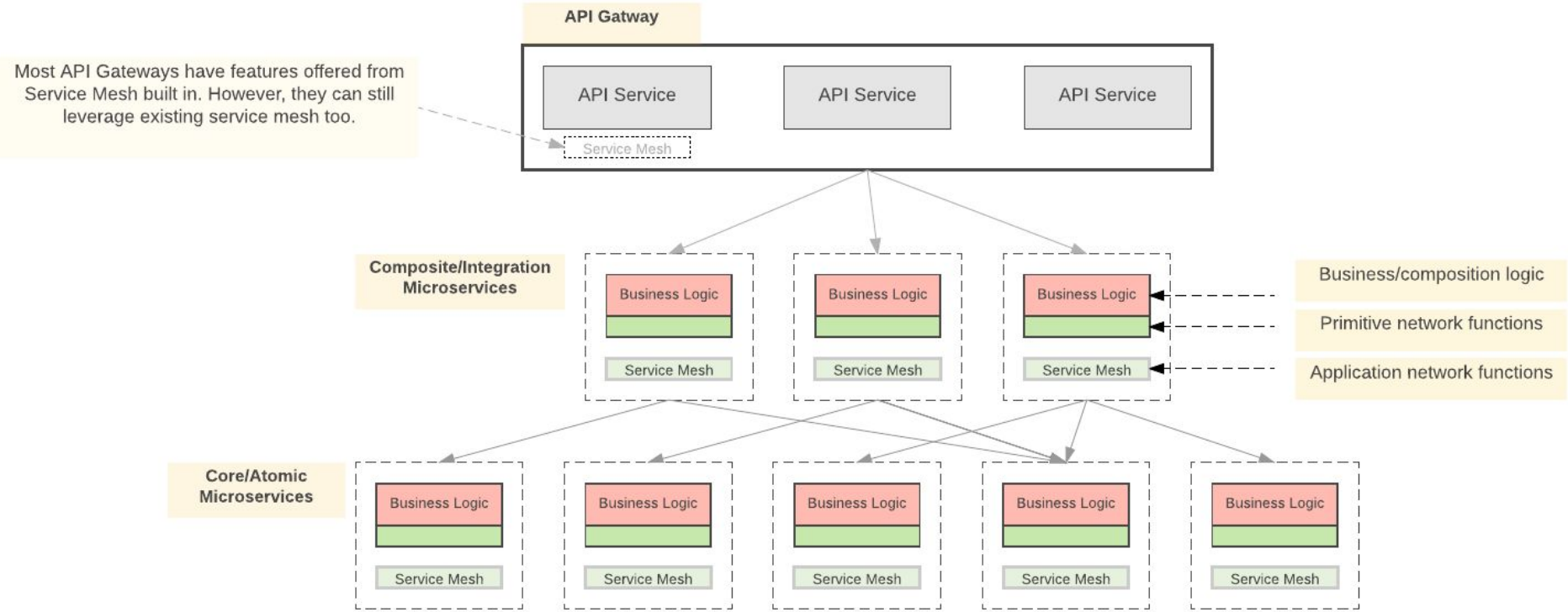
Powered by **Google, IBM, Lyft**
and many others ...

Free, Open-Source
https://istio.io/

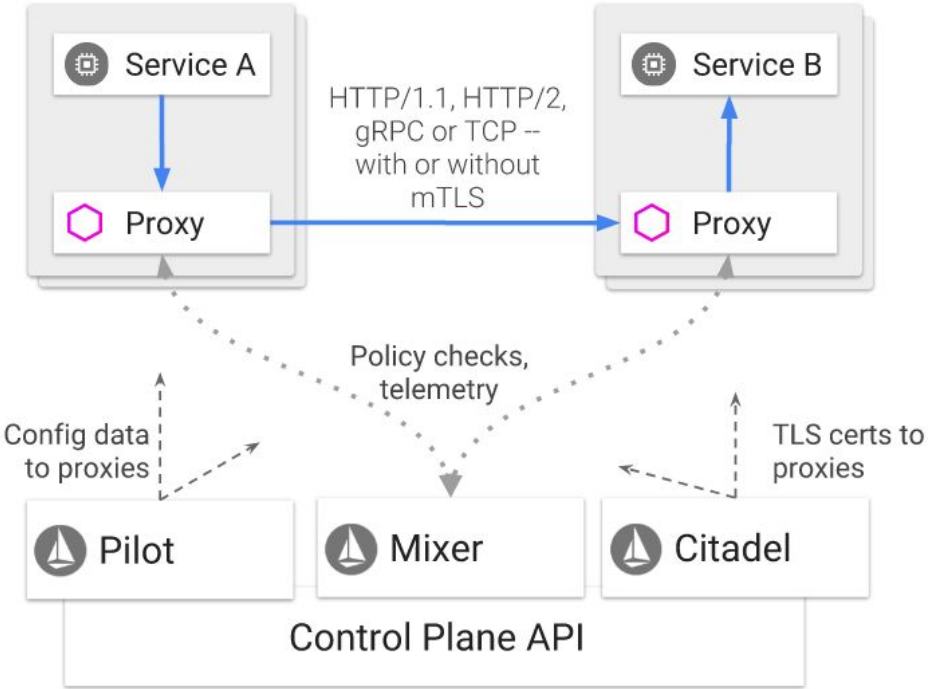

API
Management
Access Control
API Contracts and Limits
Developer Portal and docs
Billing and Payments
Rate Limits
Service Mesh
Observability
Resiliency
Traffic Routing
Security


Client
API Gateway
API




API Gatway
Most API Gateways have features offered from Service Mesh built in. However, they can still leverage existing service mesh too.
API Service
API Service
API Service
Service Mesh
Composite/Integration Microservices
Business Logic
Service Mesh
Business Logic
Service Mesh
Business Logic
Service Mesh
Business/composition logic
Primitive network functions
Application network functions
Core/Atomic Microservices
Business Logic
Service Mesh
Business Logic
Service Mesh
Business Logic
Service Mesh
Business Logic
Service Mesh
Business Logic
Service Mesh

SIDE CAR PROXY PATTERN



Concepts

- Sidecar injection
- Gateway
- VirtualService
- DestinationRule

Gateway

HTTP/HTTPS traffic to our service mesh

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: mesh-arena-gateway
spec:
  selector:
    istio: ingressgateway # use istio default controller
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
    hosts:
    - "*"
```

VirtualService

Defines a set of traffic routing rules to apply when a host is addressed

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: mesh-arena
spec:
  hosts:
  - "*"
  gateways:
  - mesh-arena-gateway
  http:
  - match:
    route:
    - destination:
        host: ui
        port:
          number: 8080
```

DestinationRule

Defines policies that apply to traffic intended for a service after routing has occurred

