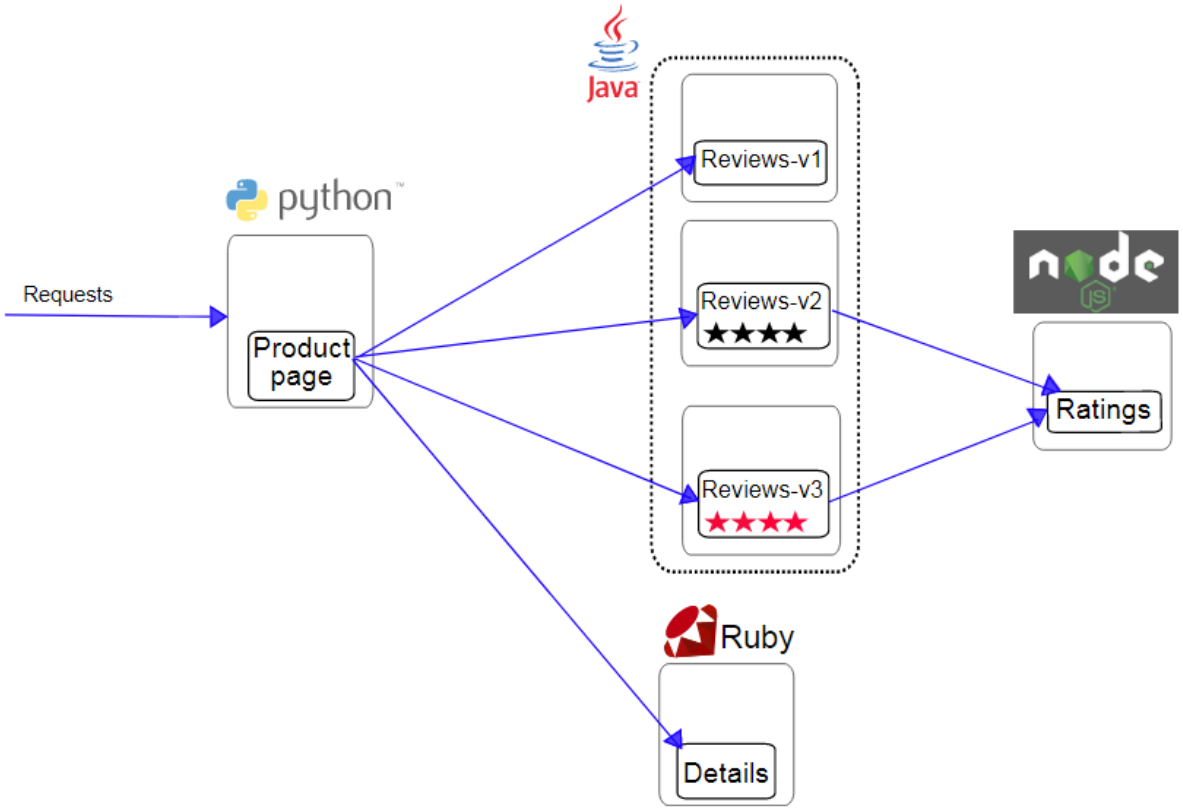
Deploy microservices application:

To play with Istio and demonstrate some of it's capabilities, we will deploy the example BookInfo application, which is included the Istio package.

