

# ISTIO with Bookinfo App

DHANUSH GOWDA VIDYARANYA

*Site Reliability Engineer, Cisco*

**DEVOPS  
ENTERPRISE  
SUMMIT**

AN  REVOLUTION EVENT



# Dhanush Gowda Vidyaranya

Site Reliability Engineer, Cisco

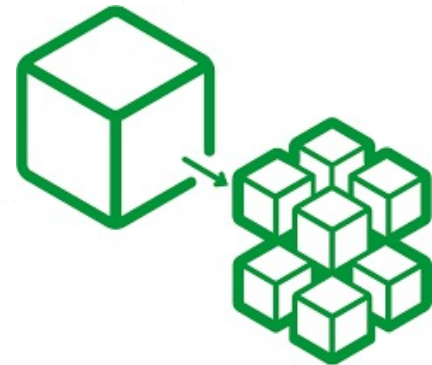
Dhanush is working as a Site Reliability Engineer for Cisco DevNet with focus around Cloud Infrastructure Management and Automation. He deeply cares about helping developers and platform team to automate various aspects of DevOps. Dhanush is passionate about Terraform, K8s, Git, Ansible, Docker, Jenkins and puppet. Currently certified in AWS, Kubernetes, Docker and Terraform.

# AGENDA

- What are Micro Services?
- Example Micro Service Architecture
- Challenges of Microservice Architecture
- ISTIO Architecture
- Core features of Istio
- Traffic management
- Demo

# What are Microservices ??

- Microservices are an architectural approach to building applications.
- Single application is composed of many loosely coupled and independently deployable smaller components, or services



# BookInfo Application

The screenshot displays a web application titled "BookInfo Sample" with a "Sign in" button in the top right corner. The main content area features the title "The Comedy of Errors" in blue. Below the title is a summary: "Summary: Wikipedia Summary: The Comedy of Errors is one of William Shakespeare's early plays. It is his shortest and one of his most farcical comedies, with a major part of the humour coming from slapstick and mistaken identity, in addition to puns and word play." The page is divided into two columns: "Book Details" on the left and "Book Reviews" on the right. The "Book Details" column lists the following information: Type: paperback, Pages: 200, Publisher: PublisherA, Language: English, ISBN-10: 1234567890, and ISBN-13: 123-1234567890. The "Book Reviews" column contains two reviews. The first review, by "Reviewer1", states "An extremely entertaining play by Shakespeare. The slapstick humour is refreshing!" and is rated with five red stars. The second review, by "Reviewer2", states "Absolutely fun and entertaining. The play lacks thematic depth when compared to other plays by Shakespeare." and is also rated with five red stars. A "Reviews" button is located in the top right corner of the application, and a "Details" button is located in the bottom right corner. Arrows point from these buttons to their respective sections in the diagram.

BookInfo Sample

Sign in

## The Comedy of Errors

Summary: Wikipedia Summary: The Comedy of Errors is one of William Shakespeare's early plays. It is his shortest and one of his most farcical comedies, with a major part of the humour coming from slapstick and mistaken identity, in addition to puns and word play.

### Book Details

Type:  
paperback  
Pages:  
200  
Publisher:  
PublisherA  
Language:  
English  
ISBN-10:  
1234567890  
ISBN-13:  
123-1234567890

### Book Reviews

An extremely entertaining play by Shakespeare. The slapstick humour is refreshing!

— Reviewer1  
★★★★★

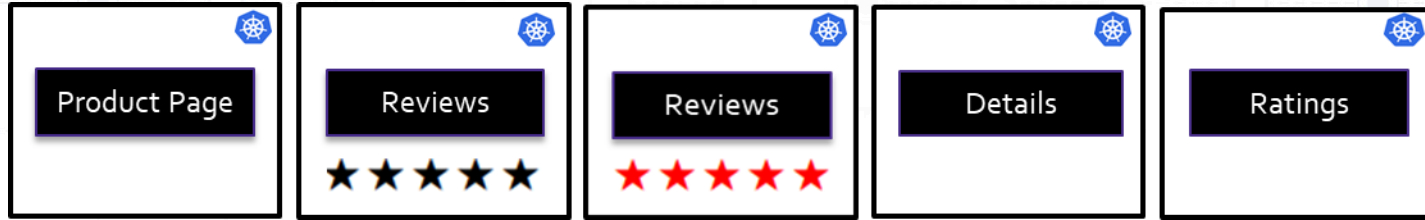
Absolutely fun and entertaining. The play lacks thematic depth when compared to other plays by Shakespeare.