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: bookinfo-ratings
spec:
  host: ratings.prod.svc.cluster.local
  trafficPolicy:
    loadBalancer:
      simple: LEAST_CONN
```

Observability

Microservices : multiplication des sources à monitorer

- X services
- multiplié par N instances
- potentiellement éphémères
  => casse-tête

Observabilité : rendre ce système plus “observable”, quelle que soit l’échelle

Game-changer: Prometheus (K8S service discovery, scraping…)

Istio et Envoy exposent des métriques pour l’observabilité du service mesh

KIALI

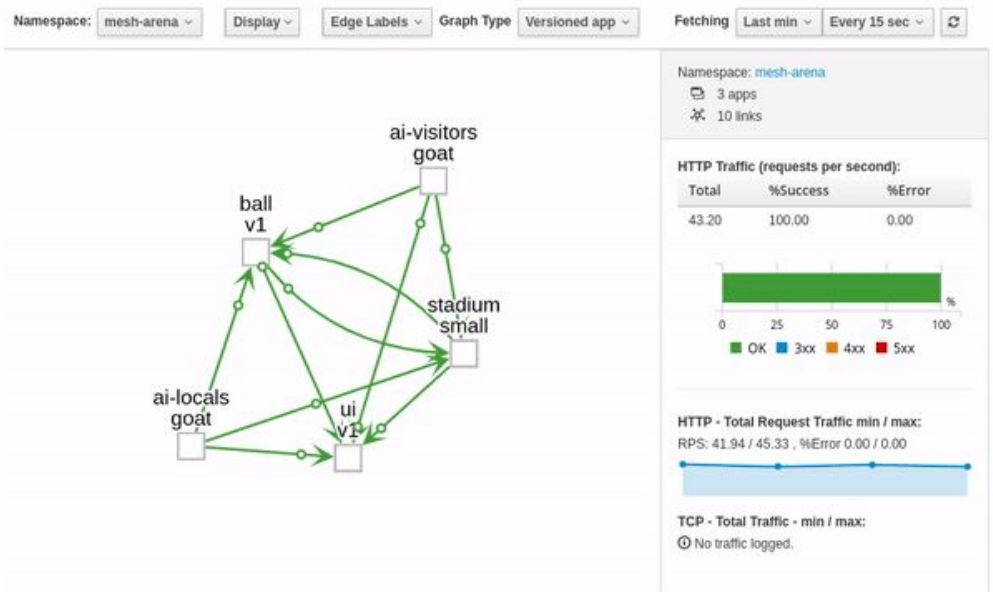
Powered by **Red Hat** with Love

Free, Open-Source
https://www.kiali.io/

Kiali

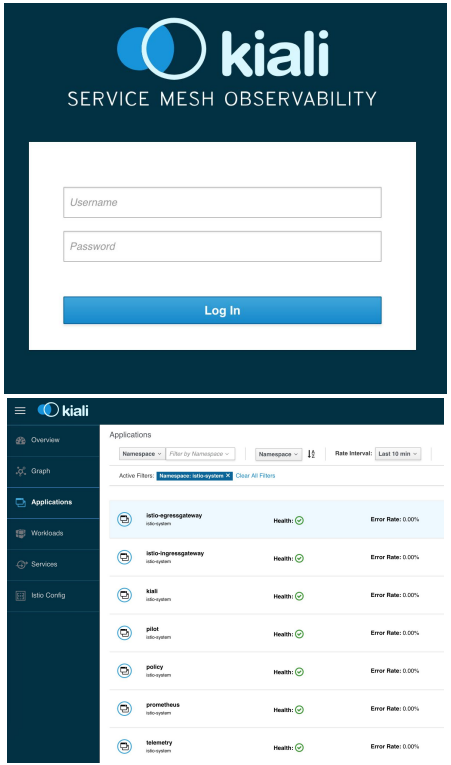
Visualisation de Service Mesh, désormais intégré à Istio.

=> Graphe, métriques, santé, validation des configurations, …



Environnement DEMO

- Minikube en local (défaut)
- Sur le cloud GKE ou AKS
- Sur Openshift
- **Sur Digital Ocean provisionné avec Rancher 2**



Présentation du jeu

Plusieurs microservices

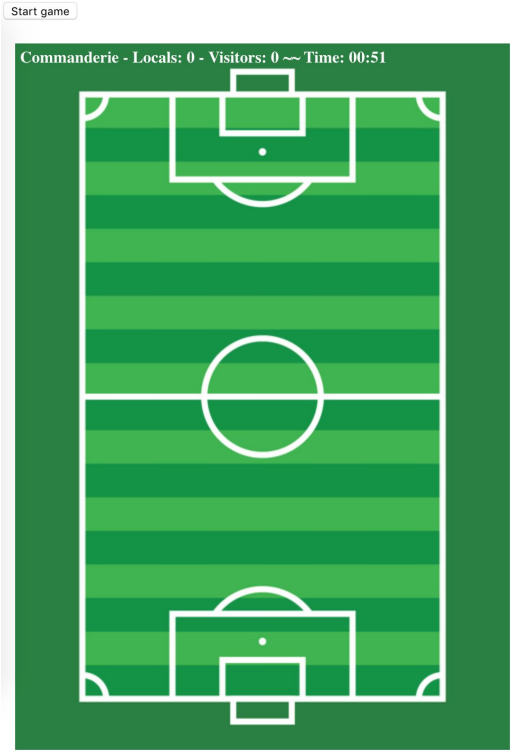
- Le stade en différentes tailles
- Chaque joueur
- Chaque ballon

On abordera certaines fonctionnalités d'Istio :

- Le routing intelligent
- Le shadowing
- Du circuit breaking (outlier detection)

Déploiement d'un service avec Istio :

```
kubectl apply -f <(istioctl kube-inject -f ./services/stadium/Deployment-Smaller.yml)
kubectl create -f ./services/stadium/Service.yml
```



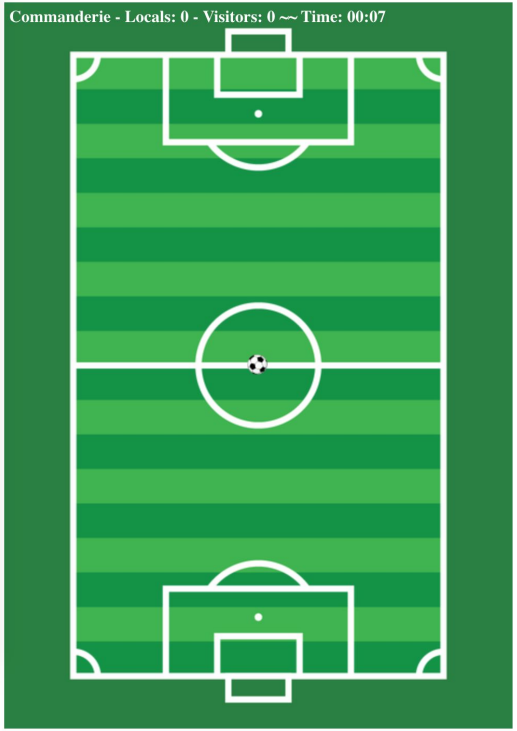
VERT.X

Ballon simple

kubectl apply -f <(istioctl kube-inject -f ./services/ball/Deployment.yml)

kubectl apply -f ./services/ball/Service.yml

```
 nicolas@macbook-pro-de-nicolas  ~/software/demo-mesh-arena  master  kubectl get po

NAME                                READY     STATUS            RESTARTS   AGE
ball-d46999f99-bmttt                0/2       PodInitializing   0          8s
stadium-small-667d48d7df-84rgw      2/2       Running           0          13m
ui-79b48cb6d-nxl7d                  2/2       Running           0          17m
 nicolas@macbook-pro-de-nicolas  ~/software/demo-mesh-arena  master  kubectl get po

NAME                                READY     STATUS    RESTARTS   AGE
ball-d46999f99-bmttt                1/2       Running   0          15s
stadium-small-667d48d7df-84rgw      2/2       Running   0          13m
ui-79b48cb6d-nxl7d                  2/2       Running   0          17m
```

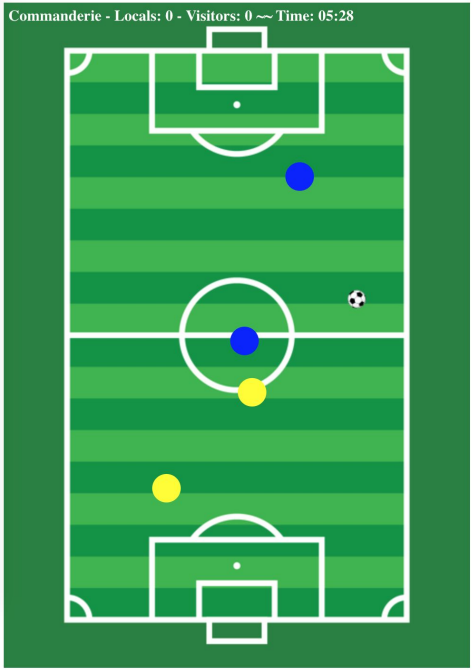


Faites entrer les joueurs

kubectl apply -f <(istioctl kube-inject -f ./services/ai/Deployment-2-locals.yml)
kubectl apply -f <(istioctl kube-inject -f ./services/ai/Deployment-2-visitors.yml)

kubectl create -f ./services/ai/Service-locals.yml
kubectl create -f ./services/ai/Service-visitors.yml

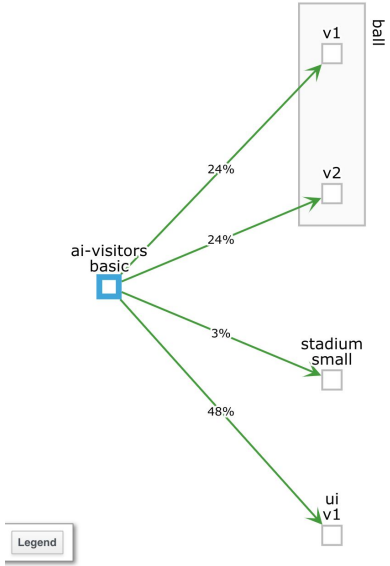
```
 nicolas@macbook-pro-de-nicolas  ~/software/demo-mesh-arena  master  kubectl get po

