Lab 5

Rong Guo

C00290890

Catalog

[Apply weight-based routing 3](#_Toc193671510)

[Route to version 1 4](#_Toc193671511)

[Test the new routing configuration 7](#_Toc193671512)

[Route based on user identity 8](#_Toc193671513)

[Injecting an HTTP delay fault 9](#_Toc193671514)

[Testing the delay configuration 11](#_Toc193671515)

[Fixing the bug 11](#_Toc193671516)

[Injecting an HTTP abort fault 12](#_Toc193671517)

[Testing the abort configuration 13](#_Toc193671518)

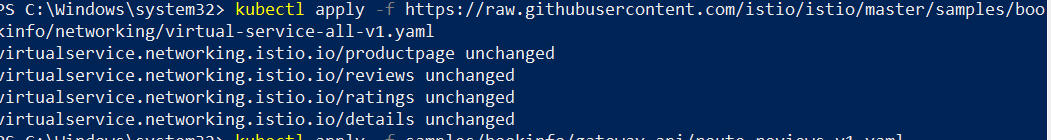
[Configuring the circuit breaker 13](#_Toc193671519)

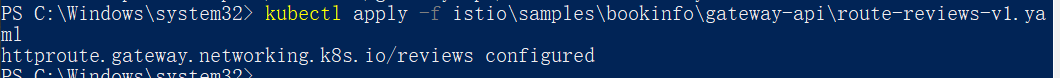
[Adding a client 14](#_Toc193671520)

[Tripping the circuit breaker 15](#_Toc193671521)

Apply weight-based routing

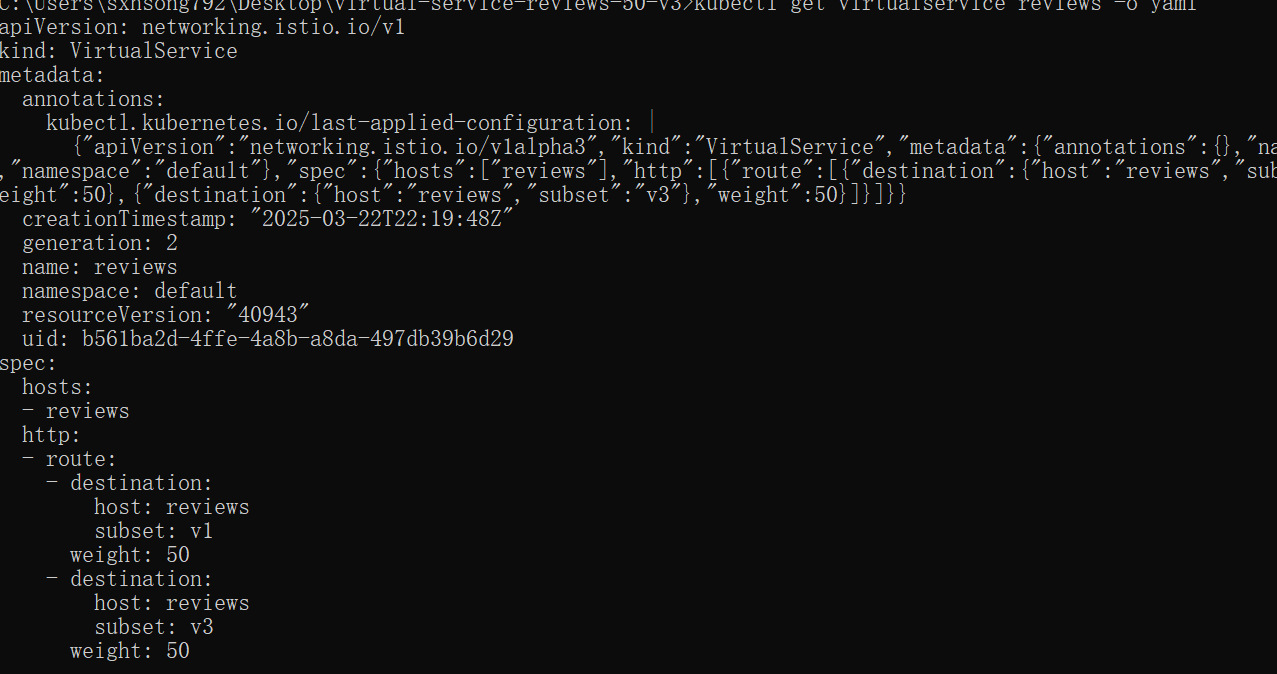
To get started, run this command to route all traffic to the v1 version



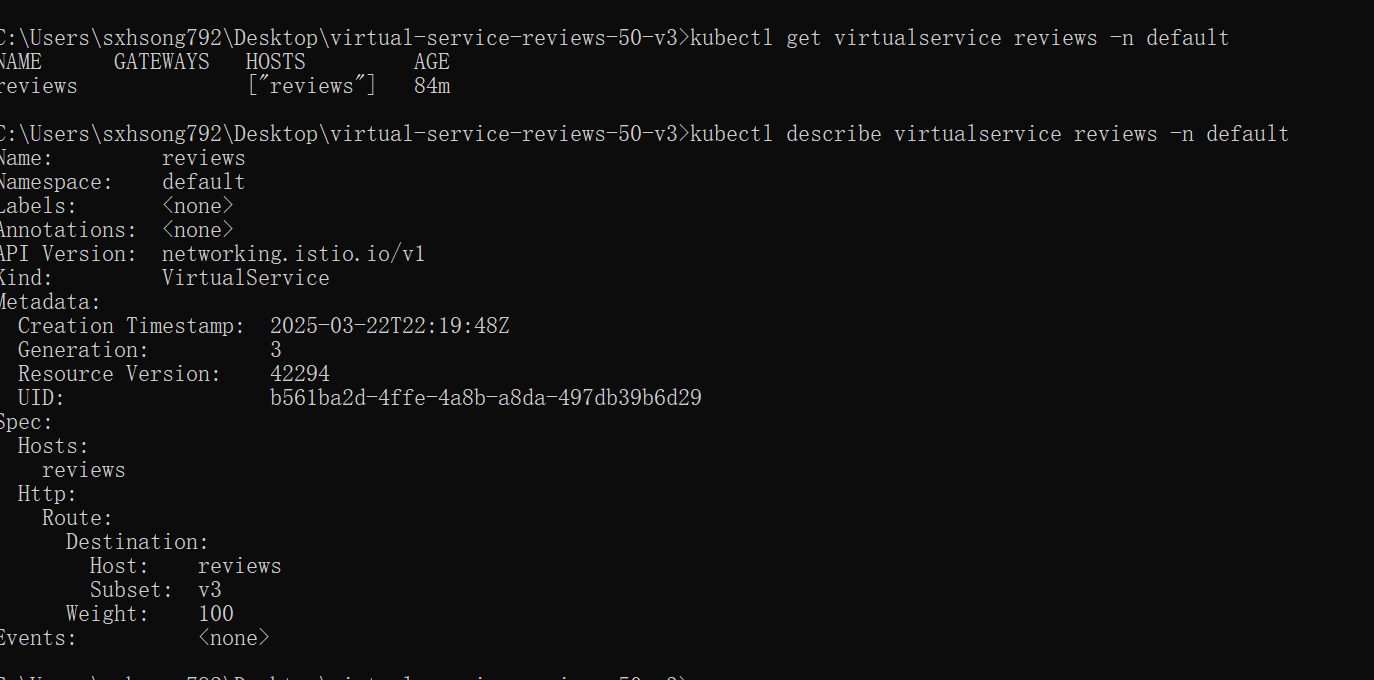


Transfer 50% of the traffic from reviews:v1 to reviews:v3 with the following command:

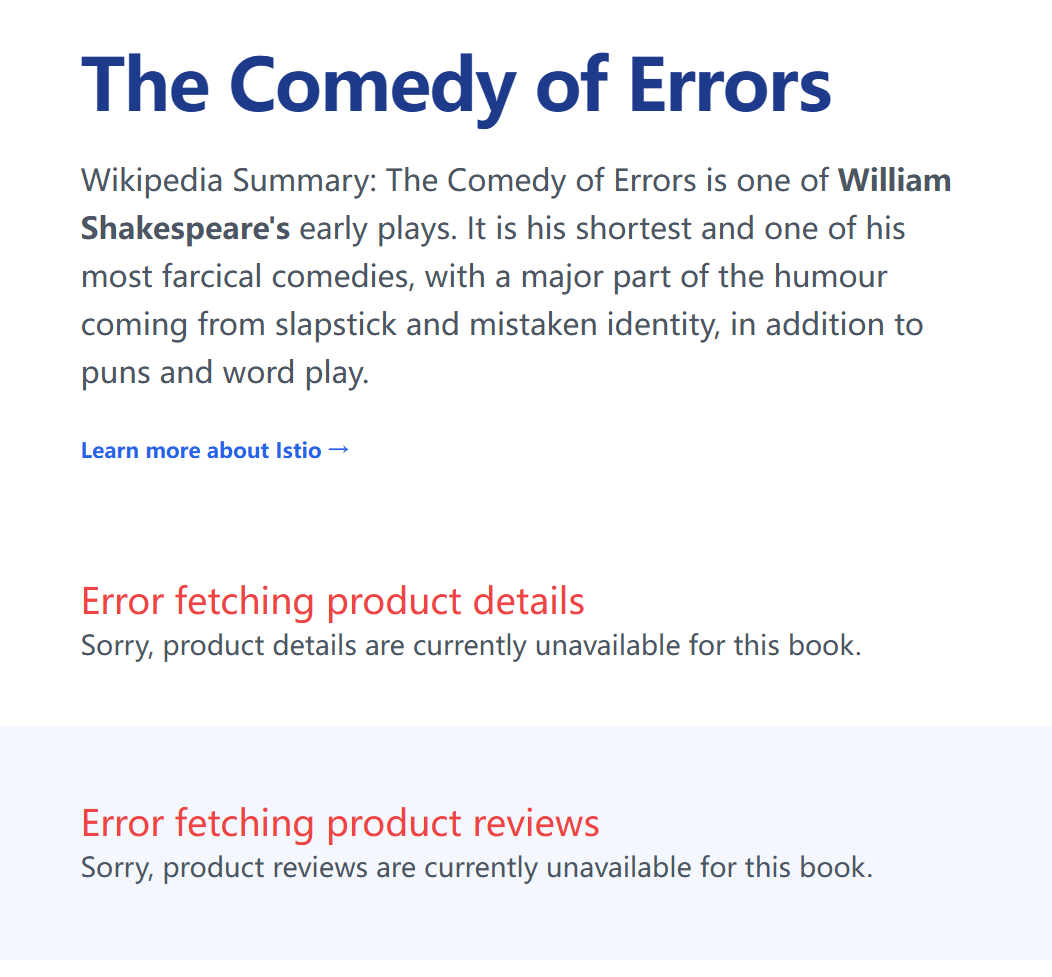




Assuming you decide that the reviews: v3 microservice is stable, you can route 100% of the traffic to reviews: v3 by applying this virtual service

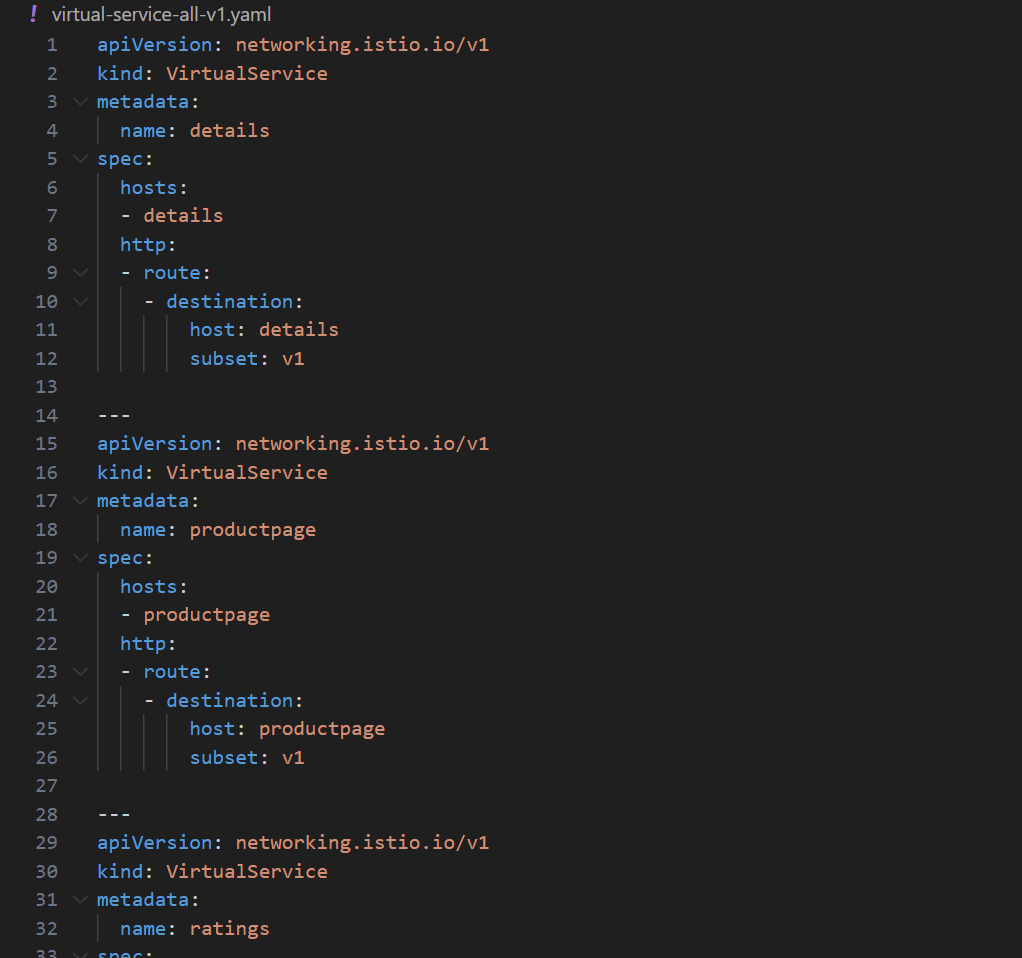


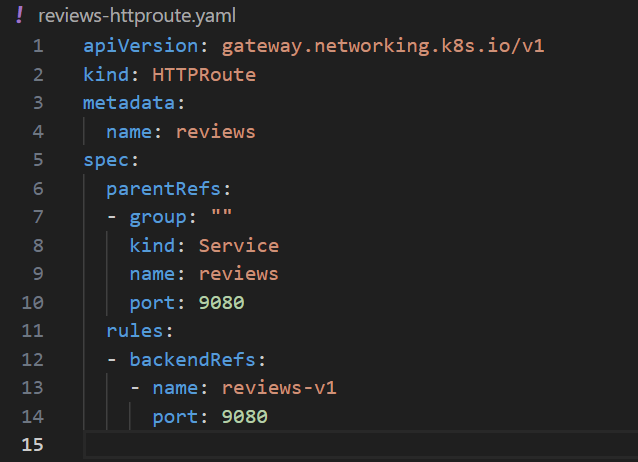
Refresh the /productpage several times.



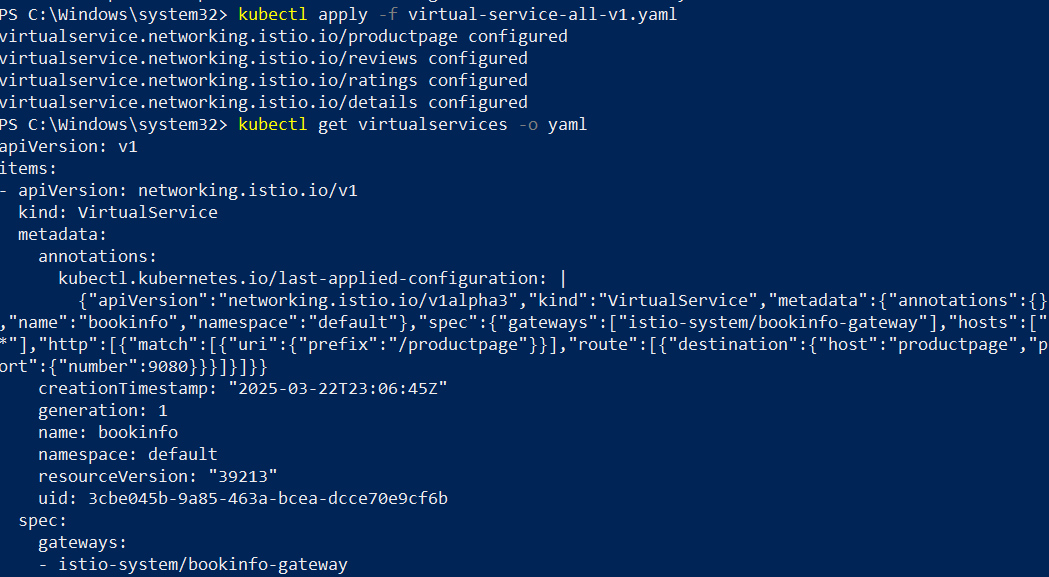
Route to version 1

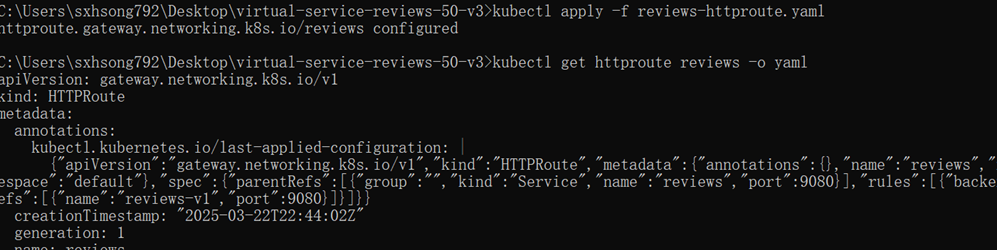
Save the YAML configuration to a file

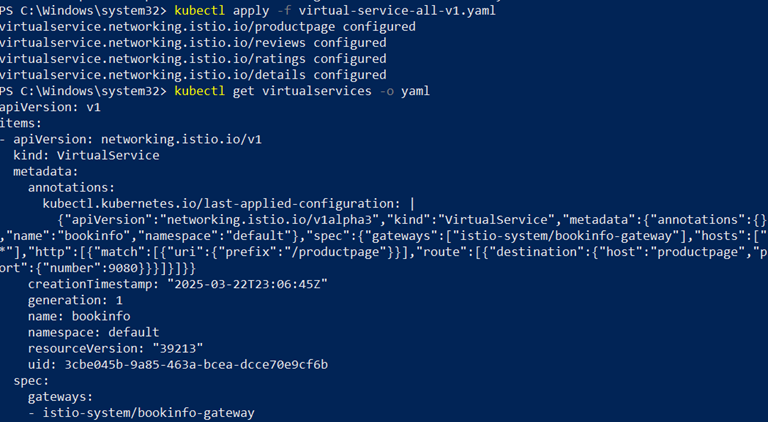


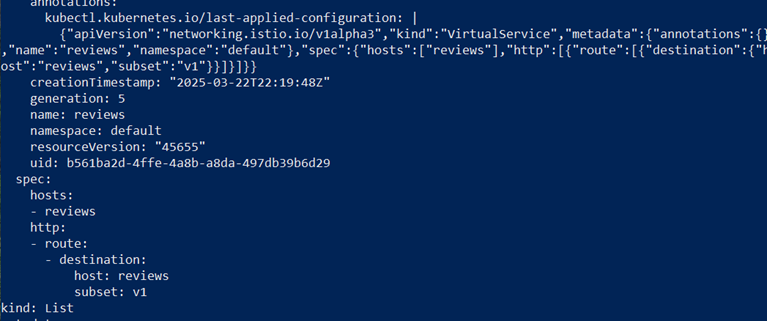


Apply the configuration file



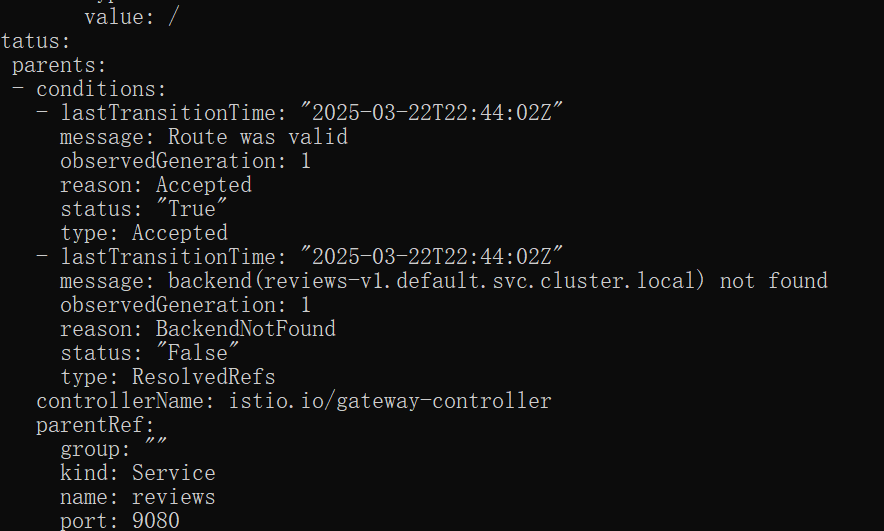


Verify the configuration

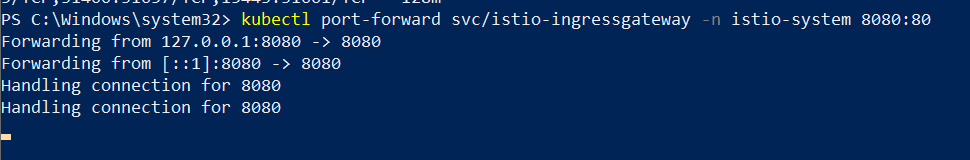


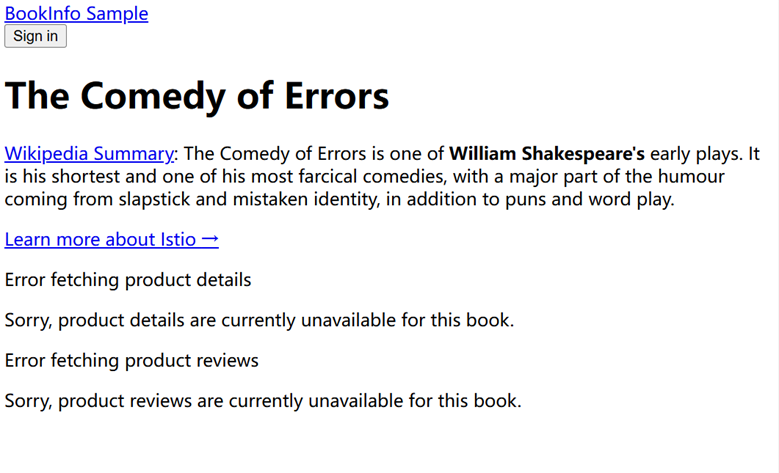
文本

描述已自动生成



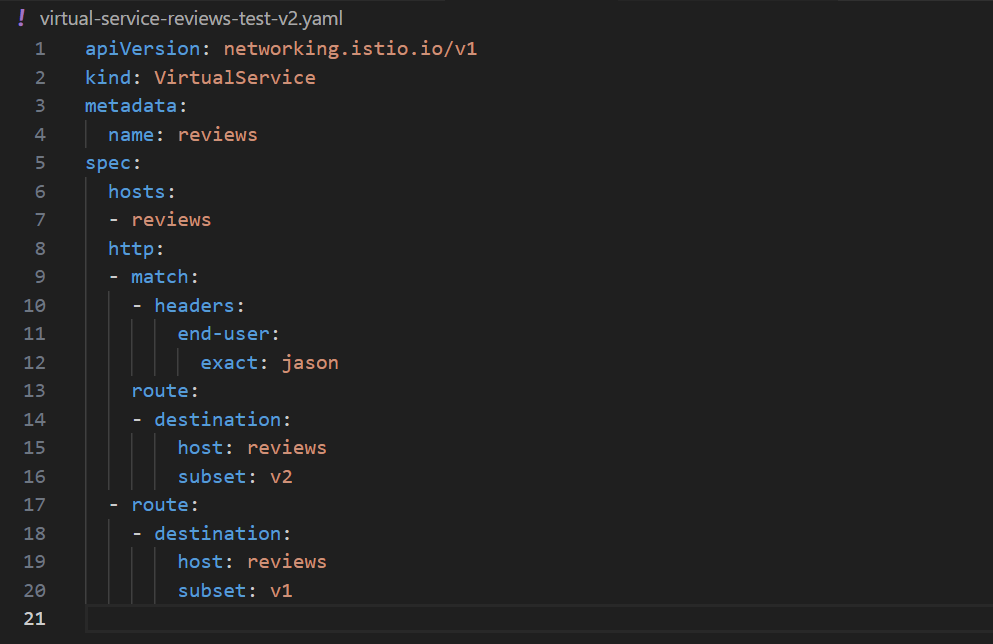
Test the new routing configuration

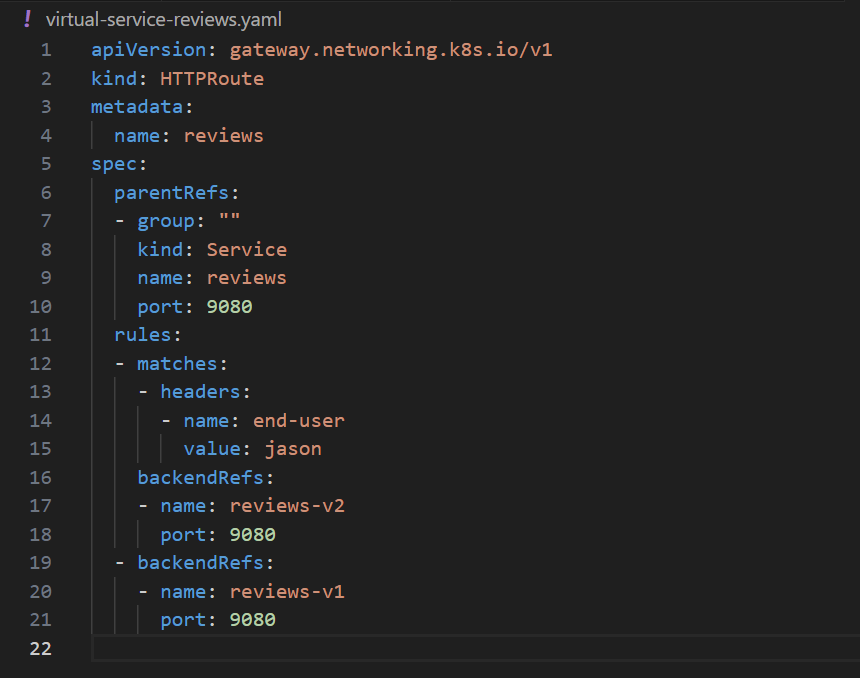


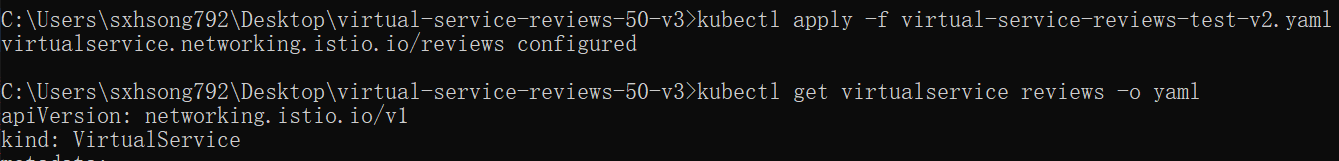


