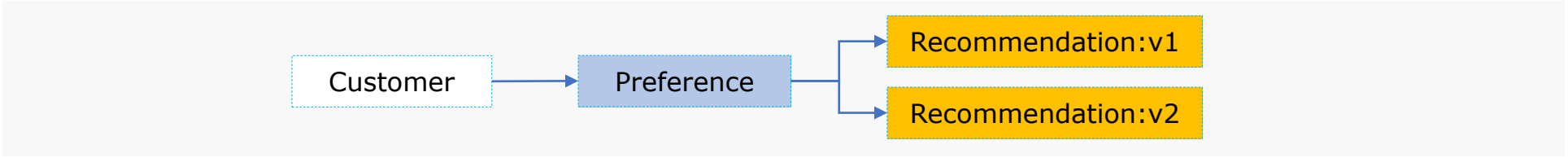
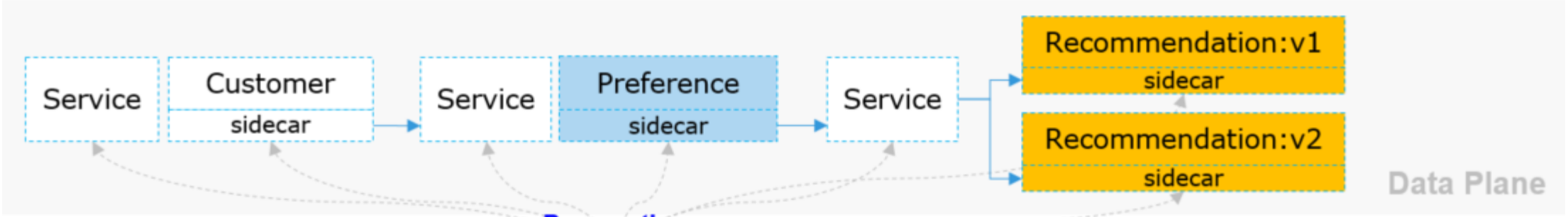
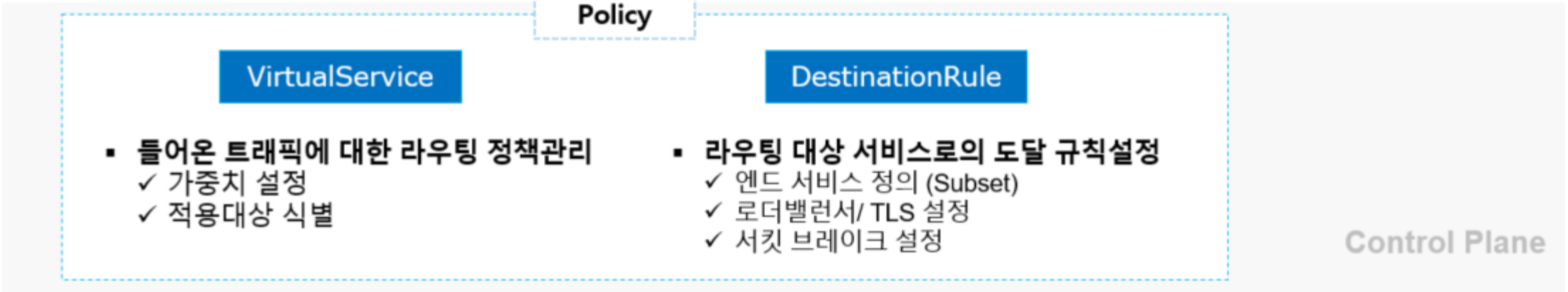
테스트 예시 서비스 흐름도