NAME                                READY     STATUS    RESTARTS   AGE
ai-locals-basic-568b9dd554-fmb4w    2/2       Running   0          1m
ai-locals-basic-568b9dd554-jgnxp    2/2       Running   0          1m
ai-visitors-basic-cb5dfd98d-4j4s4   2/2       Running   0          1m
ai-visitors-basic-cb5dfd98d-l4rcm   2/2       Running   0          1m
ball-d46999f99-bmttt                2/2       Running   0          22m
stadium-small-667d48d7df-84rgw      2/2       Running   0          35m
ui-79b48cb6d-nxl7d                  2/2       Running   0          39m
```

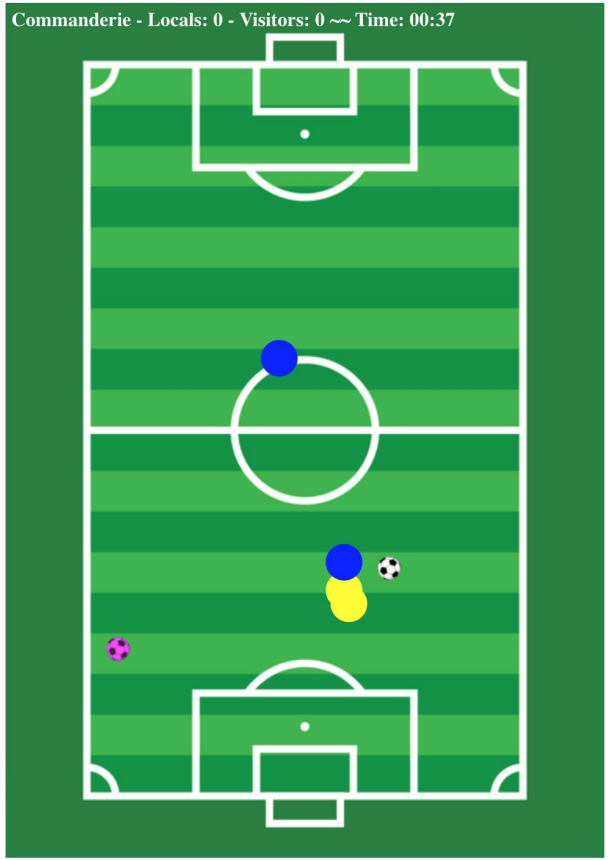


Second ballon

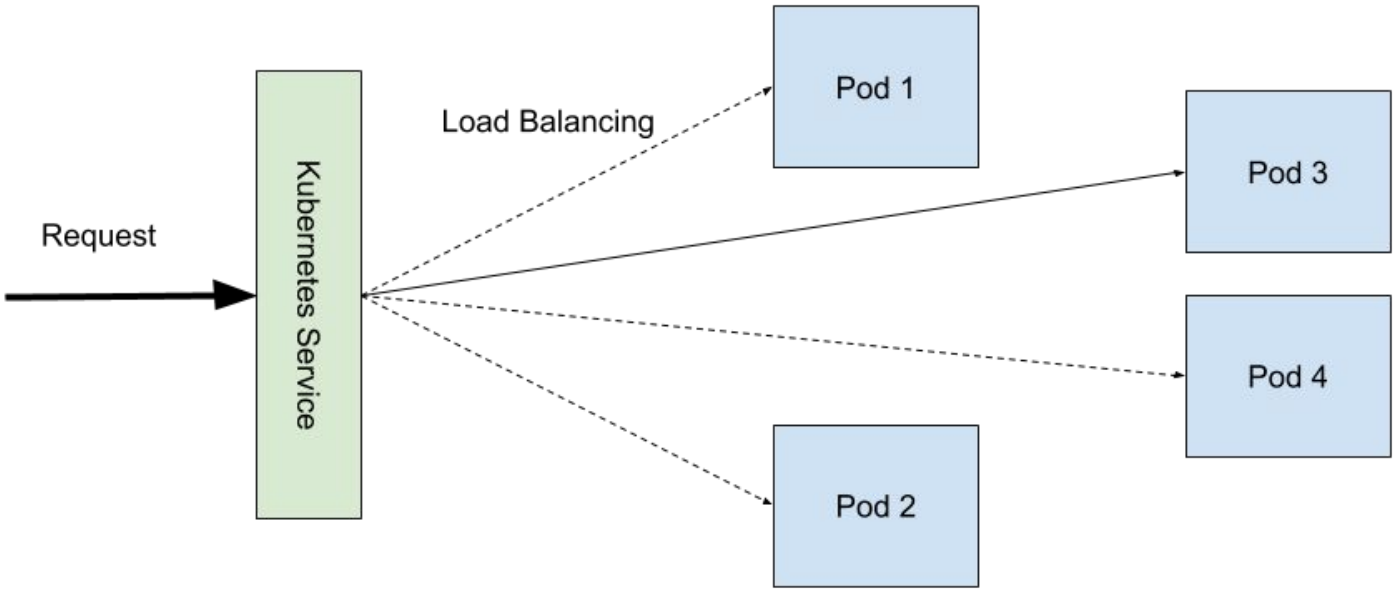
kubectl apply -f <(istioctl kube-inject -f ./services/ball/Deployment-v2.yml)

Ca court après deux ballons de façon égale





Second ballon (explication)



On pondère les ballons

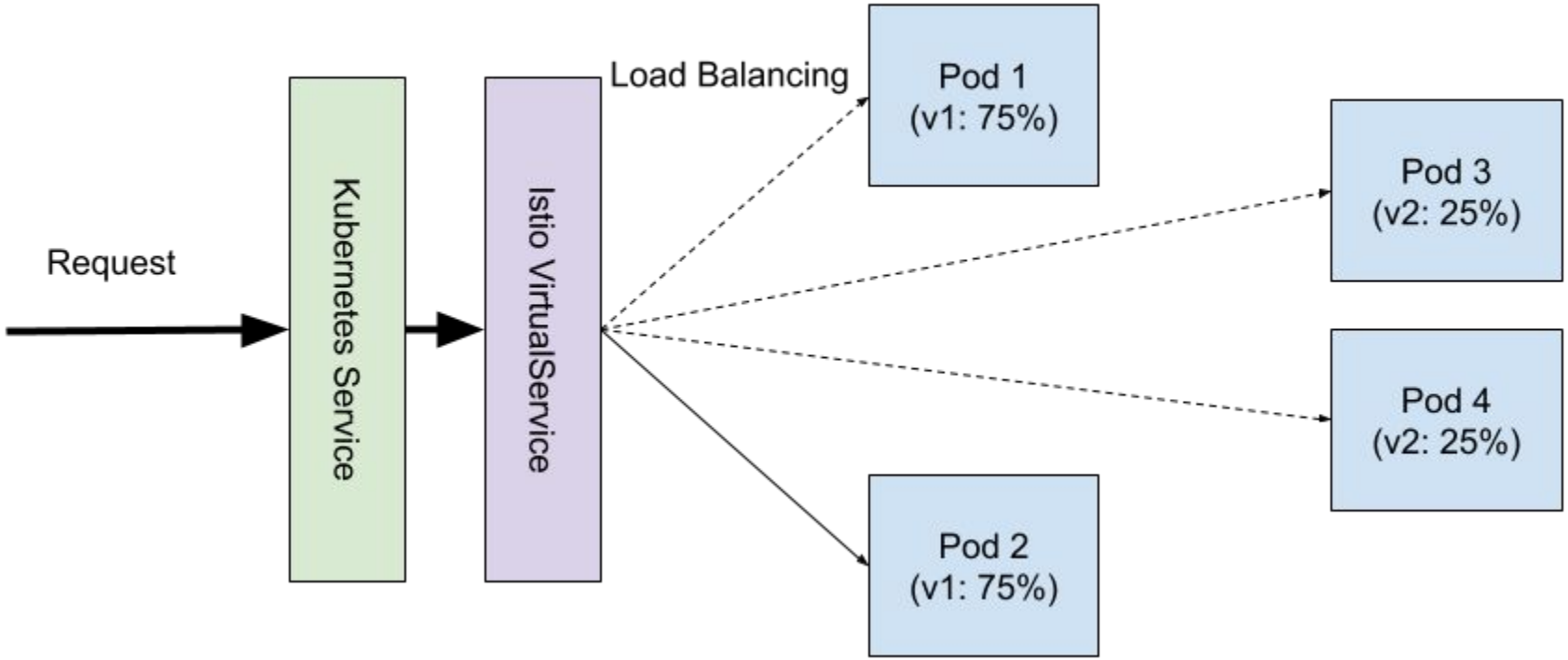
istioctl create -f ./services/ball/destrule.yml
istioctl create -f ./services/ball/virtualservice-75-25.yml

Via un VirtualService on pondère à 75 / 25

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: ball-dr
spec:
  host: ball
  subsets:
  - name: ball-v1
    labels:
      version: v1
  - name: ball-v2
    labels:
      version: v2
```

