云原生社区Meetup

第三期·杭州站

云原生社区 Cloud Native Community | PingCAP

使用 Chaos Mesh 来保障云原生系统的健壮性

演讲人：周强

GitHub 地址：https://github.com/zhouqiang-cl

PingCAP 工程效率负责人，ChaosMesh 负责人

The incident in the production environment

Incident happens anywhere anytime

云原生社区 Cloud Native Community | PingCAP

AWS

Summary of the AWS Service Event in the Sydney Region

We'd like to share more detail about the AWS service disruption that occurred this past weekend in the AWS Sydney Region. The service disruption primarily affected EC2 instances and their associated Elastic Block Store ("EBS") volumes running in a single Availability Zone.

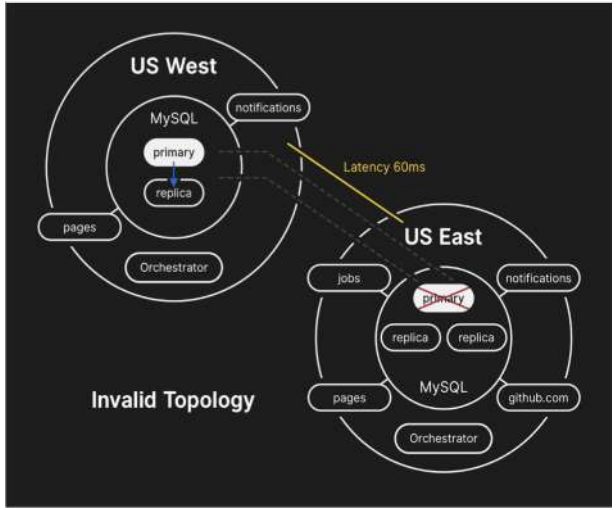
Loss of Power

At 10:25 PM PDT on June 4th, our utility provider suffered a loss of power at a regional substation as a result of severe weather in the area. This failure resulted in a total loss of utility power to multiple AWS facilities. In one of the facilities, our power redundancy didn't work as designed, and we lost power to a significant number of instances in that Availability Zone.

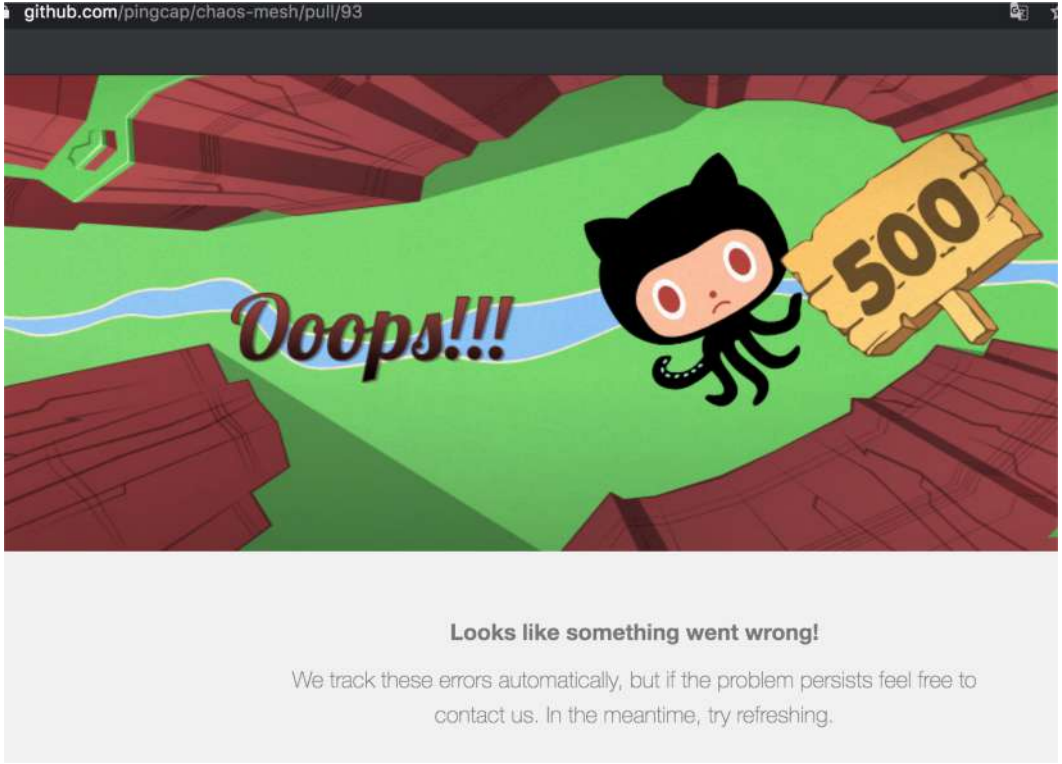
Incident happens on MySQL

Github

At 22:52 UTC on October 21, routine maintenance work to replace failing 100G optical equipment resulted in the loss of connectivity between our US East Coast network hub and our primary US East Coast data center. Connectivity between these locations was restored in 43 seconds, but this brief outage triggered a chain of events that led to 24 hours and 11 minutes of service degradation.



Incident happends on Github



Chaos Engineering

Chaos Engineering
is the discipline of experimenting on a system in order to **build confidence** in the system's capability to withstand turbulent conditions in production.

