Introduction

In chaos engineering, specifically when dealing with Kubernetes and pods, there isn't a direct equivalent of an HTTP status code that applies to pods. However, chaos engineering in Kubernetes often involves inducing faults or disruptions at the pod level to test the resilience of the system.

Here's how you can think about pod disruptions in the context of chaos engineering:

1. \*\*Simulating Pod Failures\*\*: You can simulate pod failures by deleting or evicting pods randomly or based on certain criteria during chaos engineering experiments. This simulates scenarios where pods crash or become unavailable due to underlying issues such as node failures or resource constraints.

2. \*\*Introducing Pod Latency\*\*: While not directly analogous to an HTTP status code, you can introduce latency to pod responses by delaying network traffic or adding artificial delays to pod responses. This simulates scenarios where pods experience performance degradation or network issues.

3. \*\*Testing Pod Rescheduling\*\*: Chaos engineering experiments can involve scenarios where pods are rescheduled to different nodes or experience disruptions due to Kubernetes' rescheduling mechanisms. This allows you to evaluate how well the system handles pod rescheduling events and whether it maintains service availability and performance.

4. \*\*Inducing Network Partitioning\*\*: By isolating pods from each other or introducing network partitioning between pods, you can simulate network-related failures and test how the system handles communication disruptions between services running in different pods.

While chaos engineering in Kubernetes doesn't directly use HTTP status codes at the pod level, the principles remain the same: induce controlled failures and disruptions to test the resilience of the system and validate its behavior under various failure scenarios.

Mandatory Fields

|  |  |  |
| --- | --- | --- |
| **Variables** | **Description** | **Notes** |
| TARGET\_SERVICE\_PORT | Port of the service to target | This should be the port on which the application container runs at the pod level, not at the service level. Defaults to port 80 |
| STATUS\_CODE | Modified status code for the HTTP response | If no value is provided, then a random value is selected from the list of supported values. Multiple values can be provided as comma separated, a random value from the provided list will be selected Supported values: [200, 201, 202, 204, 300, 301, 302, 304, 307, 400, 401, 403, 404, 500, 501, 502, 503, 504]. Defaults to random status code |
| MODIFY\_RESPONSE\_BODY | Whether to modify the body as per the status code provided. | If true, then the body is replaced by a default template for the status code. Defaults to true |

Optional Fields

|  |  |  |
| --- | --- | --- |
| **Variables** | **Description** | **Notes** |
| RESPONSE\_BODY | Body string to overwrite the http response body | This will be used only if MODIFY\_RESPONSE\_BODY is set to true. If no value is provided, response will be an empty body. Defaults to empty body |
| CONTENT\_ENCODING | Encoding type to compress/encodde the response body | Accepted values are: gzip, deflate, br, identity. Defaults to none (no encoding) |
| CONTENT\_TYPE | Content type of the response body | Defaults to text/plain |
| PROXY\_PORT | Port where the proxy will be listening for requests | Defaults to 20000 |
| NETWORK\_INTERFACE | Network interface to be used for the proxy | Defaults to eth0 |
| TOXICITY | Percentage of HTTP requests to be affected | Defaults to 100 |
| CONTAINER\_RUNTIME | container runtime interface for the cluster | Defaults to containerd, supported values: docker, containerd and crio for litmus and only docker for pumba LIB |
| SOCKET\_PATH | Path of the containerd/crio/docker socket file | Defaults to /run/containerd/containerd.sock |
| TOTAL\_CHAOS\_DURATION | The duration of chaos injection (seconds) | Default (60s) |
| TARGET\_PODS | Comma separated list of application pod name subjected to pod http status code chaos | If not provided, it will select target pods randomly based on provided appLabels |
| PODS\_AFFECTED\_PERC | The Percentage of total pods to target | Defaults to 0 (corresponds to 1 replica), provide numeric value only |
| LIB\_IMAGE | Image used to run the netem command | Defaults to litmuschaos/go-runner:latest |
| RAMP\_TIME | Period to wait before and after injection of chaos in sec |  |
| SEQUENCE | It defines sequence of chaos execution for multiple target pods | Default value: parallel. Supported: serial, parallel |

**Impact**

The impact of Pod HTTP Status Code chaos attack can be seen using: **On browser Inspect in Network** on the service frontend.