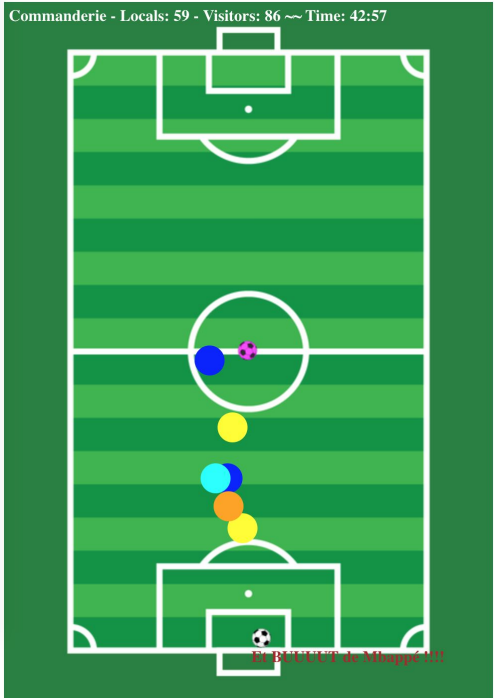
```yaml
# Scenario 1: 75-25
---
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: ball-vs
spec:
  hosts:
  - ball
  http:
  - route:
    - destination:
        host: ball
        subset: ball-v1
      weight: 75
    - destination:
        host: ball
        subset: ball-v2
      weight: 25
```

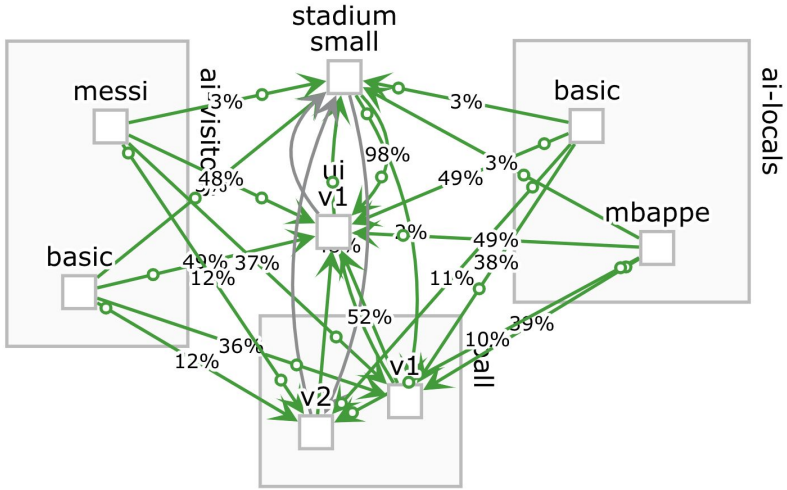
On pondère les ballons (explication)



On fait rentrer de nouveaux joueurs

kubectl apply -f <(istioctl kube-inject -f ./services/ai/Deployment-Messi.yml)
kubectl apply -f <(istioctl kube-inject -f ./services/ai/Deployment-Mbappe.yml)



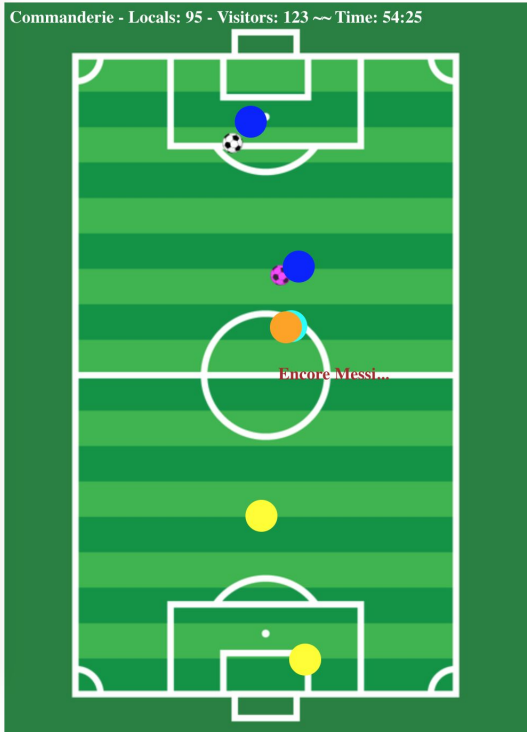


Chacun son ballon

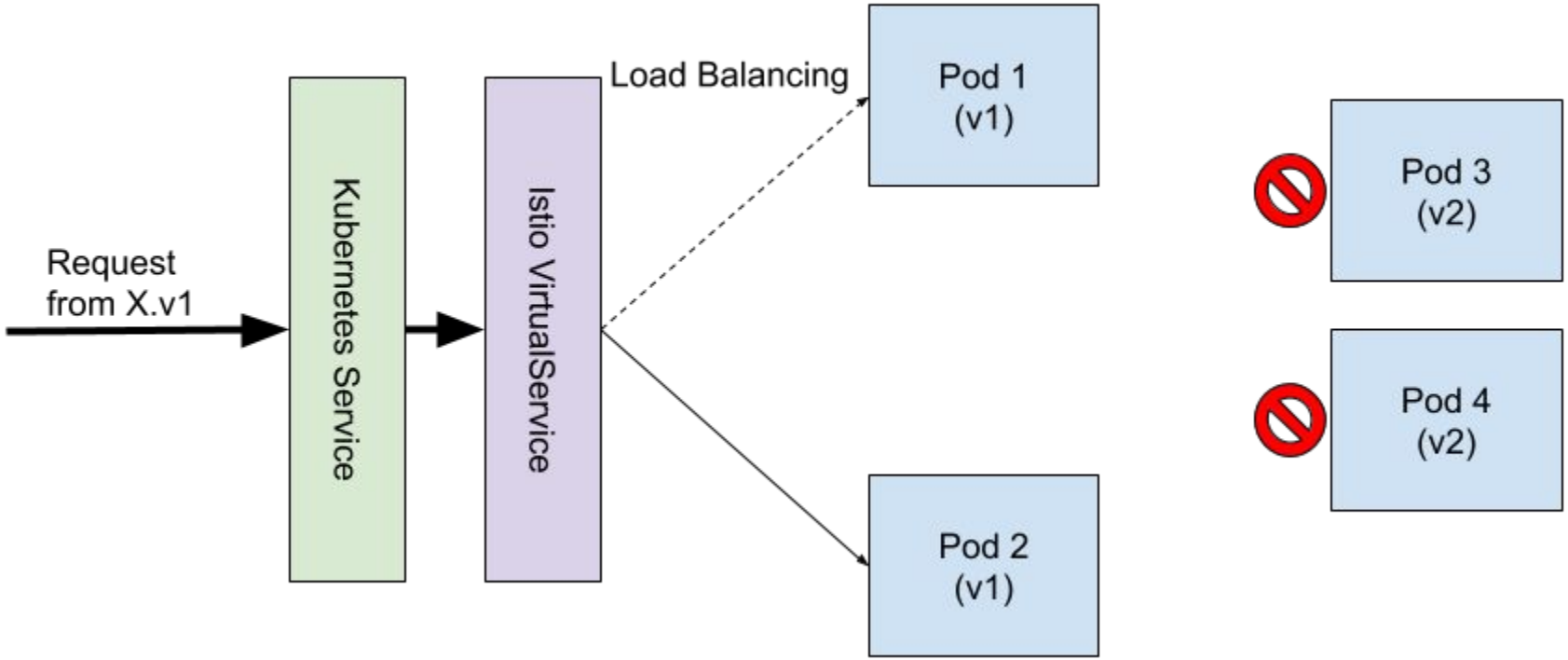
istioctl replace -f ./services/ball/virtualservice-by-label.yml

Les deux bons joueurs ont leur propre
Les autres chèvres ont le leur.

```
# Scenario 2: By label
---
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: ball-vs
spec:
  hosts:
  - ball
  http:
  - match:
    - sourceLabels:
        version: basic
    route:
    - destination:
        host: ball
        subset: ball-v1
  - route:
    - destination:
        host: ball
        subset: ball-v2
```



Chacun son ballon (explication)



Un ballon se crève

On fait rentrer au vestiaire Messi et MBappé
kubectl delete -f ./services/ai/Deployment-Messi.yml
kubectl delete -f ./services/ai/Deployment-Mbappe.yml

Plus de filtrage au niveau du ballon
istioctl delete -f ./services/ball/virtualservice-by-label.yml
Les joueurs hésitent équitablement entre les deux ballons

On crève le second ballon v2, configuré en traffic shadowing
istioctl create -f ./services/ball/virtualservice-mirrored.yml
kubectl apply -f <(istioctl kube-inject -f ./services/ball/Deployment-burst.yml)

Les joueurs se concentrent sur le premier ballon, le second reçoit tout de même les requêtes

Le ballon crevé est ignoré

On supprime le traffic shadowing
istioctl delete -f ./services/ball/virtualservice-mirrored.yml
Les joueurs hésitent entre les deux ballons
Le taux d'erreurs grimpe

On introduit l' "outlier detection" (circuit breaker)
istioctl replace -f ./services/ball/destrule-outlier.yml
Les joueurs hésitent encore un peu… puis oublient le ballon crevé

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: ball-dr
spec:
  host: ball
  subsets:
  - name: ball-v1
    labels:
      version: v1
  - name: ball-v2
    labels:
      version: v2
# Scenario 3: With outlier detection
  trafficPolicy:
    outlierDetection:
      consecutiveErrors: 1
      baseEjectionTime: 10s
      maxEjectionPercent: 50
```

Questions & Réponses

- Retrouvez la démo, rejouez-la vous-même: https://github.com/jotak/demo-mesh-arena

- Twitter: @zepouet (Nicolas), @jotak (Joël)