Improving Kubernetes Service Availability Through Chaos

Nivedita Prasad

Supervisors:
Ricardo Rocha, Spyridon Trigazis

15/09/2022
During the development process, different challenges and difficulties may arise. And handling those challenges during the development is far better than after the product is worldwide and is ready to serve. If any failure arises at that time, then it is not only a hectic process but also a costly one.

For example, Facebook went down for 6 hours on Oct 5th, 2021, not only Facebook but WhatsApp, Instagram, and Messenger. That outage cost $160 million to the global economy (source). The question is, what could be done to fix it before this kind of situation arises? Answer - Chaos Engineering
Chaos Engineering is the discipline of experimenting with system in order to build confidence in the system’s capability to withstand turbulent conditions in production.

The objective of Chaos Engineering is to cause failure on purpose to identify where and under what conditions our system can fail, and improve our capacity.

We intentionally inject fault in the system to check whether the system is resilient or not. The harder it is to disrupt the steady state, the more confidence we have in the behaviour of the system. If a weakness is uncovered, we now have a target for improvement before that behaviour manifests in the system at large.
Chaos In Practice

Animation Link

Principal of Chaos Engineering

Define Steady State

Define Hypothesis

Add Variables
Real world scenarios

Disprove the Hypothesis

Automation to run continuously

Minimize the Blast Radius

Nivedita Prasad
What is Chaos Mesh?

- Chaos Mesh is an open-source cloud-native Chaos Engineering platform that orchestrates chaos in Kubernetes environments.

- Chaos Mesh includes a fault injection method for complex systems on Kubernetes and covers faults in Pods, the network, the file system and even the kernel.

- Chaos Mesh is built on Kubernetes CRD (Custom Resource Definition). To manage different chaos experiments, Chaos Mesh defines multiple CRD types. These CRDs mainly categorized into three main fault types: basic resource faults, platform faults, and application-layer faults.
How Does It Work?

- There are three ways to interact with the API server via dashboard, kubectl or client.Apply().
- When we perform any experiment first it’ll interact with the API server afterwards Kube API server calls the controller manager that’ll take care of deletion, creation or update of the event.
- Then, it goes to the chaos daemon and it’s primarily responsible for accepting the command from the chaos controller manager. It’s run on every node also it has privileged permission mainly to interact with network devices, file systems, kernels by hacking into the target pod Namespace.
How We Implemented It?
Helm Chart

```
chaosmesh/
  .helmignore
  Chart.yaml
  values.yaml
  README.md
  script/
    install-script.sh
  templates/
    configmap.yaml
    daemonset.yaml
    network-delay.yaml
    stress-cpu.yaml
    stress-memory.yaml
    workflow.yaml
    Notes.txt
    helpers.tpl
```
**Stress CPU**

```yaml
{{- if .Values.experiments.stressCpu }}
{{- $namespaces := .Values.namespaces }}
{{- range $app := .Values.apps }}
apiVersion: chaos-mesh.org/v1alpha1
kind: StressChaos
metadata:
  name: stress-cpu
  namespace: {{ $.Release.Namespace }}
spec:
  mode: all
  selector:
    namespaces:
    - {{ $namespaces }}
    labelSelectors:
      app: "{{ $app }}"
  stressors:
    cpu:
      workers: 1
      load: 100
      duration: "30s"
{{- end }}
{{- end }}
```
Stress Memory

```yaml
apiVersion: chaos-mesh.org/v1alpha1
kind: StressChaos
metadata:
  name: stress-memory
  namespace: {{ $.Release.Namespace }}
spec:
  mode: all
  selector:
    namespaces: |
      - {{ $namespaces }}
    labelSelectors: |
      - app: "{{ $app }}"
  stressors:
    memory:
      workers: 1
      size: 50MiB
      duration: "10s"
```

Nivedita Prasad
These workers will be scheduled and run for 30s in the pod, meaning we should expect to see our Nginx pods’ CPU & Memory spike for 30s and then drop back to near 0.
That will introduce a latency of 10ms in the network of pods with labels app:nginx i.e nginx pod for the next 30 seconds.

```
{% if .Values.experiments.networkDelay %}
{% $namespaces := .Values.namespaces %}
{% range $app := .Values.apps %}
apiVersion: chaos-mesh.org/v1alpha1
kind: NetworkChaos
metadata:
  name: network-delay
  namespace: {{ $.Release.Namespace }}
spec:
  action: delay
  mode: all
  duration: '30s'
  selector:
    namespaces:
      - {{ $namespaces }}
    labelSelectors:
      app: {{ $app }}
  delay:
    latency: '10ms' #indicates the network latency
    correlation: '100' #correlation b/w the current latency and previous one
    jitter: '0ms' #indicates the range of the network policy
    direction: to

{% end %}
{% end %}
```
To perform network chaos experiment it’s required to have NET_SCH_NETEM module installed on every node.

Daemonset in Kubernetes allow you to run a pod on every node.