Propagation



Istio Tutorial 셋업

git clone https://github.com/redhat-developer-demos/istio-tutorial

```
root@labs-910775232:/home/project# git clone https://github.com/redhat-developer-demos/istio-tutorial
Cloning into 'istio-tutorial'...
remote: Enumerating objects: 9264, done.
remote: Counting objects: 100% (449/449), done.
remote: Compressing objects: 100% (237/237), done.
remote: Total 9264 (delta 217), reused 374 (delta 180), pack-reused 8815
Receiving objects: 100% (9264/9264), 33.02 MiB | 12.88 MiB/s, done.
Resolving deltas: 100% (5374/5374), done.
Checking out files: 100% (407/407), done.
root@labs-910775232:/home/project# cd istio-tutorial
root@labs-910775232:/home/project/istio-tutorial# kubectl create namespace tutorial
namespace/tutorial created
```

네임스페이스 생성

kubectl create namespace tutorial

Customer Service 배포 생성확인

kubectl apply –f <(istioctl kube-inject –f customer/kubernetes/Deployment.yml) –n tutorial

```
root@labs-910775232:/home/project/istio-tutorial# kubectl apply -f <(istioctl kube-inject -f customer/kubernetes/Deployment.yml) -n tutorial
serviceaccount/customer created
deployment.apps/customer created
root@labs-910775232:/home/project/istio-tutorial# kubectl get all -n tutorial
NAME                            READY   STATUS    RESTARTS   AGE
pod/customer-7894f979bb-rff9h   2/2     Running   0          27s

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/customer   1/1     1            1           28s

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/customer-7894f979bb   1         1         1       28s
```

kubectl describe pod/customer-7894f979bb-rff9h -n tutorial

```
root@labs-910775232:/home/project/istio-tutorial# kubectl describe pod/customer-7894f979bb-rff9h -n tutorial
Name:         customer-7894f979bb-rff9h
Namespace:    tutorial
Priority:     0
Node:         ip-192-168-74-130.ap-northeast-2.compute.internal/192.168.74.130
Start Time:   Fri, 18 Mar 2022 05:39:39 +0000
Labels:       app=customer
              pod-template-hash=7894f979bb
              security.istio.io/tlsMode=istio
              service.istio.io/canonical-name=customer
              service.istio.io/canonical-revision=v1
              version=v1
Annotations:  kubectl.kubernetes.io/default-container: customer
              kubectl.kubernetes.io/default-logs-container: customer
              kubernetes.io/psp: eks.privileged
              prometheus.io/path: /stats/prometheus
              prometheus.io/port: 15020
              prometheus.io/scrape: true
              sidecar.istio.io/inject: true
              sidecar.istio.io/status:
                {"initContainers":["istio-init"],"containers":["istio-proxy"],"volumes":["istio-envoy","istio-data","istio-podinfo","istio-token","istiod-...
Status:       Running
IP:           192.168.95.244
IPs:
  IP:           192.168.95.244
Controlled By:  ReplicaSet/customer-7894f979bb
Init Containers:
  istio-init:
    Container ID:  docker://740f5668ee61d543c8c49006d0f8cd77a8e65c761082257b25e2357c440b4d7d
    Image:         gcr.io/istio-release/proxyv2:1.11.3
    Image ID:      docker-pullable://gcr.io/istio-release/proxyv2@sha256:28513eb3706315b26610a53e0d66b29b09a334e3164393b9a0591f34fe47a6fd
    Port:          <none>
    Host Port:     <none>
    Args:
      istio-iptables
      -p
      15001
      -z
      15006
```

kubectl create -f customer/kubernetes/Service.yml -n tutorial

Istio Gateway 설치 및 Customer 서비스 라우팅(VirtualService) 설정

```
cat customer/kubernetes/Gateway.yml
kubectl create -f customer/kubernetes/Gateway.yml -n tutorial
```

Istio-IngressGateway 를 통한 Customer 서비스 확인

해당 EXTERNAL-IP 가 Istio Gateway 주소

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

Customer 서비스 호출 - http://(istio-ingressgateway IP)/customer

Preference, Recommendation-v1 Service 배포

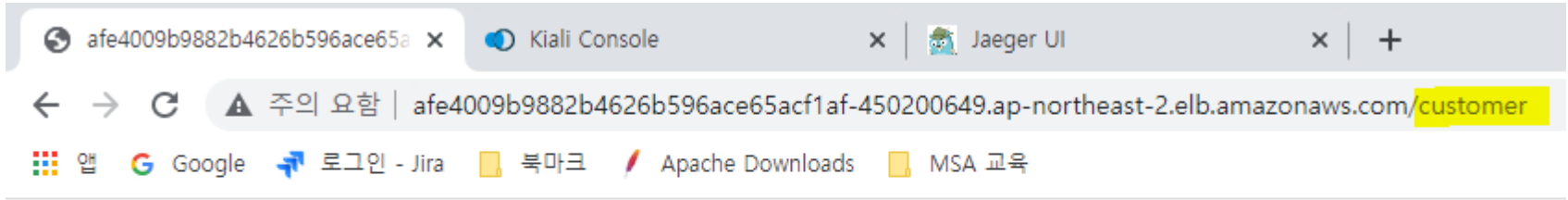
```
kubectl apply –f <(istioctl kube-inject –f preference/kubernetes/Deployment.yml) –n tutorial
kubectl create -f preference/kubernetes/Service.yml -n tutorial
kubectl apply –f <(istioctl kube-inject –f recommendation/kubernetes/Deployment.yml) –n tutorial
kubectl create -f recommendation/kubernetes/Service.yml -n tutorial
```

```
root@labs-910775232:/home/project/istio-tutorial# kubectl apply -f <(istioctl kube-inject -f preference/kubernetes/Deployment.yml) -n tutorial
serviceaccount/preference created
deployment.apps/preference-v1 created
root@labs-910775232:/home/project/istio-tutorial# kubectl create -f preference/kubernetes/Service.yml -n tutorial
service/preference created
root@labs-910775232:/home/project/istio-tutorial# kubectl apply -f <(istioctl kube-inject -f recommendation/kubernetes/Deployment.yml) -n tutorial
serviceaccount/recommendation created
deployment.apps/recommendation-v1 created
root@labs-910775232:/home/project/istio-tutorial# kubectl create -f recommendation/kubernetes/Service.yml -n tutorial
service/recommendation created
```

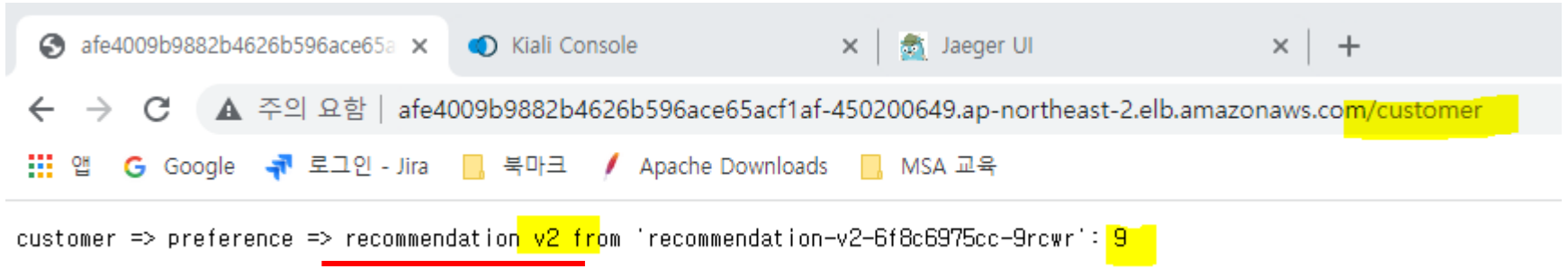
recommendation 서비스 추가 배포: v2

```
kubectl apply –f <(istioctl kube-inject –f recommendation/kubernetes/Deployment-v2.yml) –n tutorial
```

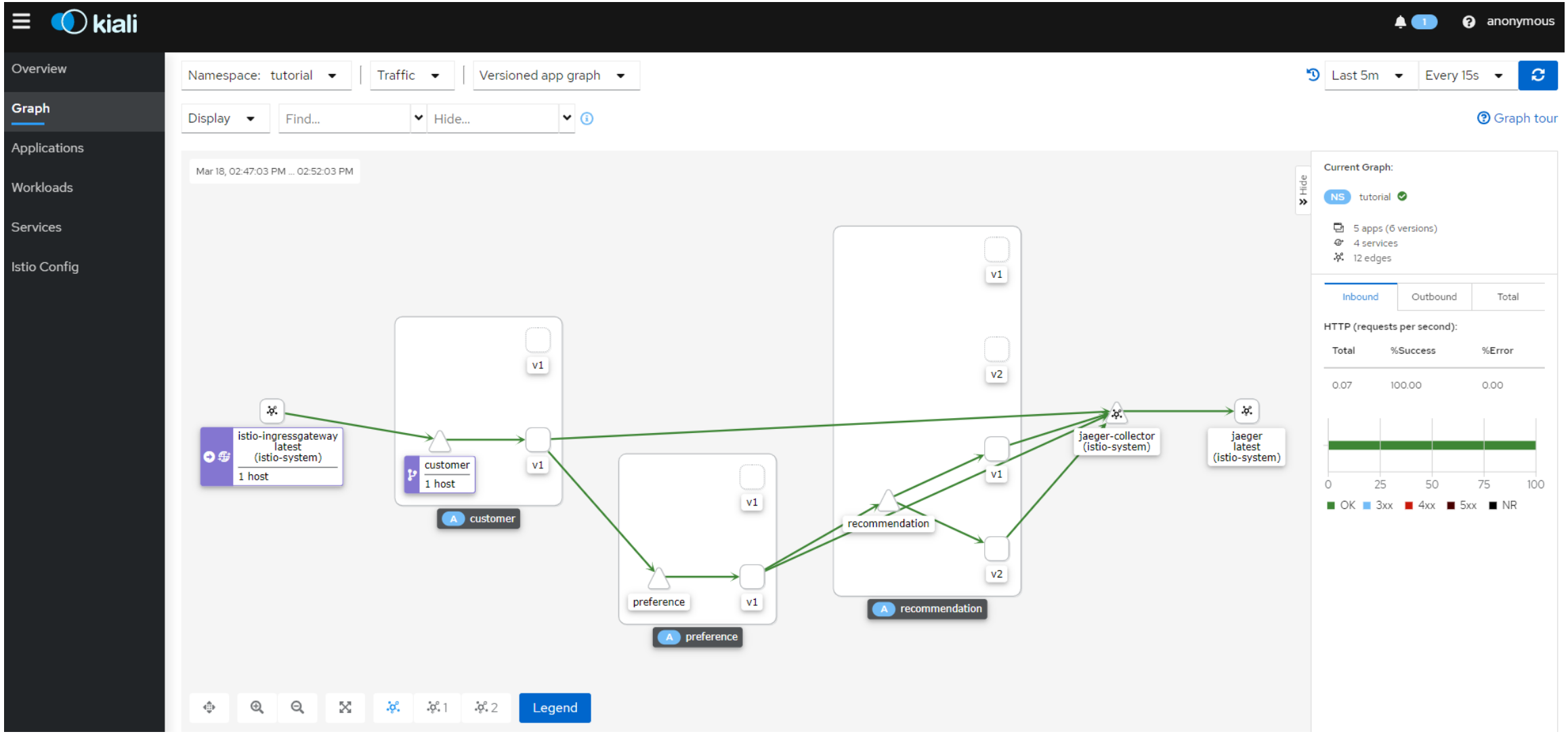
서비스 호출



Recommendation 서비스가 2개 올라가 있기 때문에 RR로 번걸아 가며 호출



Routing 결과 확인 - Kiali(Externl-IP:20001) 접속



정책(VirtualService) 설정 확인

`kubectl get VirtualService -n tutorial -o yaml`

```
root@labs-910775232:/home/project/istio-tutorial# kubectl get VirtualService -n tutorial -o yaml
apiVersion: v1
items:
- apiVersion: networking.istio.io/v1beta1
  kind: VirtualService
  metadata:
    creationTimestamp: "2022-03-18T05:45:06Z"
    generation: 1
    managedFields:
    - apiVersion: networking.istio.io/v1alpha3
      fieldsType: FieldsV1
      fieldsV1:
        f:spec:
          .: {}
          f:gateways: {}
          f:hosts: {}
          f:http: {}
      manager: kubectl
      operation: Update
      time: "2022-03-18T05:45:06Z"
    name: customer-gateway
    namespace: tutorial
    resourceVersion: "257065"
    uid: 71daf64d-ca64-41a2-bd68-3c87d57937e9
  spec:
    gateways:
    - customer-gateway
    hosts:
    - '*'
    http:
    - match:
      - uri:
          prefix: /customer
      rewrite:
        uri: /
      route:
      - destination:
          host: customer
          port:
            number: 8080
kind: List
metadata:
  resourceVersion: ""
  selfLink: ""
```

정책(DestinationRule) 설정 확인

`kubectl get DestinationRule -n tutorial -o yaml`

```
root@labs-910775232:/home/project/istio-tutorial# kubectl get DestinationRule -n tutorial -o yaml
apiVersion: v1
items: []
kind: List
metadata:
  resourceVersion: ""
  selfLink: ""
```

가중치 기반 스마트 라우팅

recommendation 서비스 v1 의 가중치를 100 으로 변경

kubectl replace -f istiofiles/virtual-service-recommendation-v1.yml -n tutorial

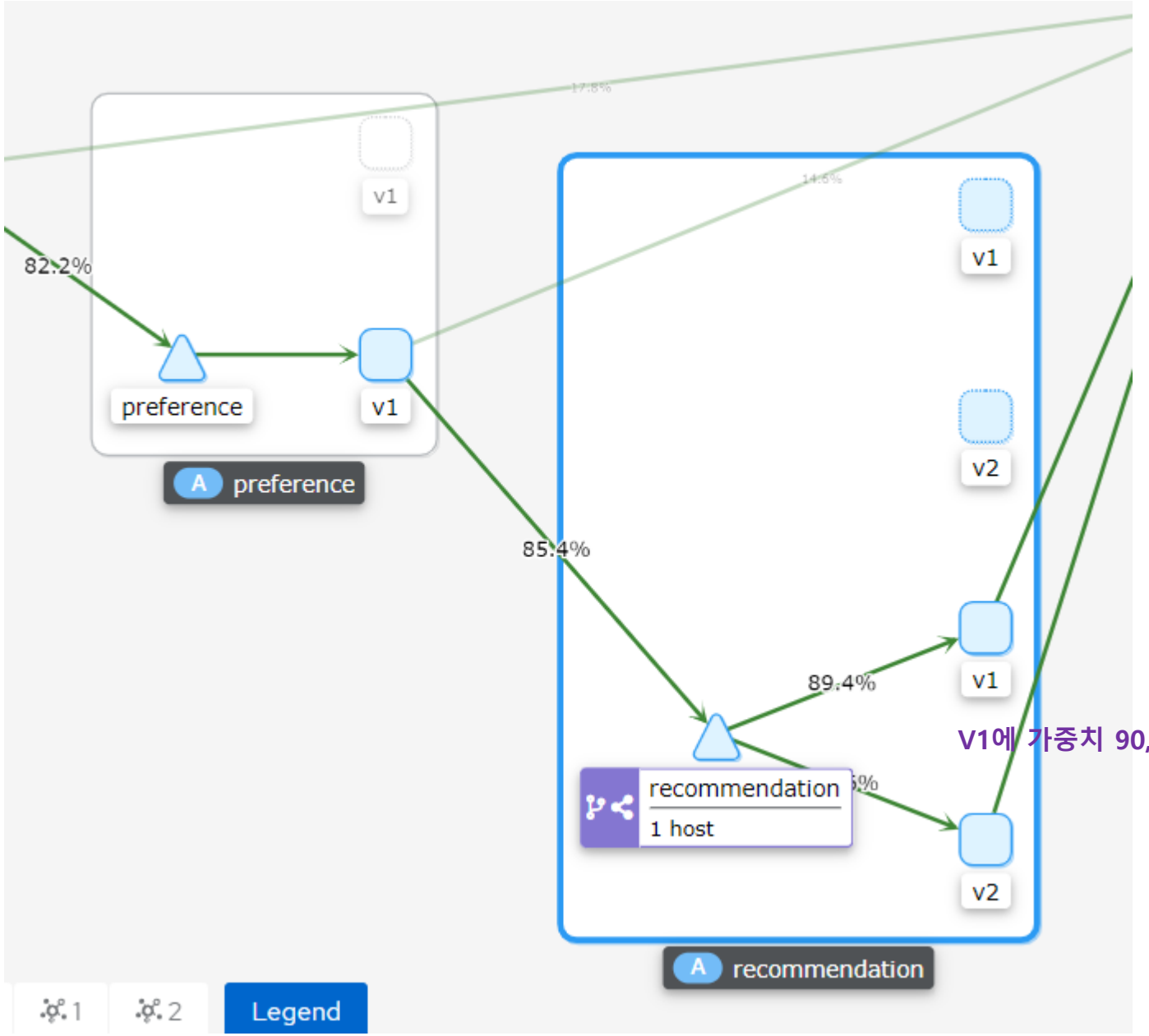
```
root@labs-910775232:/home/project/istio-tutorial# kubectl replace -f istiofiles/virtual-service-recommendation-v1.yml -n tutorial
virtualservice.networking.istio.io/recommendation replaced
root@labs-910775232:/home/project/istio-tutorial# cat istiofiles/virtual-service-recommendation-v1.yml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: recommendation
spec:
  hosts:
  - recommendation
  http:
  - route:
    - destination:
        host: recommendation
        subset: version-v1
      weight: 100
```

Canary 라우팅 비율별 배포 정책 예시(90 : 10)

kubectl apply -f istiofiles/virtual-service-recommendation-v1_and_v2.yml -n tutorial

```
root@labs-910775232:/home/project/istio-tutorial# kubectl apply -f istiofiles/virtual-service-recommendation-v1_and_v2.yml -n tutorial
virtualservice.networking.istio.io/recommendation created
root@labs-910775232:/home/project/istio-tutorial# cat istiofiles/virtual-service-recommendation-v1_and_v2.yml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: recommendation
spec:
  hosts:
  - recommendation
  http:
  - route:
    - destination:
        host: recommendation
        subset: version-v1
      weight: 90
    - destination:
        host: recommendation
        subset: version-v2
      weight: 10
```

Routing 결과 확인 - Kiali(Externl-IP:20001) 접속



V1에 가중치 90, V2에 가중치 10이 수렴함을 볼 수 있음

Header 기반 라우팅(브라우저 유형별)

virtual-service-firefox-recommendation-v2.yml

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: recommendation
spec:
  hosts:
  - recommendation
  http:
  - match:
    - headers:
        baggage-user-agent:
          regex: .*Firefox.*
    route:
    - destination:
        host: recommendation
        subset: version-v2
  - route:
    - destination:
        host: recommendation
        subset: version-v1
---
```

kubectl apply -f istiofiles/virtual-service-firefox-recommendationv2.yml -n tutorial

```
---root@labs-910775232:/home/project/istio-tutorial# vi istiofiles/virtual-service-firefox-recommendation-v2.yml
root@labs-910775232:/home/project/istio-tutorial# kubectl apply -f istiofiles/virtual-service-firefox-recommendation-v2.yml -n tutorial
Warning: kubectl apply should be used on resource created by either kubectl create --save-config or kubectl apply
virtualservice.networking.istio.io/recommendation configured
```

크롬으로 호출하니 v1으로만 라우팅 됨을 확인할 수 있음.