— Reviewer2  
★★★★★

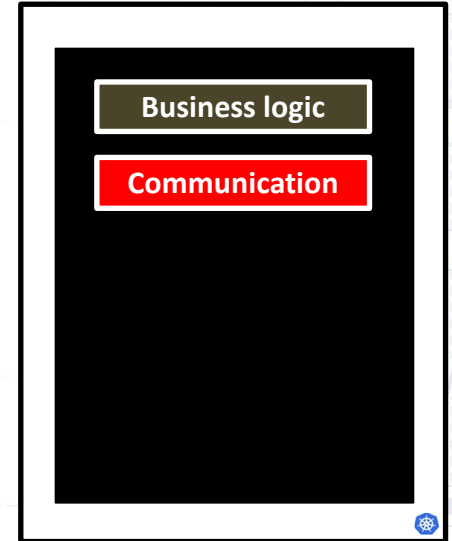
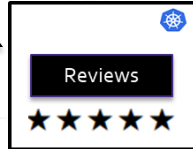
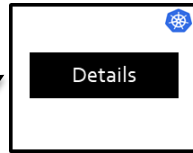
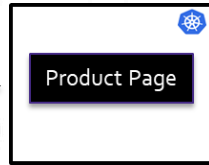
Reviews

Details

## Challenges of Microservice Architecture

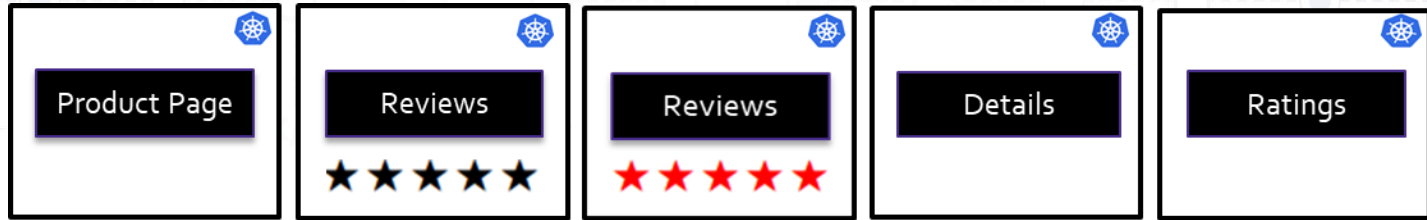


Communication

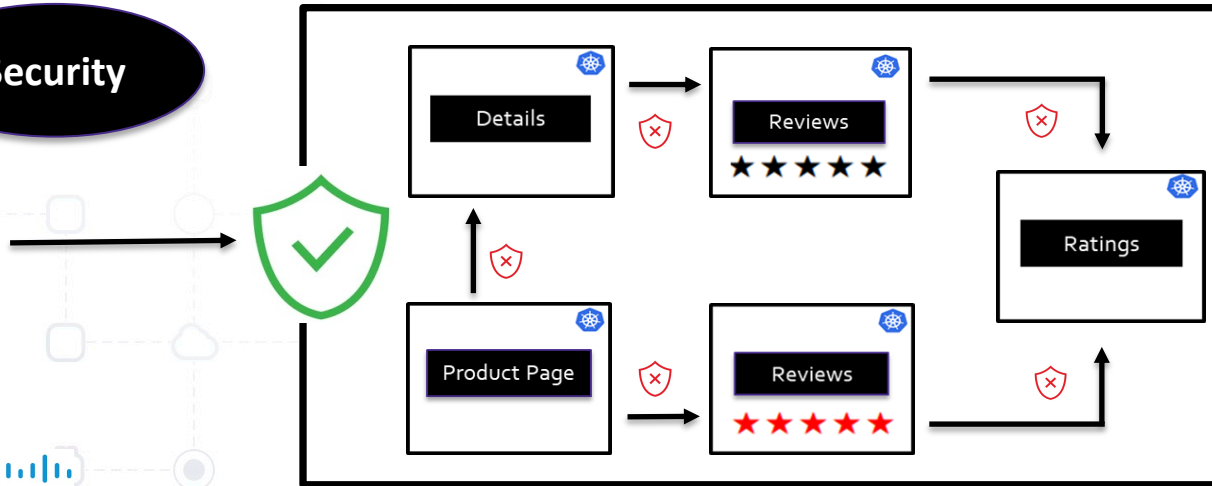




## Challenges of Microservice Architecture



Security



## Challenges of Microservice Architecture

Product Page

Reviews



Reviews



Details

Ratings

Retry Logic

Business logic

Communication

Security

Retry



## Challenges of Microservice Architecture

Product Page

Reviews



Reviews



Details

Ratings

Metrics  
&  
Tracing

Business logic

Communication

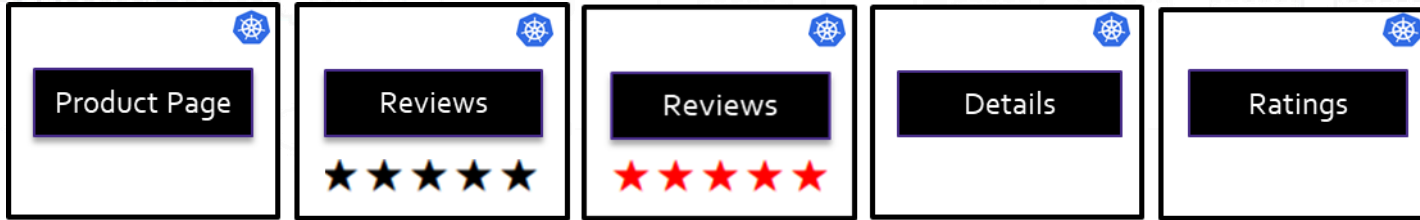
Security

Retry

Metrics

Tracing

## Challenges of Microservice Architecture



These Nonbusiness logic must be added to each application

Developers don't work on actual service/Application logic

Adds complexity to the services

Business logic

Communication

Security

Retry

Metrics

Tracing

## Solution: Service Mesh with Sidecar Pattern

**Sidecar Proxy**

**Control Plane**

Handles networking logic

Acts as a Proxy

Third party Application

Cluster operators can configure it easily

Developers can focus on the actual business Logic

Control Plane injects the Sidecar Proxy

Communication

Security

Retry

Metrics

Tracing

Proxy

Business logic

MS

# What is Service Mesh ?? ISTIO ??

Service Mesh is  
a Pattern

ISTIO is an  
Implementation

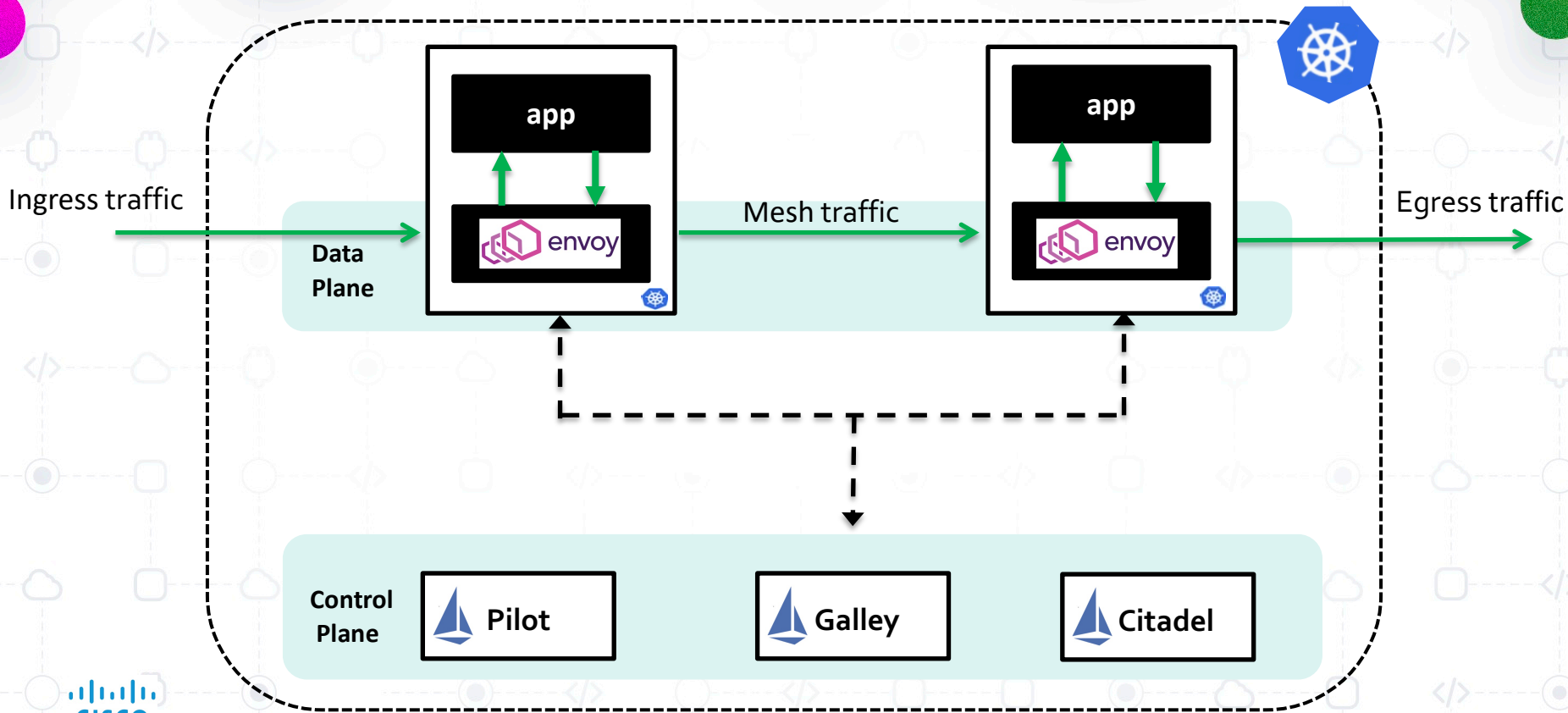


A service mesh is a platform layer on top of the infrastructure layer that enables managed, observable, and secure communication between individual services.



Istio is a configurable, open-source service-mesh layer that connects, monitors, and secures the containers in a Kubernetes cluster.

# Istio Architecture



# ISTIO core features



**Traffic Management**



**Authentication between services**



**Authorization between services**



**Secure Communication between services**



**Observability(tracing, monitoring and logging)**



# Configure ISTIO

- YAML files
- CustomResourceDefinitions(CRD)
- No need to learn Istio specific language.

# ISTIO Installation Configuration Profiles

**default:** This is recommended for production deployments and configures the default settings of the IstioOperator API

**demo:** This is to play around with Istio and for learning purpose, especially when using Minikube or a setup that has limited resources

**minimal:** It contains a minimum number of features just to support traffic management.

**external:** used for configuring a remote cluster that is managed by an external control plane or by a control plane in a primary cluster of a multicluster mesh

**empty:** deploys nothing. This can be useful as a base profile for custom configuration

**preview:** the preview profile contains features that are experimental

# Traffic management API resources

- Gateway
- Virtual Service
- DestinationRule

# ISTIO Gateway

Gateways are primarily used to manage ingress traffic, but we can also configure egress gateways.

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

```

# ISTIO Virtual Service

- Istio VirtualService is one level higher than Kubernetes service. A VirtualService defines a set of traffic routing rules to apply when a host is addressed.
- It can be used to apply traffic routing, fault injection, retries , redirect, rewrite and many other configurations to services.

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: reviews-route
spec:
  hosts:
  - reviews.prod.svc.cluster.local
  http:
  - name: "reviews-v2-routes"
    match:
    - uri:
        prefix: "/wpcatalog"
    - uri:
        prefix: "/consumercatalog"
    rewrite:
      uri: "/newcatalog"
    route:
    - destination:
        host: reviews.prod.svc.cluster.local
        subset: v2
    - name: "reviews-v1-route"
      route:
      - destination:
          host: reviews.prod.svc.cluster.local
          subset: v1
```

# ISTIO DestinationRule

DestinationRule defines policies that apply to traffic intended for a service after routing has occurred.

```
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: productpage
spec:
  host: productpage
  subsets:
  - name: v1
    labels:
      version: v1
---
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: reviews
spec:
  host: reviews
  subsets:
  - name: v1
    labels:
      version: v1
  - name: v2
    labels:
      version: v2
  - name: v3
    labels:
      version: v3
---
```

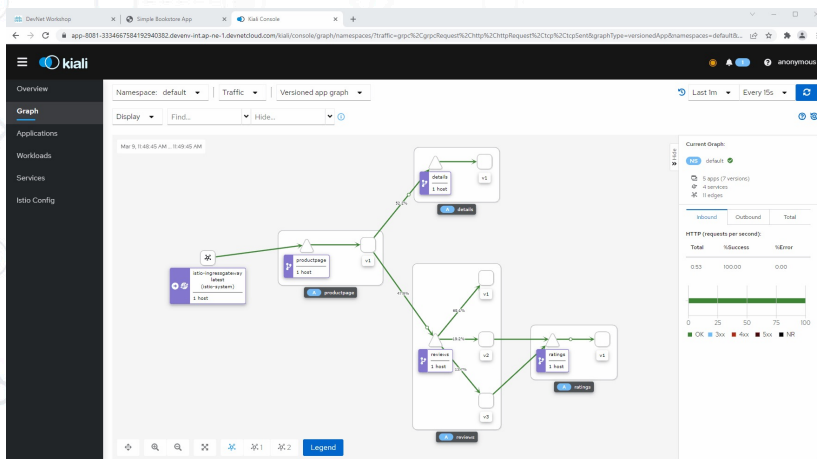


# Traffic Management

- Routing to a specific version.
- Traffic Splitting.
- User Identity-Based Routing.

# Routing to a specific version

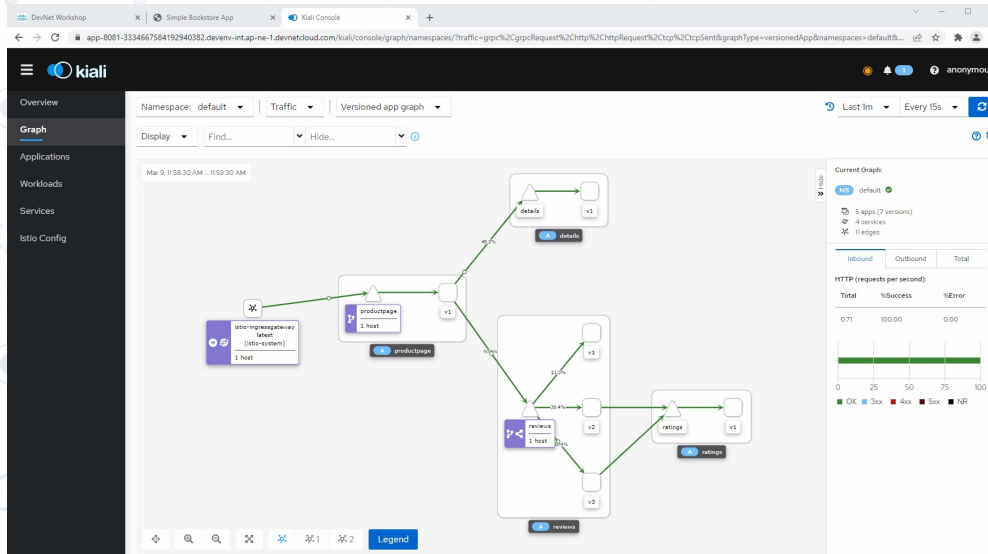
Requests can be routed dynamically to multiple versions of microservice. In order to route to one version only, we can apply virtual services that sets the default version of microservice.



```
dhanushgowda91@master-node:~/Istio/istio-1.11.2$ cat samples/bookinfo/networking/virtual-service-all-v1.yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: productpage
spec:
  hosts:
  - productpage
  http:
  - route:
    - destination:
        host: productpage
        subset: v1
  ---
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: reviews
spec:
  hosts:
  - reviews
  http:
  - route:
    - destination:
        host: reviews
        subset: v1
  ---
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: ratings
spec:
  hosts:
  - ratings
  http:
  - route:
    - destination:
        host: ratings
        subset: v1
  ---
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: details
spec:
  hosts:
  - details
  http:
  - route:
    - destination:
        host: details
        subset: v1
  ---
```

# Traffic shifting

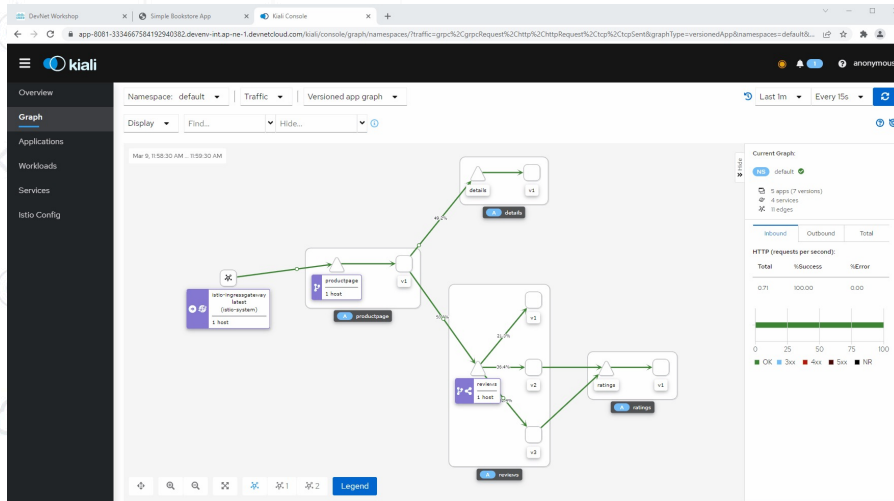
Traffic can be shifted from one version to another version of microservice. A common use case is to migrate traffic gradually from an older version to a new one



```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: reviews
spec:
  hosts:
    - reviews
  http:
    - route:
        - destination:
            host: reviews
            subset: v2
            weight: 50
        - destination:
            host: reviews
            subset: v3
            weight: 50
```

# User Identity-Based Routing

Traffic from a specific user can be routed to a specific micro service version. In the screenshot shown traffic from user Jason will be routed to a service reviews v2 and rest all the traffic routed to v1. Use case is to route traffic for users from different browser agents like **Mozilla Firefox/IE/Safari** to different version of microservices.



```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: reviews
spec:
  hosts:
    - reviews
  http:
    - match:
      - headers:
          end-user:
            exact: jason
      route:
        - destination:
            host: reviews
            subset: v2
        - route:
            destination:
              host: reviews
              subset: v1
```