And, we pack the script file in the ConfigMap to installed the network simulator on every node.
WorkFlow (Pod Kill & Pod Failure)
WorkFlow- Output

Before

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>chaos-controller-manager-807006edc15-4gpc8</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2d11h</td>
</tr>
<tr>
<td>chaos-controller-manager-807006edc15-lbwf2</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2d11h</td>
</tr>
<tr>
<td>chaos-controller-manager-807006edc15-xknz2f</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2d11h</td>
</tr>
<tr>
<td>chaos-daemon-knzm</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2d11h</td>
</tr>
<tr>
<td>chaos-daemon-xm4f4</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2d11h</td>
</tr>
<tr>
<td>chaos-dashboard-806949f99b-5snw8</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2d11h</td>
</tr>
<tr>
<td>daemonset-jjthj</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2d11h</td>
</tr>
<tr>
<td>daemonset-rxdvy</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2d11h</td>
</tr>
<tr>
<td>nginx-6799fc88d8-9vtcn</td>
<td>1/1</td>
<td>Running</td>
<td>1</td>
<td>89s</td>
</tr>
<tr>
<td>nginx-6799fc88d8-cv7x</td>
<td>1/1</td>
<td>Running</td>
<td>1</td>
<td>89s</td>
</tr>
<tr>
<td>nginx-6799fc88d8-dmzhhq</td>
<td>1/1</td>
<td>Running</td>
<td>1</td>
<td>89s</td>
</tr>
<tr>
<td>nginx-6799fc88d8-gldwq</td>
<td>1/1</td>
<td>Running</td>
<td>1</td>
<td>89s</td>
</tr>
<tr>
<td>nginx-6799fc88d8-jmknl2</td>
<td>1/1</td>
<td>Running</td>
<td>1</td>
<td>89s</td>
</tr>
<tr>
<td>nginx-6799fc88d8-h5scb</td>
<td>1/1</td>
<td>Running</td>
<td>1</td>
<td>89s</td>
</tr>
<tr>
<td>nginx-6799fc88d8-zck6c</td>
<td>1/1</td>
<td>Running</td>
<td>1</td>
<td>89s</td>
</tr>
<tr>
<td>nginx-6799fc88d8-xwzdr</td>
<td>1/1</td>
<td>Running</td>
<td>1</td>
<td>89s</td>
</tr>
</tbody>
</table>

After

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>chaos-controller-manager-807006edc15-4gpc8</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2d11h</td>
</tr>
<tr>
<td>chaos-controller-manager-807006edc15-lbwf2</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2d11h</td>
</tr>
<tr>
<td>chaos-controller-manager-807006edc15-xknz2f</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2d11h</td>
</tr>
<tr>
<td>chaos-daemon-knzm</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2d11h</td>
</tr>
<tr>
<td>chaos-daemon-xm4f4</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2d11h</td>
</tr>
<tr>
<td>chaos-dashboard-806949f99b-5snw8</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2d11h</td>
</tr>
<tr>
<td>daemonset-jjthj</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2d11h</td>
</tr>
<tr>
<td>daemonset-rxdvy</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2d11h</td>
</tr>
<tr>
<td>nginx-6799fc88d8-9vtcn</td>
<td>1/1</td>
<td>Running</td>
<td>1</td>
<td>2m46s</td>
</tr>
<tr>
<td>nginx-6799fc88d8-cv7x</td>
<td>1/1</td>
<td>Running</td>
<td>1</td>
<td>2m46s</td>
</tr>
<tr>
<td>nginx-6799fc88d8-dmzhhq</td>
<td>1/1</td>
<td>Running</td>
<td>1</td>
<td>2m46s</td>
</tr>
<tr>
<td>nginx-6799fc88d8-gldwq</td>
<td>1/1</td>
<td>Running</td>
<td>1</td>
<td>2m46s</td>
</tr>
<tr>
<td>nginx-6799fc88d8-jmknl2</td>
<td>1/1</td>
<td>Running</td>
<td>1</td>
<td>2m46s</td>
</tr>
<tr>
<td>nginx-6799fc88d8-h5scb</td>
<td>1/1</td>
<td>Running</td>
<td>1</td>
<td>2m46s</td>
</tr>
<tr>
<td>nginx-6799fc88d8-zck6c</td>
<td>1/1</td>
<td>Running</td>
<td>1</td>
<td>2m46s</td>
</tr>
<tr>
<td>nginx-6799fc88d8-xwzdr</td>
<td>1/1</td>
<td>Running</td>
<td>1</td>
<td>2m46s</td>
</tr>
</tbody>
</table>

Nivedita Prasad
Future Work

- Add platform faults, and application-layer faults.
- Integrate Chaos Mesh with other tools like Grafana, Gitlab.
- Provide more user-friendly features like notifications, scheduling, and visualization.
Thank You!

Questions?

nivedita.prasad@cern.ch

niveditatrasad81@gmail.com

@NiveditaPrasa15