2010

2014

Netflix open position of Chaos Engineer

2015

Principles of Chaos Engineering

2016

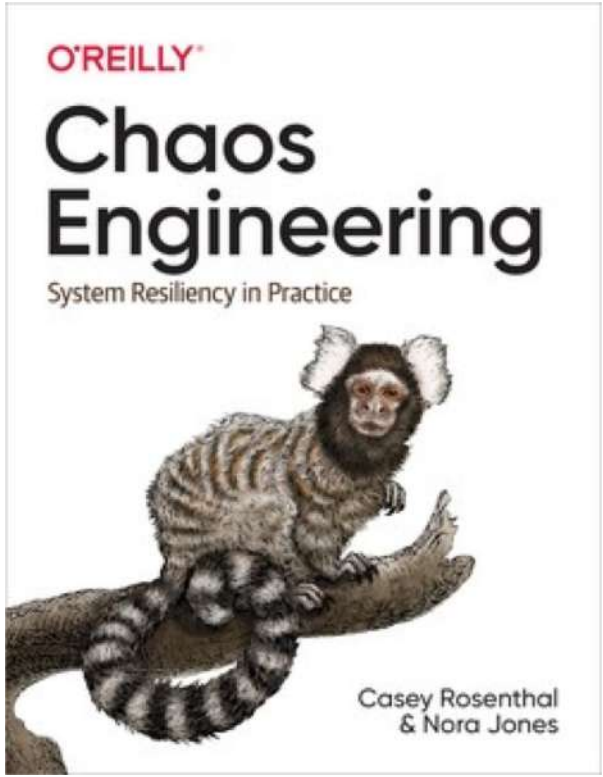
2019

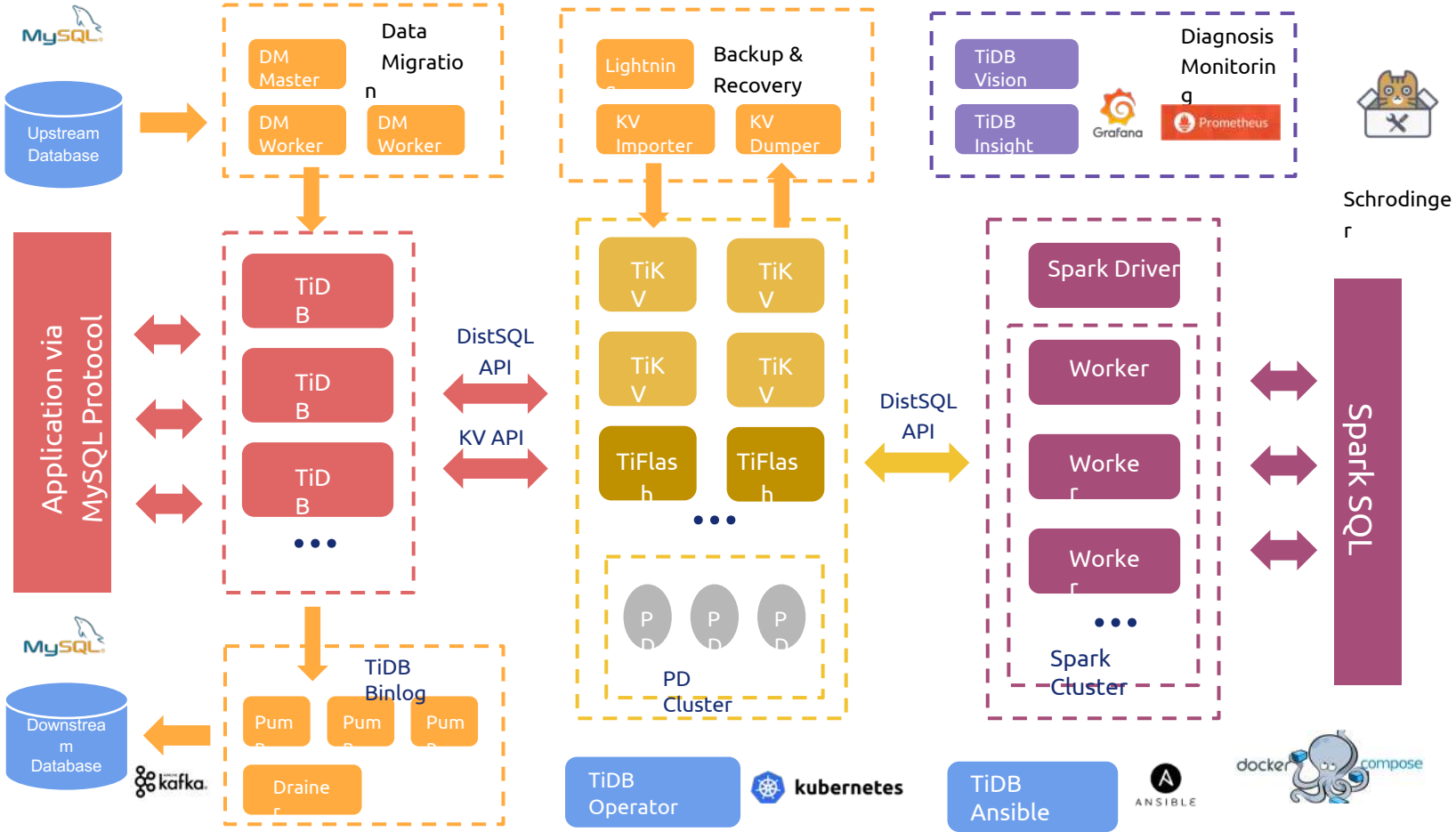
2020

Netflix / AWS / Tencent / Netease in practice of Chaos Engineering

云原生社区 Cloud Native Community | PingCAP

Chaos Engineering In TiDB

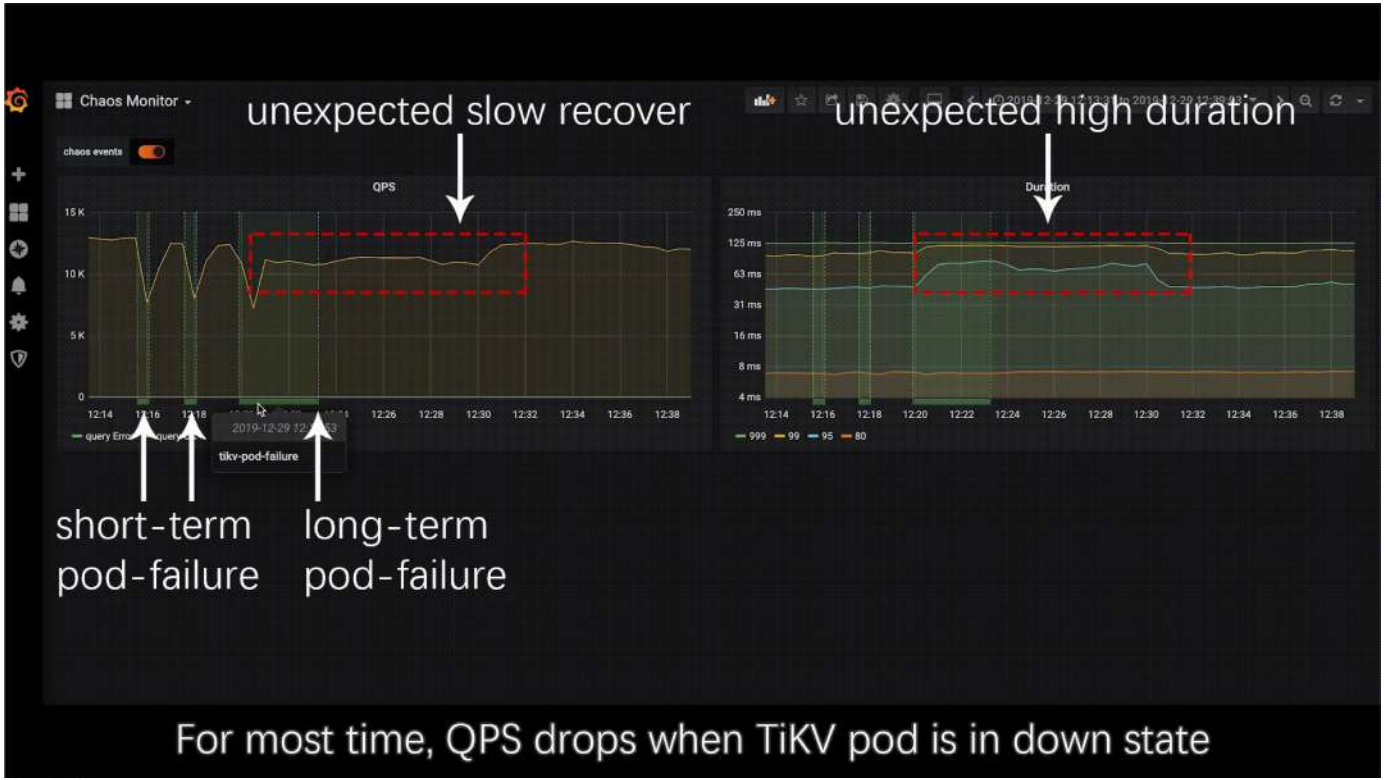



云原生社区
Cloud Native Community
PingCAP
MySQL
Upstream Database
DM Master
DM Worker
DM Worker
Data Migration
Lightning
Backup & Recovery
KV Importer
KV Dumper
TiDB Vision
TiDB Insight
Grafana
Prometheus
Diagnosis Monitoring
Schrodinger
Application via MySQL Protocol
TiDB
TiDB
TiDB
DistSQL API
KV API
TiKV
TiKV
TiKV
TiKV
TiFlash
TiFlash
PD
PD
PD
PD Cluster
DistSQL API
Spark Driver
Worker
Worker
Worker
Spark Cluster
Spark SQL
MySQL
Downstream Database
kafka
TiDB Binlog
Pump
Pump
Pump
Drainer
TiDB Operator
kubernetes
TiDB Ansible
ANSIBLE
docker compose

An experiment on TiDB

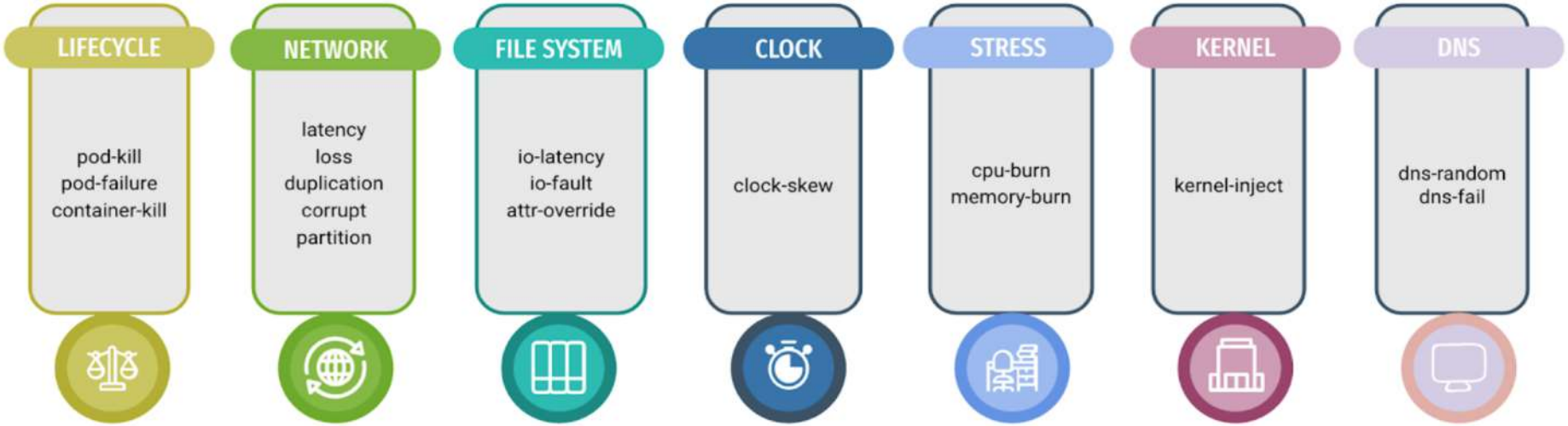
- Steady state
  - QPS metrics
- Hypothesis
  - TiDB uses Raft consensus algorithm to replicate data and provide fault-tolerance
  - Kill one TiDB instance, if the instance has a leader replica, the QPS may drop because the client can’t write data into the replica now
  - Other replicas will elect a new leader soon to service the client’s write again
  - The QPS will be recovered
- Run experiment
  - Kill one TiDB instance randomly
- Verify
  - The QPS dropped but not recovered anymore
  - A bug is found !!!

Example


Chaos Monitor
unexpected slow recover
unexpected high duration
chaos events
QPS
Duration
15 K
10 K
5 K
0
250 ms
125 ms
63 ms
31 ms
16 ms
8 ms
4 ms
12:14
12:16
12:18
12:20
12:22
12:24
12:26
12:28
12:30
12:32
12:34
12:36
12:38
tikv-pod-failure
999
99
95
80
short-term
pod-failure
long-term
pod-failure
For most time, QPS drops when TiKV pod is in down state

Chaos Mesh

Chaos In ChaosMesh



Comparison of Chaos Engineering Platforms

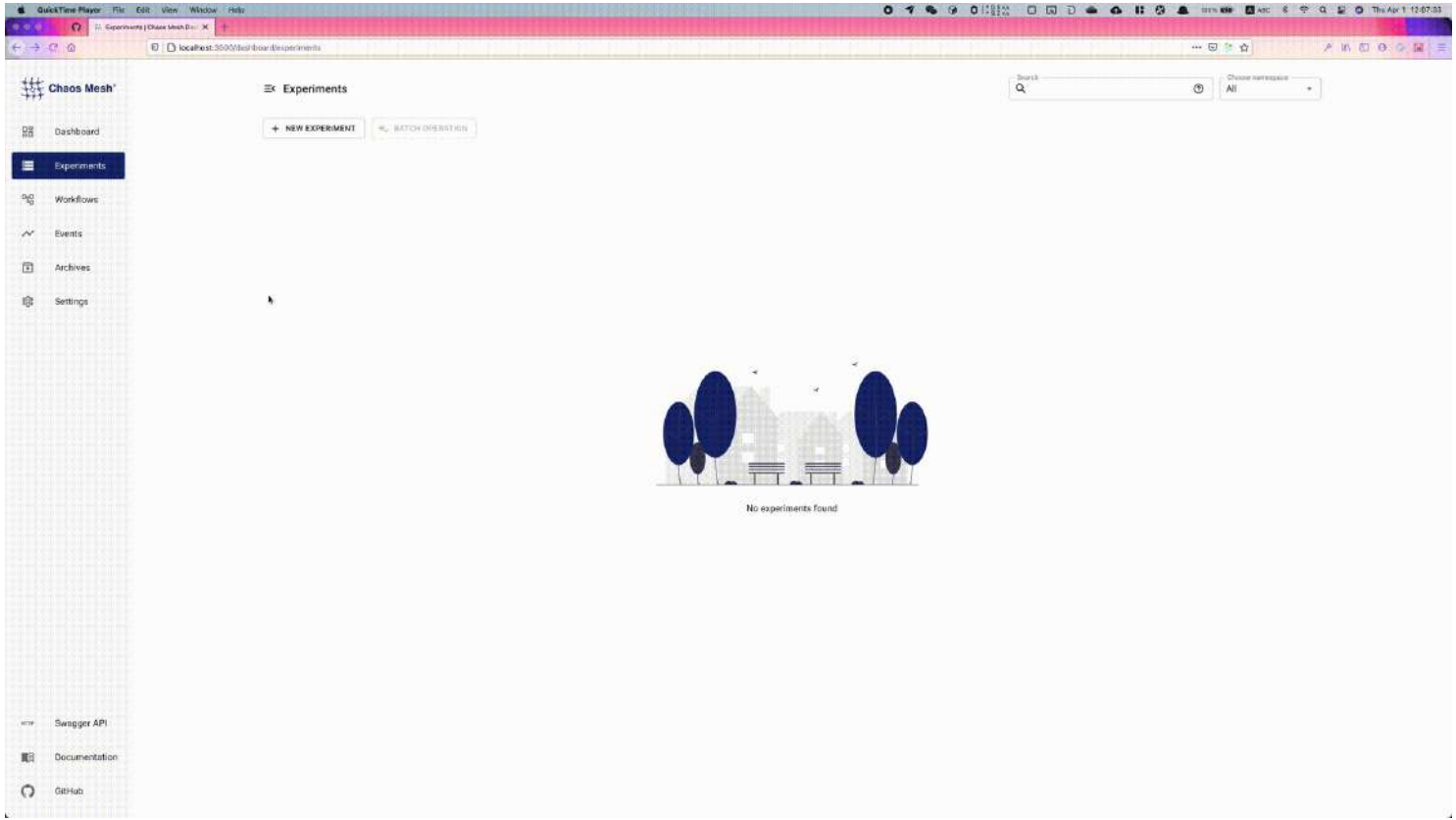
	chaos-mesh (latest)	chaosmonkey(v2.0.2)	chaosblade(v0.5.0)	chaoskube(v0.19.0)	litmus(v1.3.0)
Platform supported	VMs / K8s	VMs / Container	VMs / Container / K8s	K8s	K8s
CPU burn	✓	✗	✓	✗	✓
Mem burn	✓	✗	✓	✗	✓
container kill	✓	✓	✓	✗	✓
pod failure	✓	✗	✗	✗	✗
pod kill	✓	✗	✓	✓	✓
network partition	✓	✗	✗	✗	✗
network duplication	✓	✗	✓	✗	✗
network corrupt	✓	✗	✓	✗	✓
network loss	✓	✗	✓	✗	✓
network delay	✓	✗	✓	✗	✓
DNS failure	✓	✗	✓	✗	✗
I/O delay	✓	✗	✓	✗	✗
I/O errno	✓	✗	✓	✗	✗
Disk fill	✓	✗	✓	✗	✓
Disk loss	✓	✗	✓	✗	✓
Time skew	✓	✗	✗	✗	✗
Kernel chaos	✓	✗	✗	✗	✗
JVM Chaos	✓	✗	✓	✗	✗

Apply using kubectl

```yaml
apiVersion: chaos-mesh.org/v1alpha1
kind: NetworkChaos
metadata:
  name: network
  namespace: chaos-testing
spec:
  action: partition
  mode: one
  selector:
    labelSelectors:
      "app.kubernetes.io/component": "tikv"
  direction: to
  target:
    selector:
      labelSelectors:
        "app.kubernetes.io/component": "tikv"
    mode: one
  duration: "10s"
  scheduler:
    cron: "@every 15s"
```

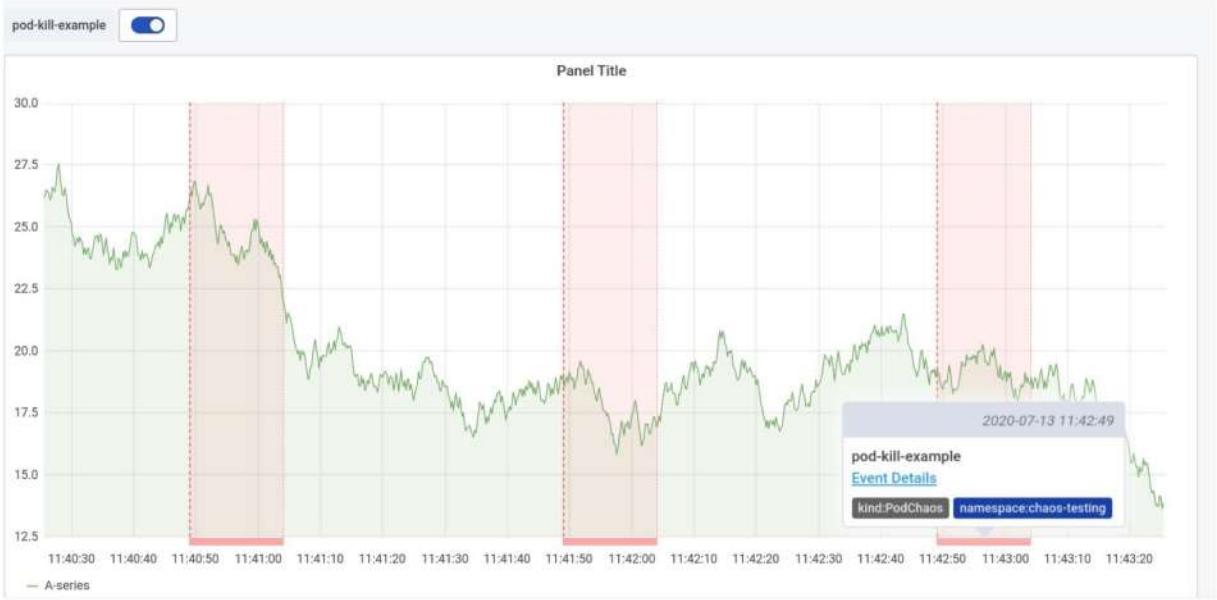
- kubectl apply -f ./network.yaml
- kubectl describe NetworkChaos network

Apply using Dashboard

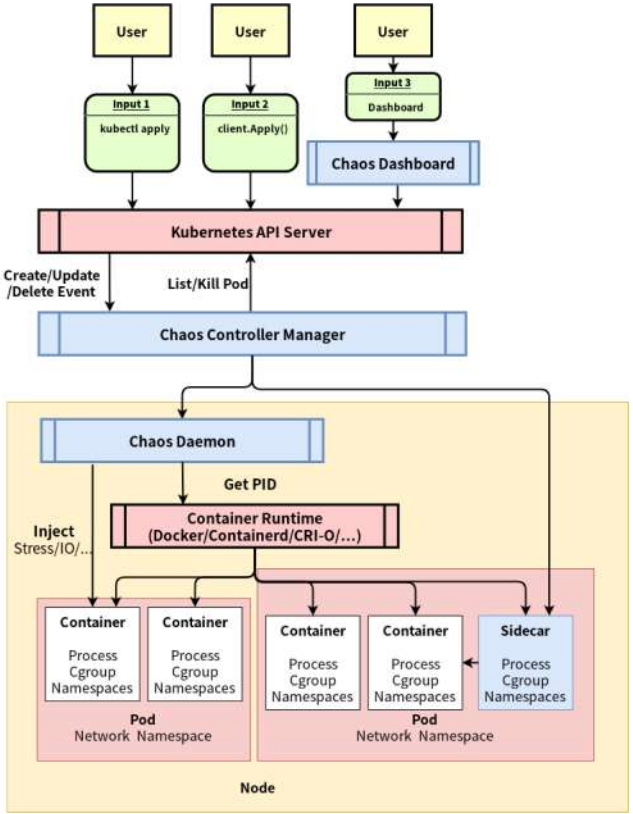


Chaos Mesh Data Source

Visualize Chaos Mesh Events with Grafana.



architecture



Chaos Mesh community

Powered by PingCAP

XPENG

- A leading Chinese electric vehicle and technology company

- Use Chaos Mesh to
  - Improve maintenance window
  - Test monitoring and alerting system
  - Simulate poor internet connection

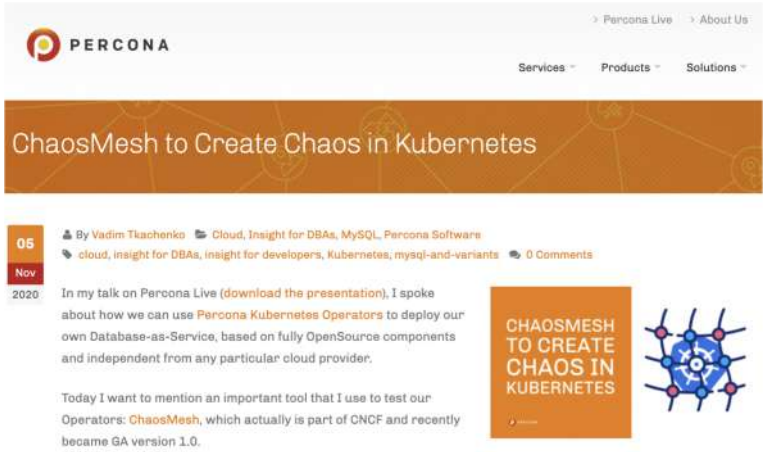
FUXI-Lab

- Testing components(redis rabbitmq scheduler) **3+**
- Testing bugs **20**

问题描述	问题原因	测试方案	解决方案
官方术语Cluster Network Partition，或Split-Brain 		600s随机kill一个pod	参照官方文档，关于"partition handling strategies"部分，涉及三种auto handling策略。 这里考虑融入autoheal策略
Error: {:aborted, {:no_exists, [:rabbit_vhost, [{{:vhost, :"$1", :_}, [], [:"$1"]}]}}} 		600s随机kill一个pod	这种情况目前看是down掉的broker node还没起来或者上没有join到集群导致
启动失败 		600s随机kill一个pod	这个问题，通过引入initContainer，对PV下的mnesia db进行清理操作，目前镜像yaml已更新，且运行后没有在遇到此类故障
Error: {:aborted, {:no_exists, [:rabbit_vhost, [{{:vhost, :"$1", :_}, [], [:"$1"]}]}}} 		600s随机kill一个pod	这种情况目前看是down掉的broker node还没起来或者上没有join到集群导致
启动失败 		600s随机kill一个pod	这个问题，通过引入initContainer，对PV下的mnesia db进行清理操作，目前镜像yaml已更新，且运行后没有在遇到此类故障

DEVOPS

Chaos Engineering on Kubernetes : Chaos Mesh Generally Available with v1.0



Active User Community

3,400+

Stars

90+

Contributors
and growing

Sandbox

Cloud Native
Computing Foundation

欢迎加入云原生社区稳定性 SIG

https://i.cloudnative.to/stability/

云原生社区 Cloud Native Community | PingCAP

云原生社区Meetup

第三期·杭州站

THANKS