Traffic Mirroring:

Traffic mirroring, also called shadowing, is a powerful concept that allows development teams to bring changes to production with as little risk as possible. Mirroring sends a copy of live traffic to a mirrored service.

Traffic mirroring can be used in cases when you don't want to **release** the new version and expose users to it, but you'd still like to **deploy** it and observe how it works, gather telemetry data and compare the performance and functionality of the existing service with the new service.

You might ask — **what is the difference between deploying and releasing something?**

When we talk about **deploying a service** to production, we are merely moving the executable code (binaries, containers, whatever form needed for the code to be able to execute) to live in the production environment, but not sending any production traffic to it. The service is there, but it's not (hopefully!) affecting any existing services that are running next to it.

Releasing a service involves taking that deployed service and start routing production traffic to it. At this point, the code we moved to production is being executed and it will probably impact other services and end users.

Routing traffic between two versions, doing blue-green releases is helpful and useful, but there are risks involved — what if the service breaks or malfunctions? Even if service is receiving only 1% of the production traffic, it can still negatively impact a lot of users.

The idea behind traffic mirroring is to minimize the risk of exposing users to potentially buggy service. Instead of deploying, releasing and routing traffic to the new service, we deploy the new service and then just mirror the production traffic being sent to the released version of the service.

Service receiving mirrored traffic can then be observed for errors without impacting any production traffic. In addition to running a variety of tests on the deployed version of the service, you can now also use actual production traffic and increase the testing coverage which could give you more confidence and minimize the risk of releasing a buggy service.

Mirroring sends a copy of live traffic to a mirrored service.

First all traffic will go to `reviews:v1`, then the rule will be applied to mirror a portion of traffic to `reviews:v2`.

Apply the virtual services which will route all traffic to `reviews:v1` of each microservice:

```
kubectl apply -f virtual-service-all-v1.yaml
```

Change the route rule to mirror traffic to `reviews:v2`:

```
kubectl apply -f virtual-service-mirror-v2.yaml
```

Check the logs on both pods reviews:v1 and reviews:v2 by splitting the byobu screen and checking logs of the pod: