Atit Shetty

Chaos Engineering Experiment Report

About Topology

In order to create the required topology, I have made use of three Express servers.

The first server is called as “gateway” server and is running on port 6000. The task of this server is to route the traffic to api server and perform load balancing. In order to keep things simple for the experiment I have used *bool()* method of *random-js* module, to perform load balancing.

The second server is called as “ratings” server, and it will take the role of micro-service. This service is being used by “api” server to perform certain tasks.

The third server is called as “api” server. We have three routes in this server that act as three different clusters. The ‘api’ route will serve for 99% of traffic. The ‘api-exp’ and ‘api-control’ will host the remaining traffic. ‘api-control’ will use the service provided by ratings server, however ‘api-exp’ will actually query a dead ratings server. This will help us figure if api can handle loss of third-party micro-services gracefully.

About Api server

The three “api” clusters will check the ratings server. If “ratings” is down, they will return 500 status code. In our very simplistic rendition of a complex api service, we expect the server to respond gracefully if ratings goes down.

Experiment:

For the purpose of this experiment, we are calling the gateway 100 times from *main.*js and recording the response of each request in “**report.csv”** file.

As I have mentioned before, we expect the ‘api-exp’ cluster to return 500 always, because of our simplistic rendition of ‘api’ service.

Once report.csv is generated, we will run analysis.js to check the report. If the expected status of ‘api-exp’ cluster, is different from the recorded one, we can conclude that the ‘api’ service cannot handle the loss of **ratings** micro-service gracefully.

Result:

Out of the 100 requests sent, one was routed to ‘api-exp’, one to ‘api-control’ and the rest to ‘api’.

As expected the ‘api-exp’ was able to handle the loss of service gracefully, which also shows that ‘api’ service can handle such a loss in a graceful manner.

Thoughts:

The value behind the above experiment is understanding how chaos engineering works.

In our simple rendition, we were able to setup an experiment as well as control clusters that were used to determine how gracefully ‘api’ service can handle loss of micro-services.

In actual, the ‘api’ code can be very complex. Using experimental cluster and making changes to just that cluster, e.g. blocking the gateway, etc. will help us identify if there are any issues with the service, as hundred percent of traffic will be doomed for failure.

Since we are only routing a small percentage of actual traffic, most of the users will face no glitch and we can identify any potential issues on the production itself.

Conclusion:

We can use chaos engineering, to inject faults at micro-service levels.

Using small size clusters of main service, we can route small percentage of traffic without affecting the production.

The experimental cluster can be compared with control cluster, to find any anomaly. This can be logged and reported, to improve the performance of any service.