# Visualizing Mesh(Kiali)

## Node Shapes

- Workload
- App
- ⬠ Operation
- △ Service
- ⬠ Service Entry

## Edges

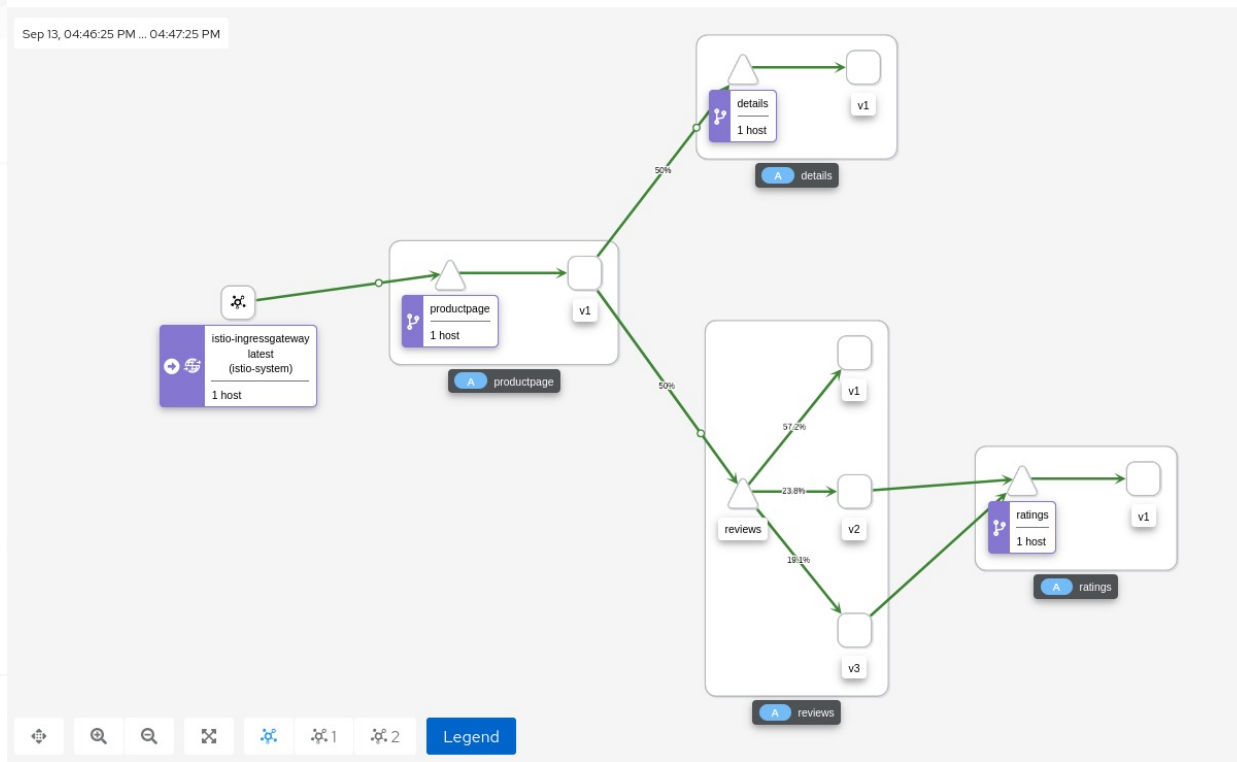
- Failure
- Degraded
- Healthy
- TCP Connection
- Idle
- 🔒 mTLS (badge)

## Node Badges

- ⚡ Circuit Breaker
- 🚫 Fault Injection
- 🌐 Gateway
- 📅 Missing Sidecar
- ⌚ Request Timeout
- 🔄 Traffic Shifting/TCP
- 🔄 Traffic Shifting
- 📡 Traffic Source
- 🔄 Virtual Service/Request Routing

## Traffic Animation

- Normal Request
- ⬠ Failed Request
- 🔄 TCP Traffic



# DEMO

- **ISTIO Installation**
- **Bookinfo app deployment**
- **Visualizing mesh using Kiali**
- **Traffic management**



## Continue your learning journey

- <https://istio.io>
- <https://github.com/istio/istio>
- <https://developer.cisco.com>



# Thank You!