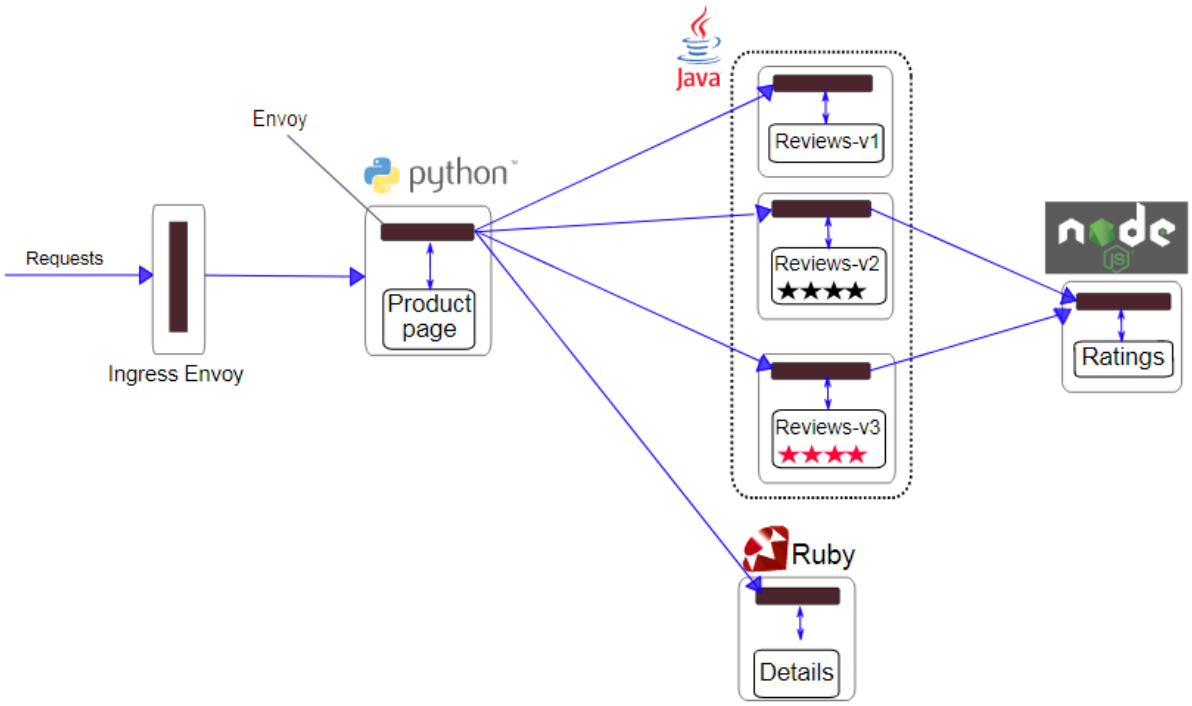
What is the BookInfo Application?

This application is a polyglot composition of microservices are written in different languages and sample BookInfo application displays information about a book, similar to a single catalog entry of an online book store. Displayed on the page is a description of the book, book details (ISBN, number of pages, and so on), and a few book reviews.



Bookinfo Application without Istio

To run the sample with Istio requires no changes to the application itself. Instead, we simply need to configure and run the services in an Istio-enabled environment, with Envoy sidecars injected along side each service. The needed commands and configuration vary depending on the runtime environment although in all cases the resulting deployment will look like this:



Bookinfo Application

Deploying Sample App with Automatic sidecar injection:

Let us now verify sidecar injector deployment & label namespace for automatic sidecar injection.

```
kubectl -n istio-system get deployment -l istio=sidecar-injector
```

Output:

```
NAME                     DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
istio-sidecar-injector   1         1         1            1           1d
```

NamespaceSelector decides whether to run the webhook on an object based on whether the namespace for that object matches the selector.

Label the default namespace with istio-injection=enabled

```
kubectl label namespace default istio-injection=enabled
```

```
kubectl get namespace -L istio-injection
```

Output:

```
NAME           STATUS    AGE       ISTIO-INJECTION
default        Active    1h        enabled
istio-system   Active    1h
kube-public    Active    1h
kube-system    Active    1h
```

Now that we have the sidecar injector with mutating webhook in place and the namespace labelled for automatic sidecar injection, we can proceed to deploy the sample app:

```
kubectl apply -f bookinfo.yaml
```

Verify Bookinfo deployment

1. Verify that previous deployments are all in a state of AVAILABLE before continuing. **Do not proceed until they are up and running.**

```
watch kubectl get deployment
```

2. Inspect the details of the pods

   Let us look at the details of the pods:

```
watch kubectl get po
```

Let us look at the details of the services:

```
watch kubectl get svc
```

Now let us pick a service, for instance productpage service, and view it's sidecar configuration:

```
kubectl get po

kubectl describe pod productpage-v1-.....
```