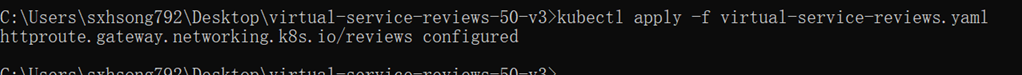
Route based on user identity

Save the YAML configuration to a file

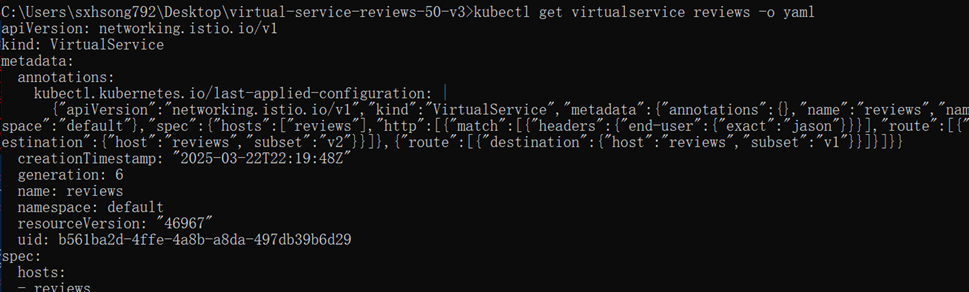




Apply the configuration file



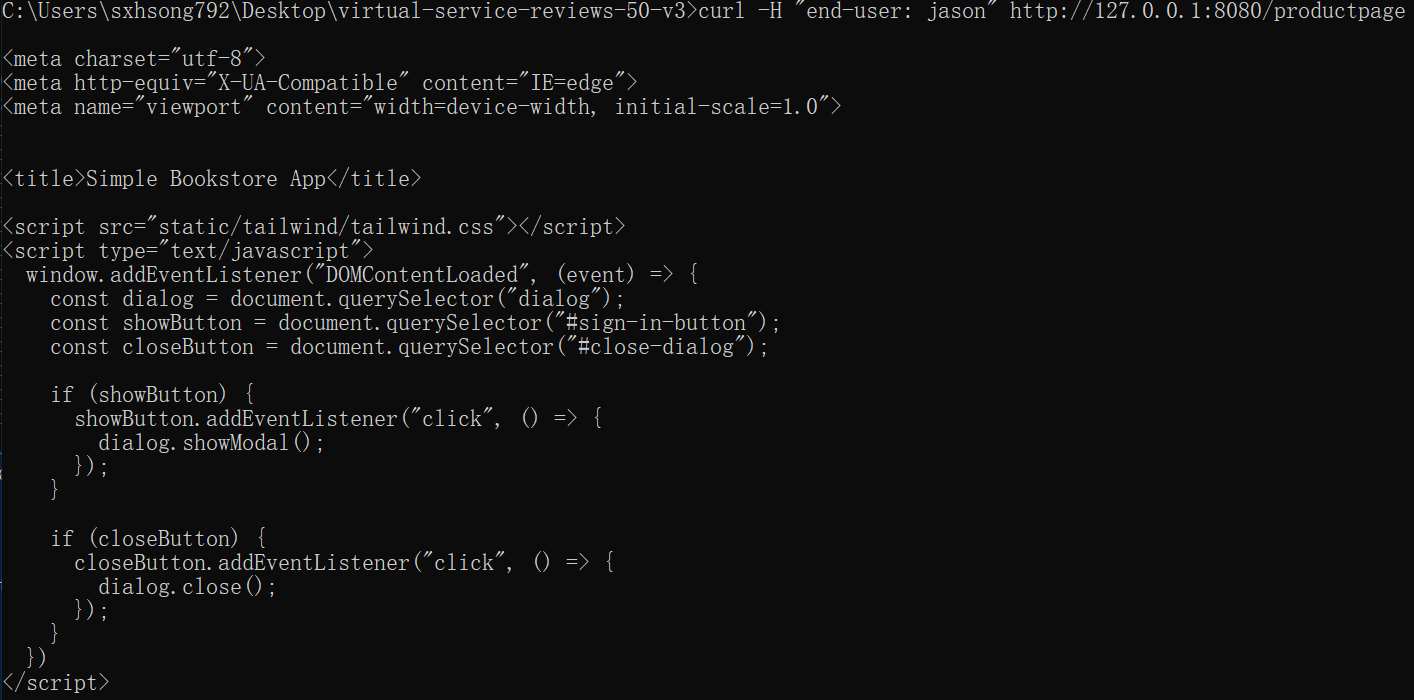
Verify the configuration



文本

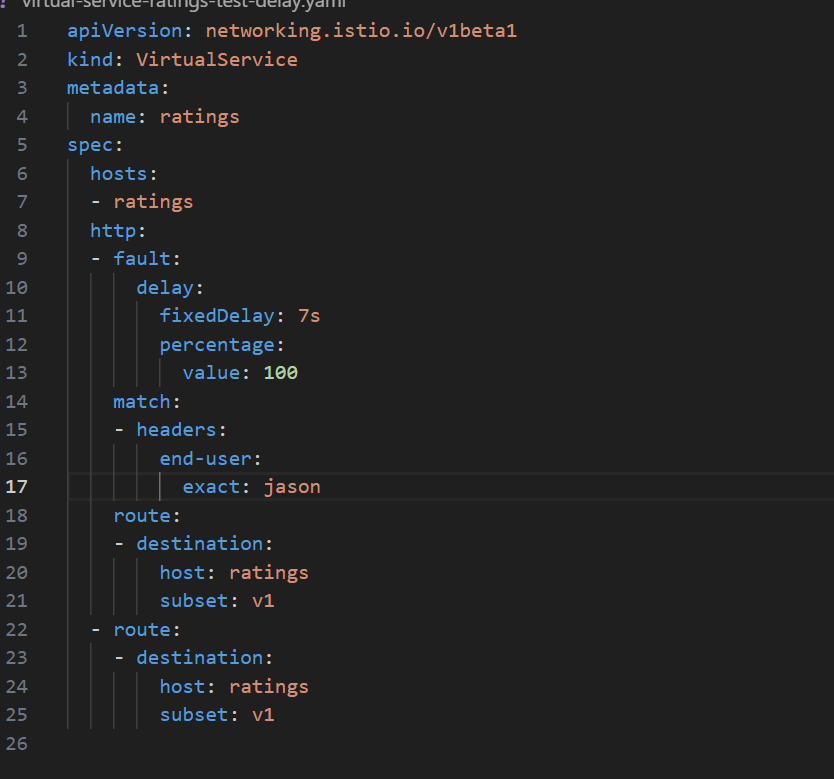
描述已自动生成

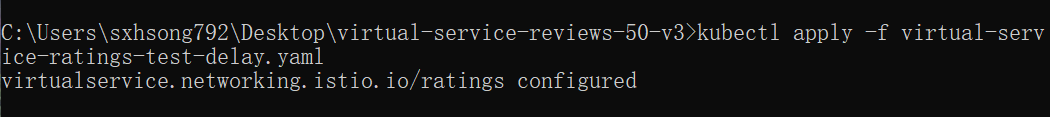
View Jason



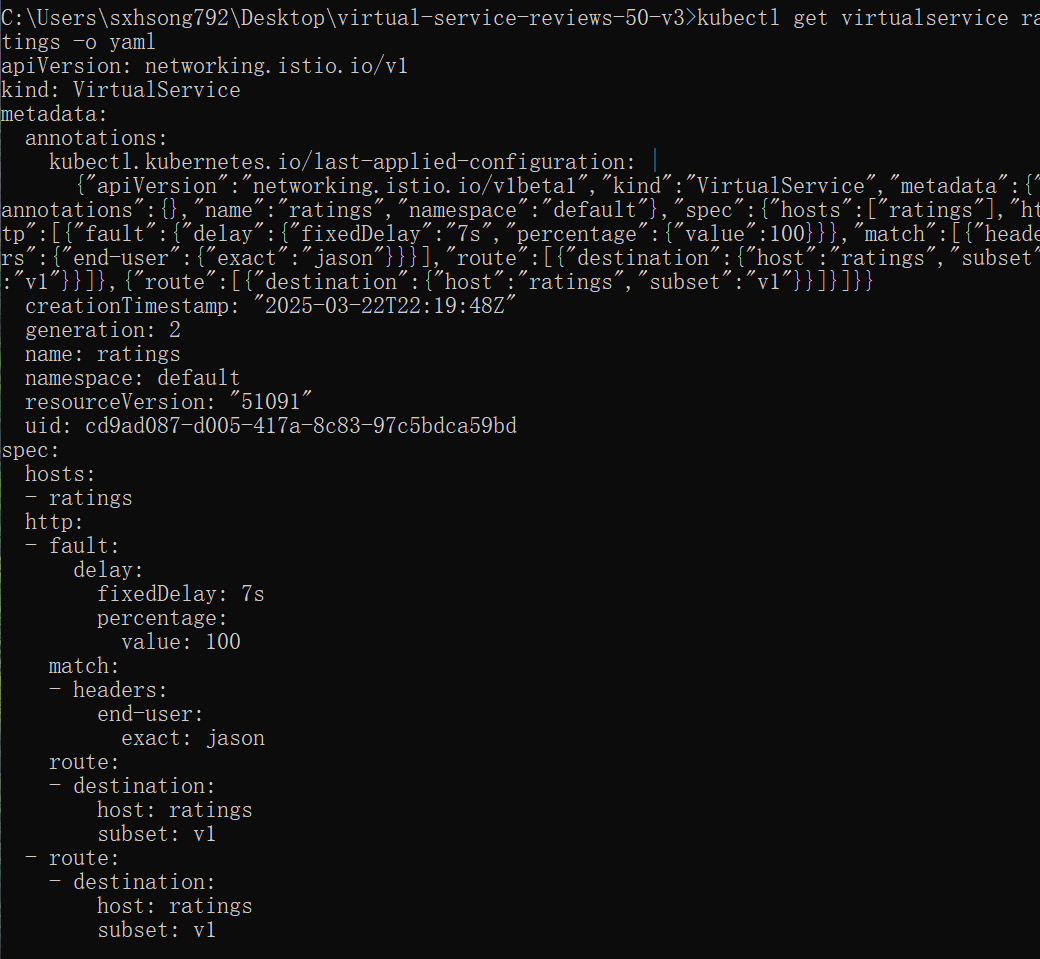
Injecting an HTTP delay fault

Create a fault injection rule to delay traffic coming from the test user jason



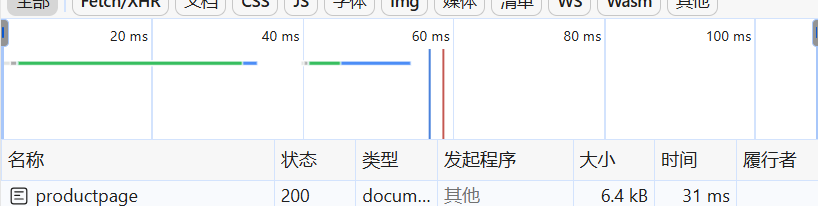


Confirm the rule was created:



Testing the delay configuration



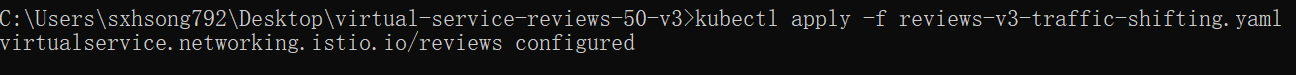


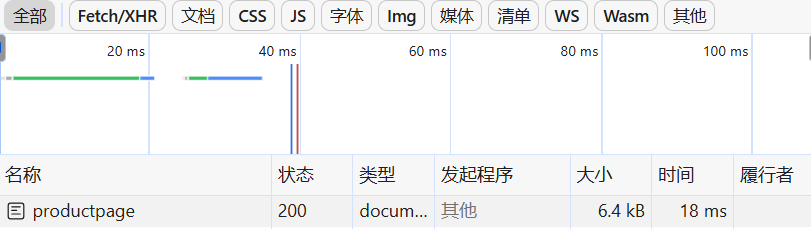
Fixing the bug

Create file



Apply it

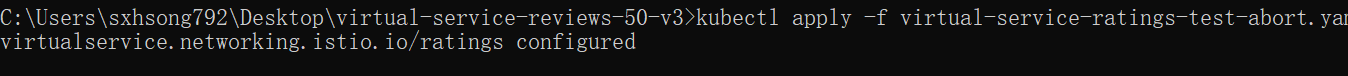




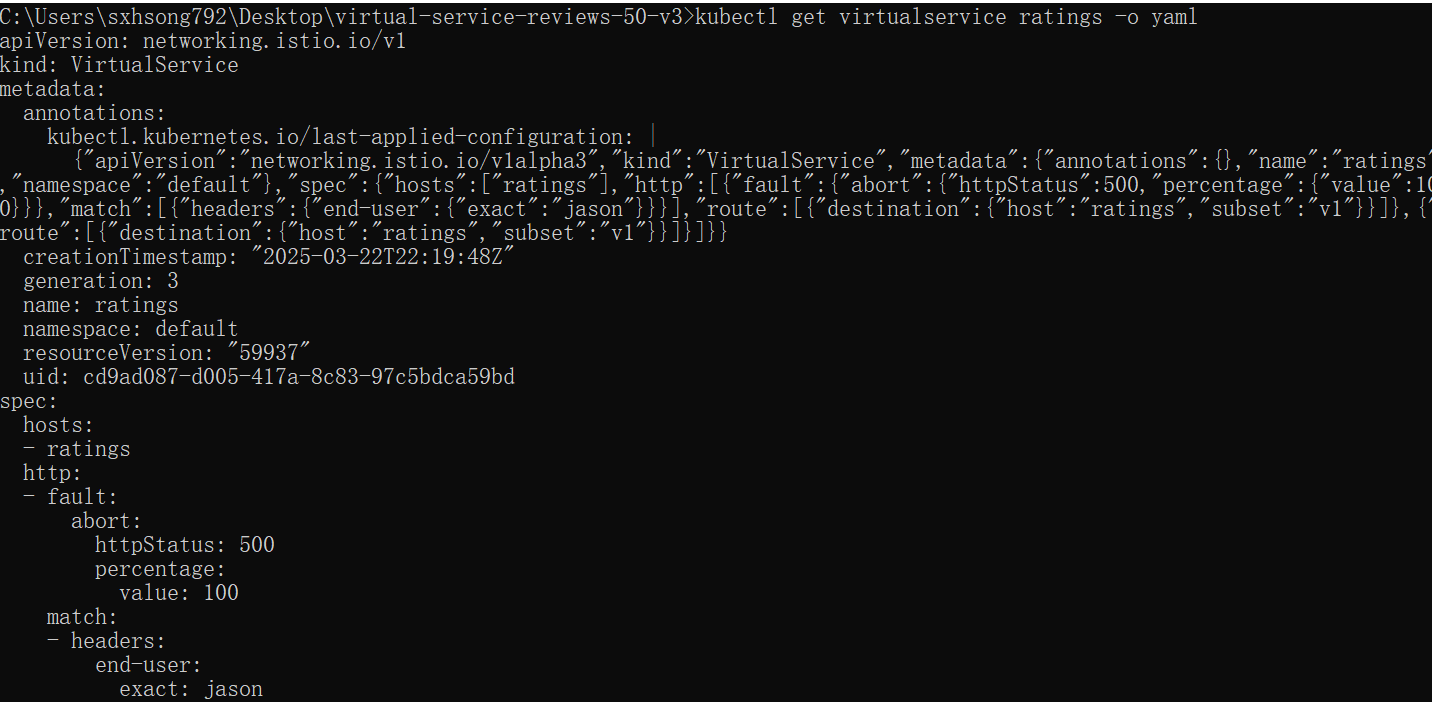
Injecting an HTTP abort fault

Create a fault injection rule to send an HTTP abort for user jason

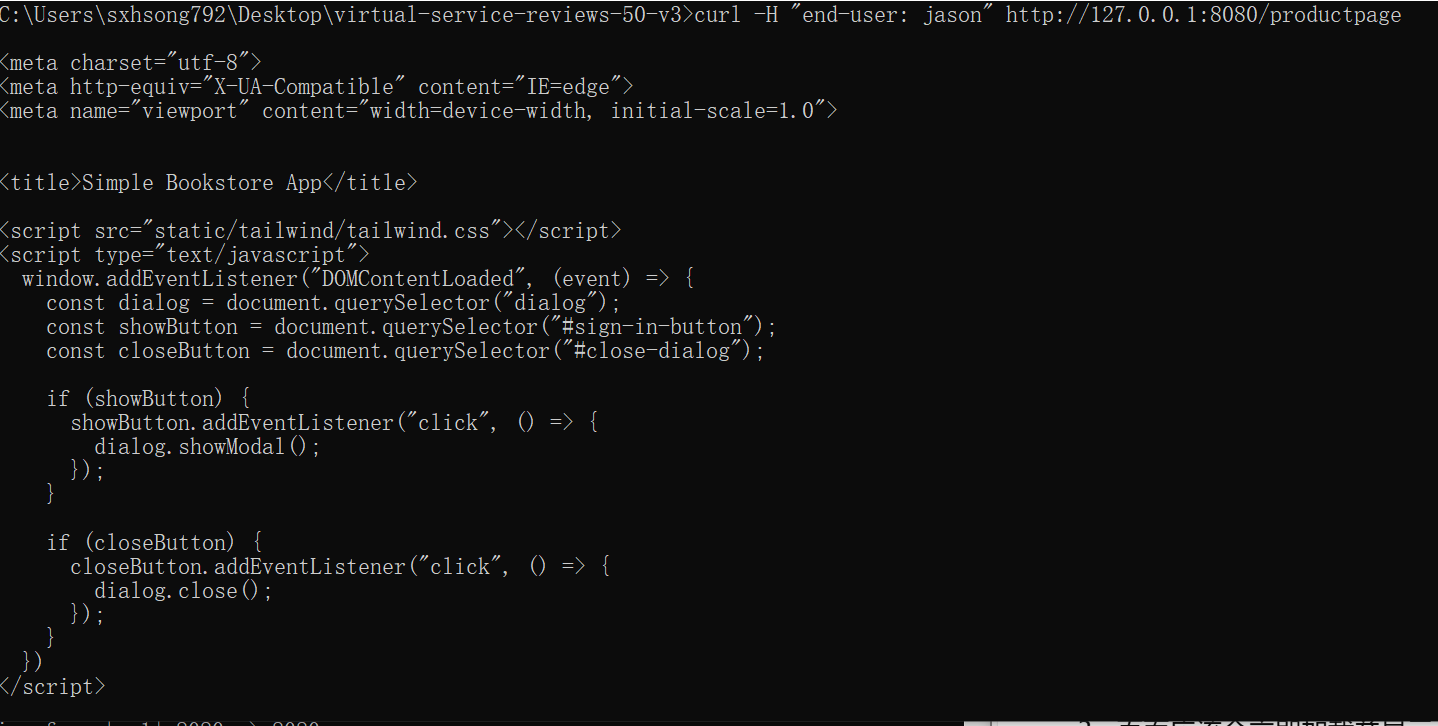




Confirm the rule was created:

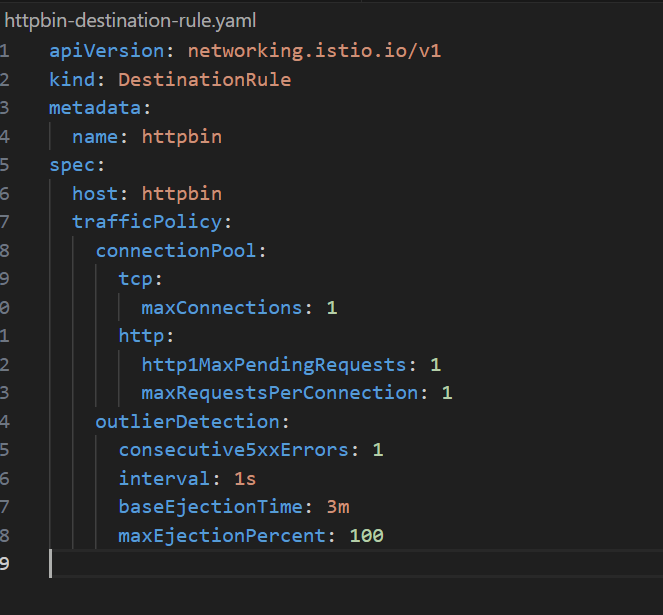


Testing the abort configuration

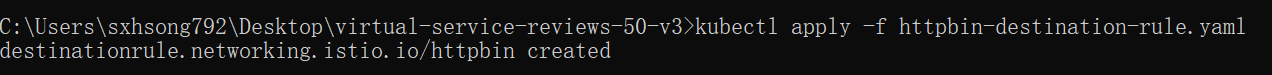


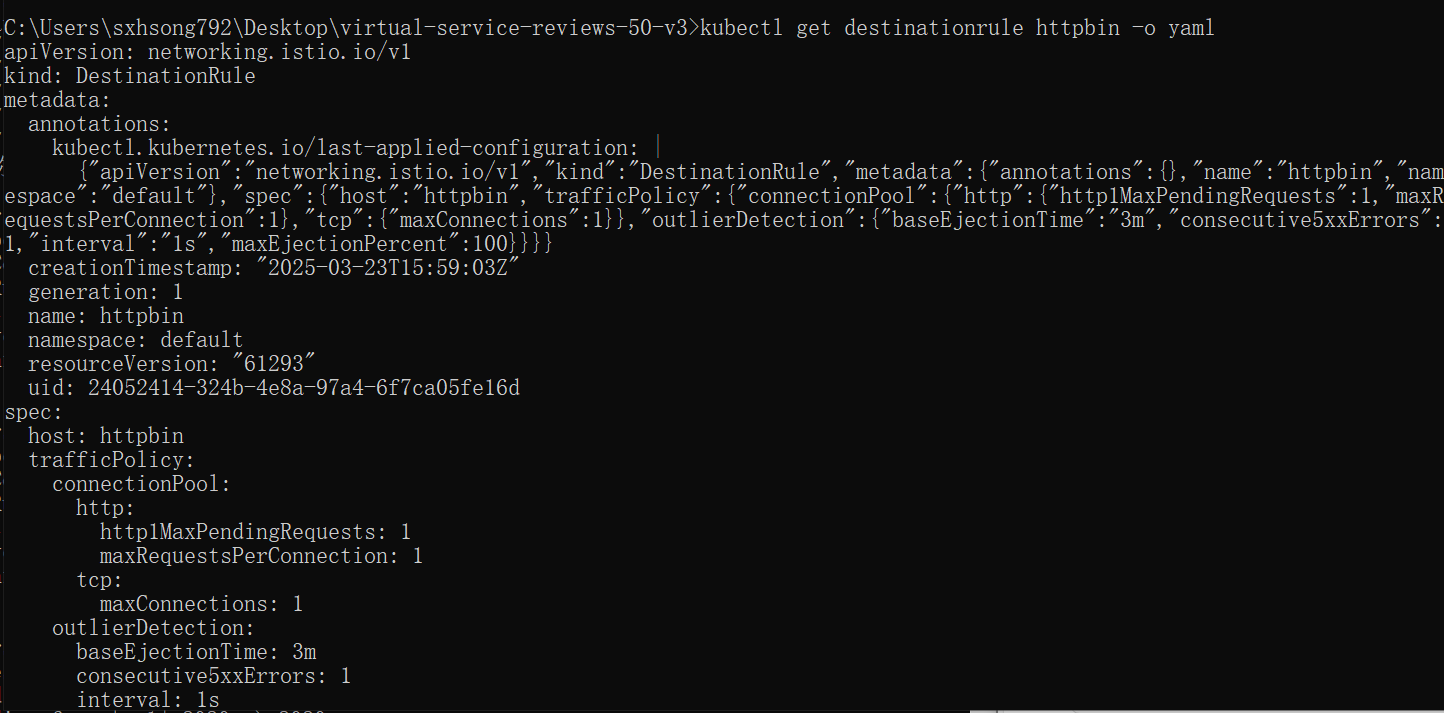
Configuring the circuit breaker

Create a destination rule to apply circuit breaking settings when calling the httpbin service

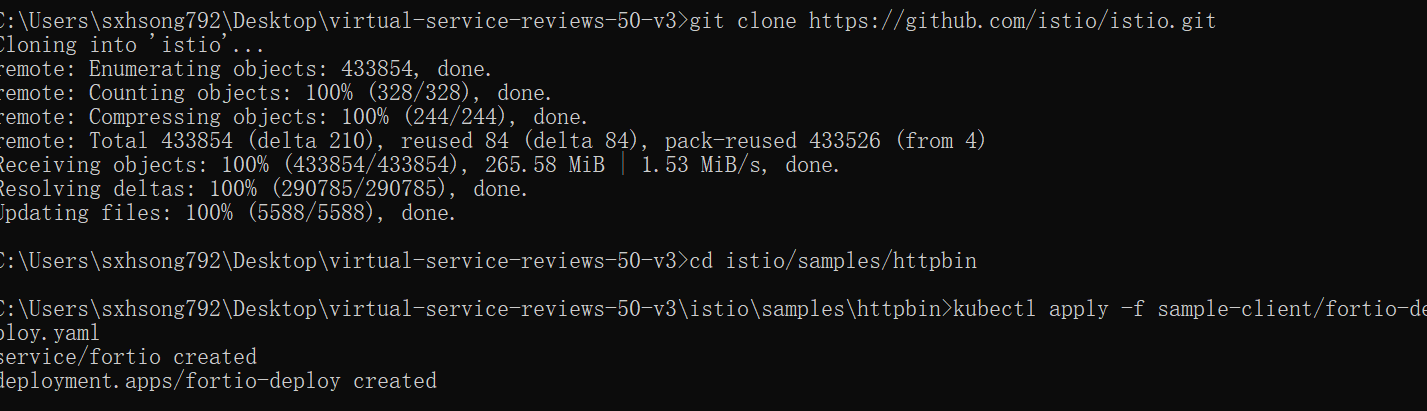


Verify the destination rule was created correctly:

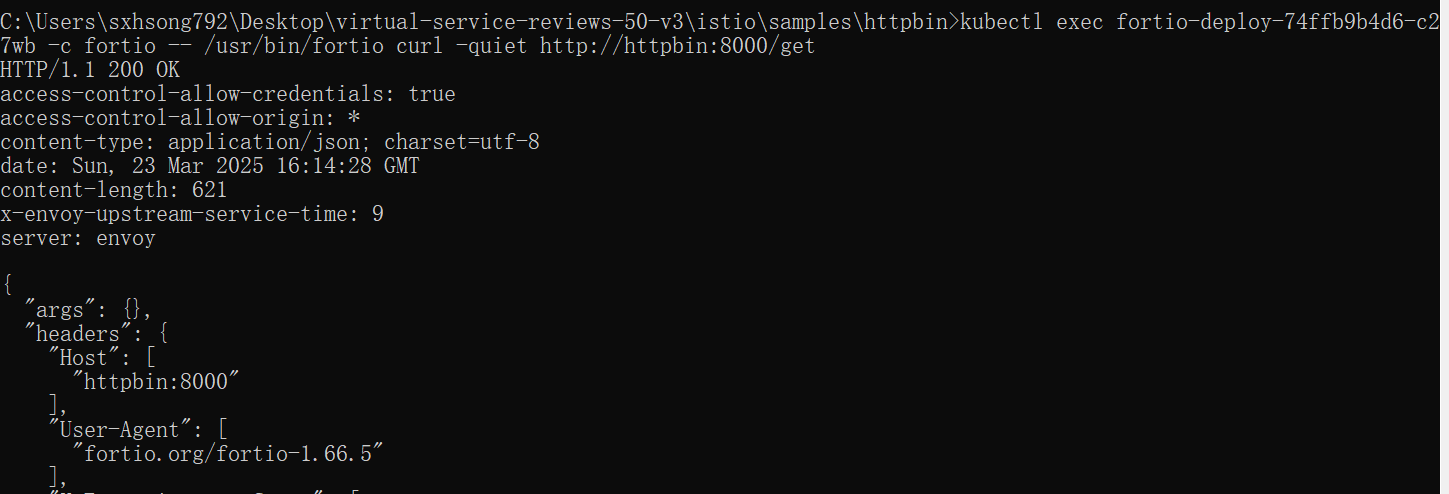


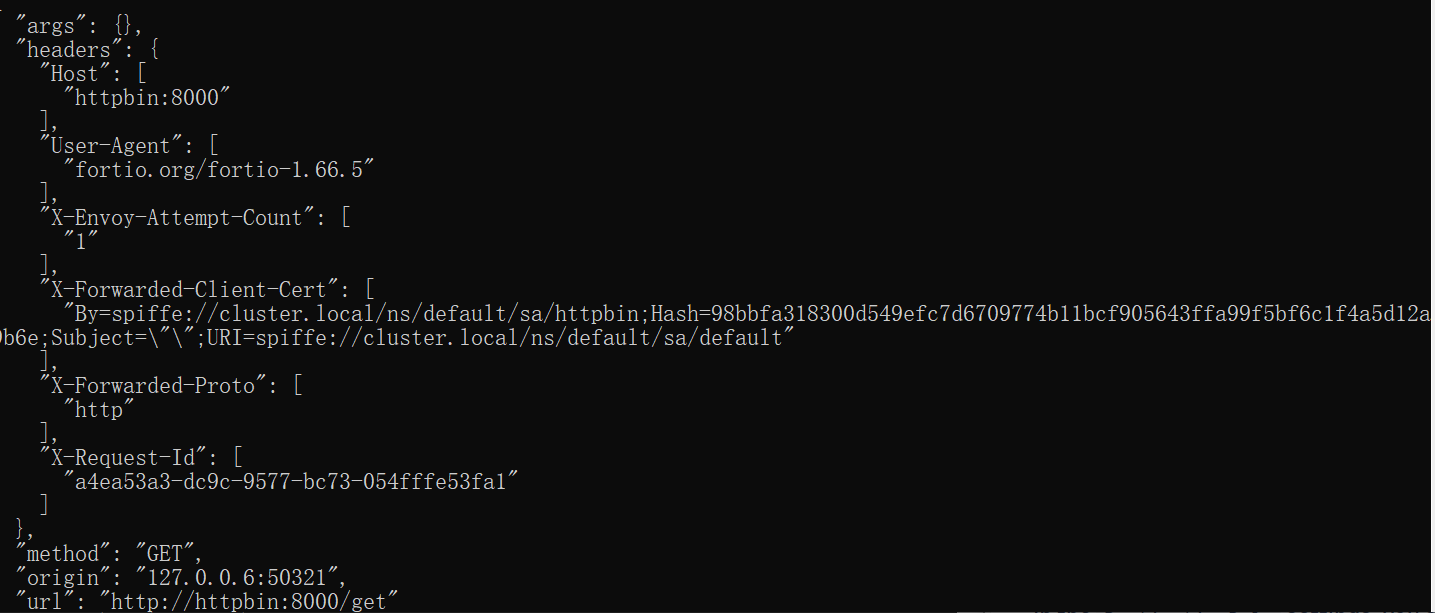


Adding a client



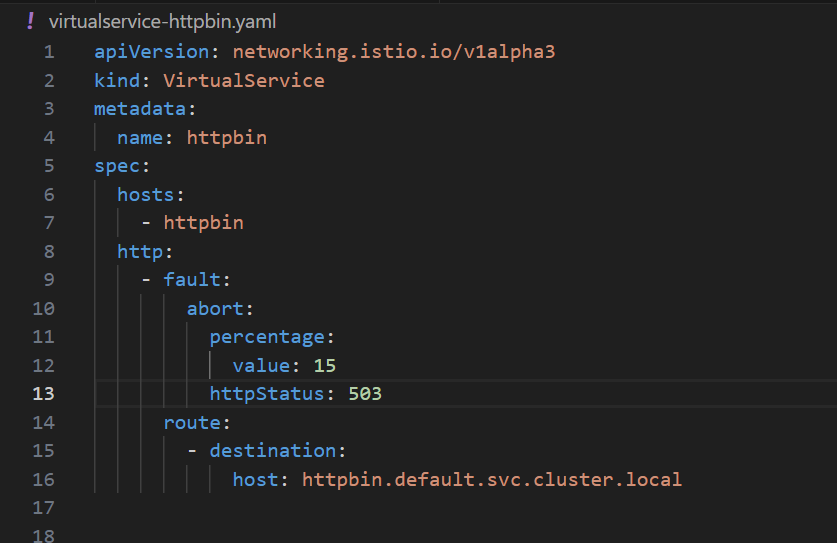
Log in to the client pod and use the fortio tool to call httpbin. Pass in curl to indicate that you just want to make one call:

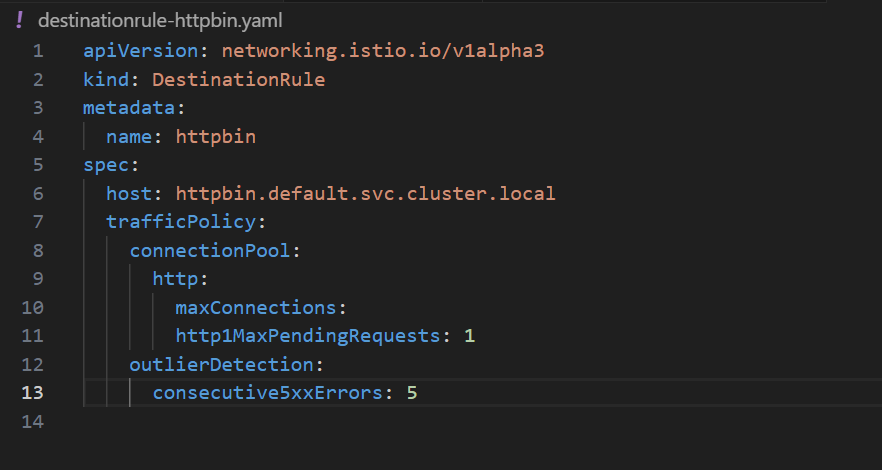


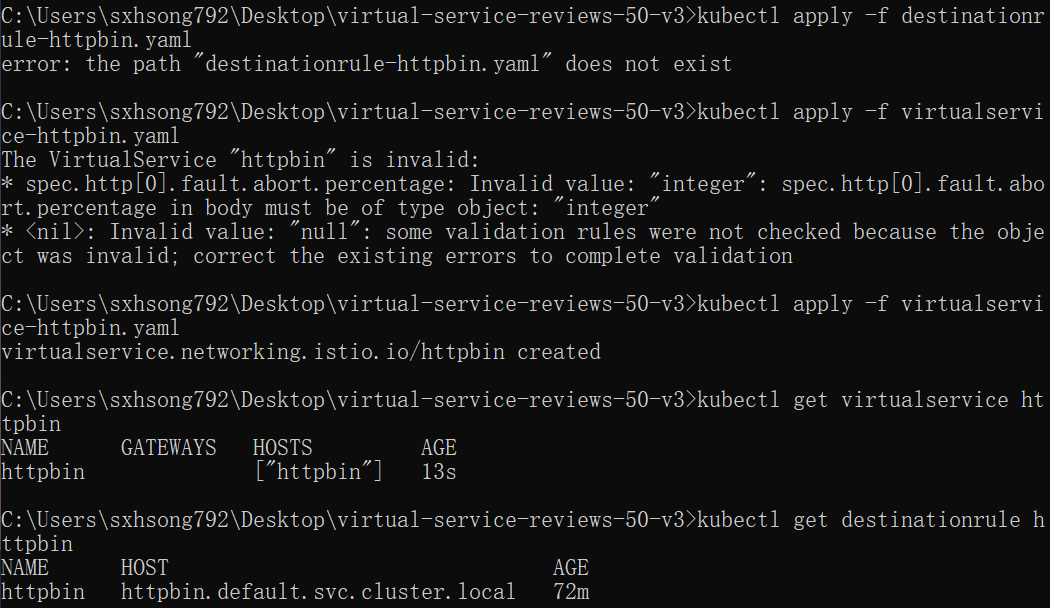


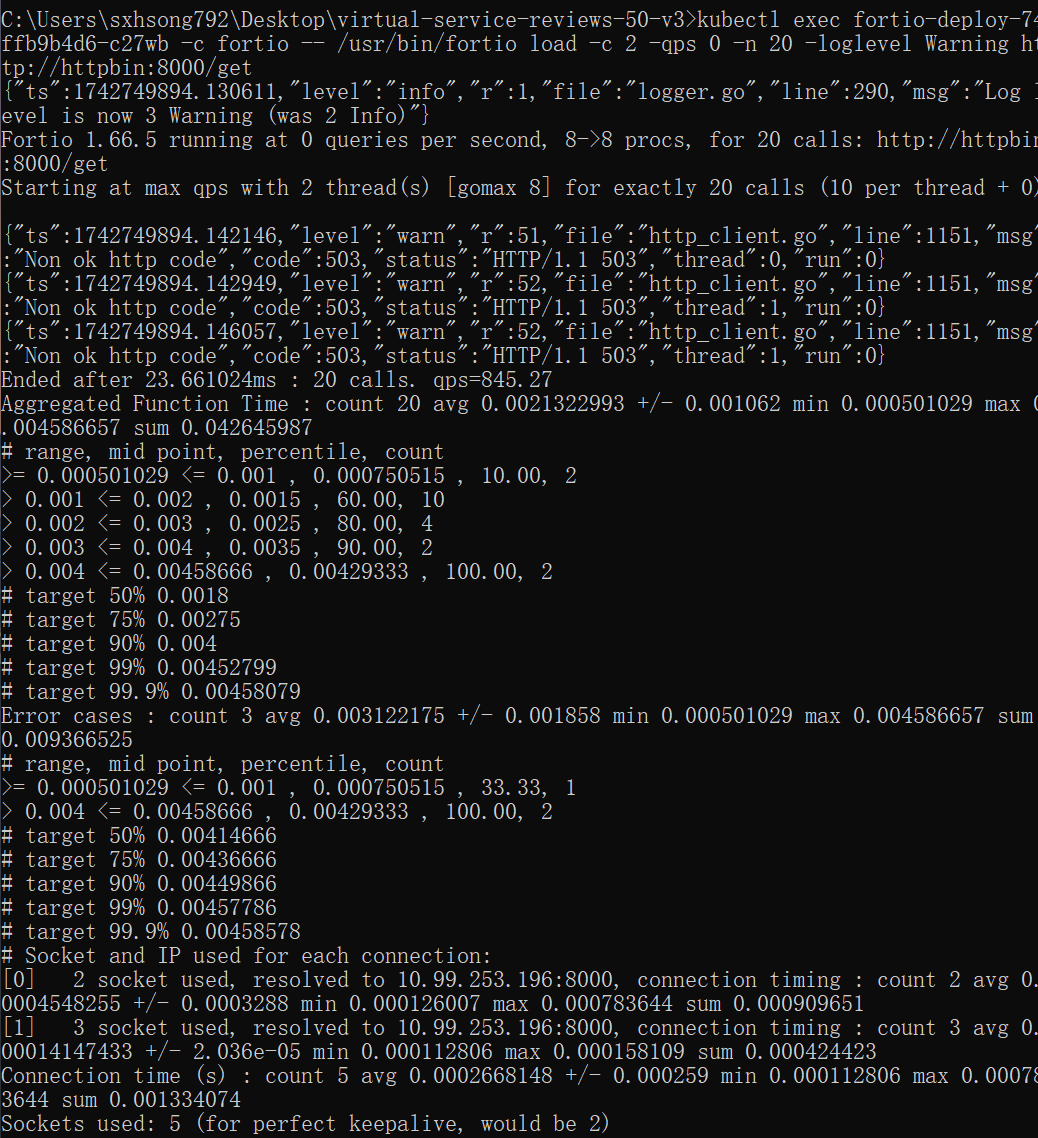
Tripping the circuit breaker

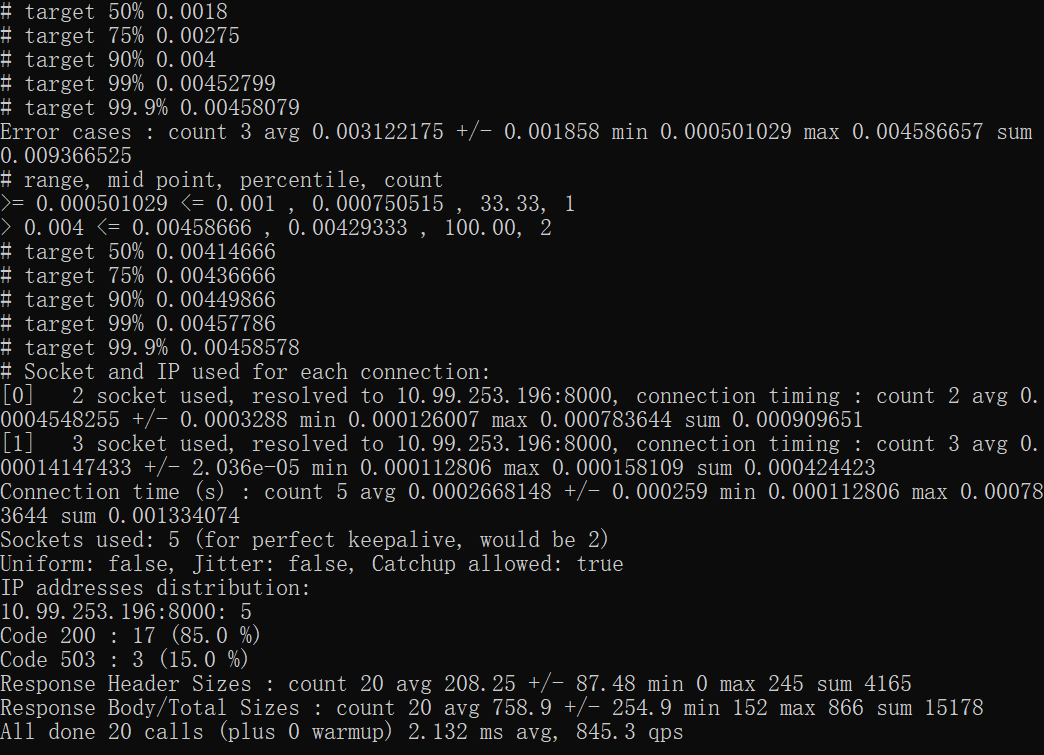
Call the service with two concurrent connections (-c 2) and send 20 requests (-n 20):



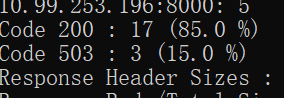




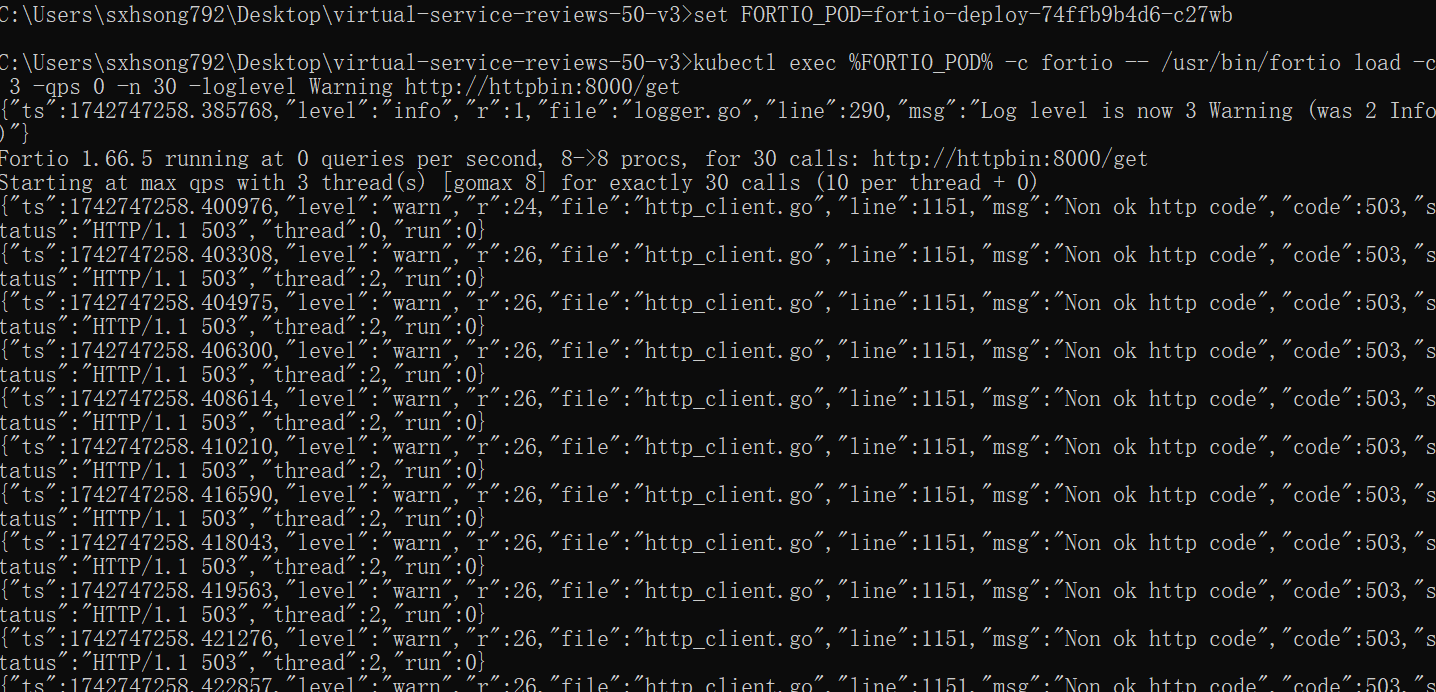


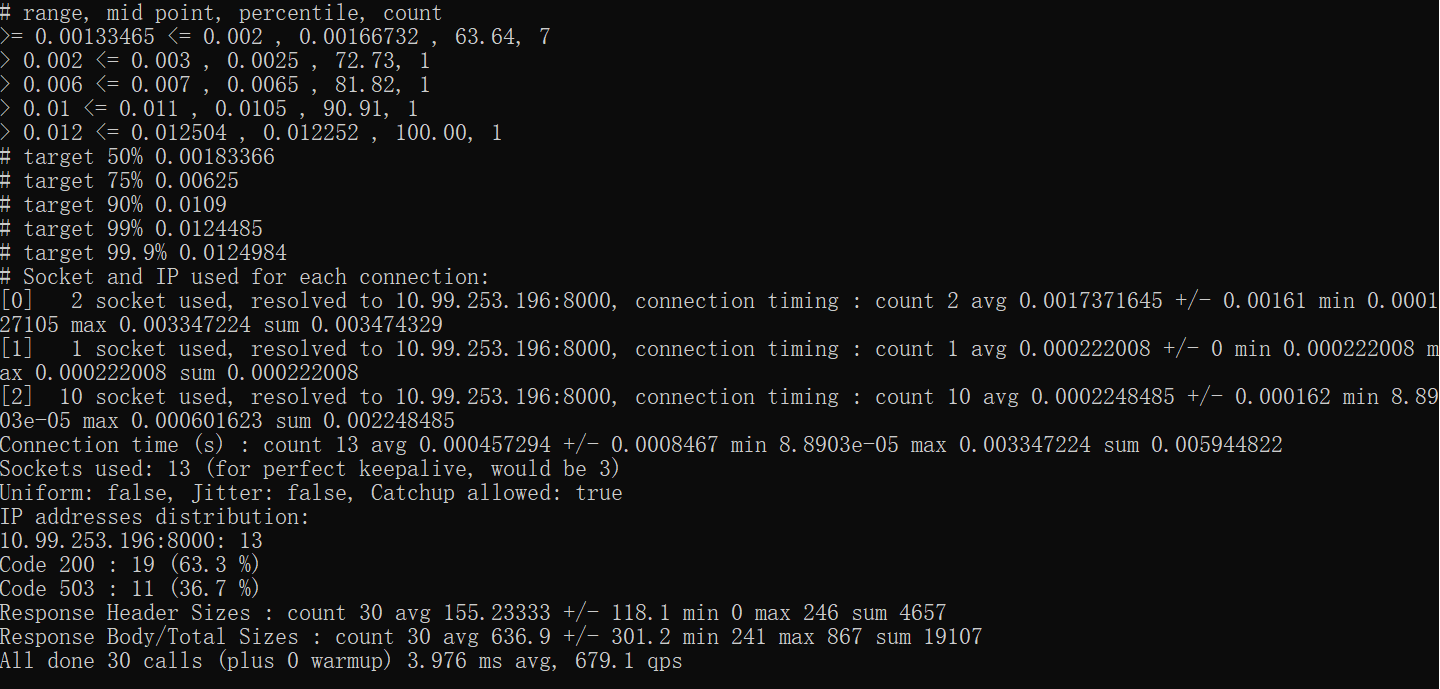


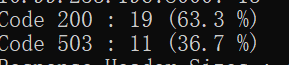
Almost all requests made it through! The istio-proxy does allow for some leeway.



Bring the number of concurrent connections up to 3







Query the istio-proxy stats to see more:

