Chaos Engineering:
A Five-Year Retrospective

SREcon21
October 12-14, 2021
Mikolaj Pawlikowski
Software Engineering Lead
@mikopawlikowski

Sachin Kamboj
Software Engineer
@backslash42

TechAtBloomberg.com
© 2021 Bloomberg Finance L.P. All rights reserved.
Table of Contents

- Chaos Engineering - refresher
- Evolution
- Concrete examples
- Lessons learned
- Getting started in 2021
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.”

-- Principles of Chaos Engineering
[https://principlesofchaos.org](https://principlesofchaos.org)
Chaos Engineering

"experimenting
to build confidence
to withstand turbulent conditions"

-- Principles of Chaos Engineering
https://principlesofchaos.org
Chaos Engineering

• It’s just Chaos Monkey
• Testing in prod
• Randomly breaking things in production
Evolution

https://en.wikipedia.org/wiki/Technology_adoption_life_cycle
Evolution
Evolution

- Motorola DynaTAC 8000X
- 1983-1994
- $3,995 in 1984 ($9,952 equivalent in 2020)
- 30 minutes talk time
- Store 30 phone numbers

https://en.wikipedia.org/wiki/Motorola_DynaTAC
Evolution

Innovators
Evolution

Innovators
Evolution

Innovators

Getting over the hump

Early majority

Early adopters
Evolution

- Innovators
- Early adopters
- Early majority
- Late majority
Examples from the trenches
Example 1 – Are you really highly available?
The story of the Kubernetes components

Flag for kubelet:

```go
fs.StringSliceVar(&s.APIServerList, "api-servers", []string{}, "List of Kubernetes API servers for publishing events, and reading pods and services. (ip:port), comma separated.")
```

[Links](https://github.com/kubernetes/kubernetes/issues/19152)
[Links](https://github.com/kubernetes/kubernetes/issues/18174)
Example 2 – What happens when you scale?

It’s always DNS

Client

$ nslookup foo.bar.bloomberg.com
nslookup: can't resolve '(null)': Name does not resolve

nslookup: can't resolve 'foo.bar.bloomberg.com': Name does not resolve

TechAtBloomberg.com
Example 2 – What happens when you scale?

User Datagram Protocol, Src Port: 42324, Dst Port: 53
Domain Name System (query)
  Transaction ID: 0x3bb6
  Flags: 0x0100 Standard query
    0... .... .... = Response: Message is a query
    .000 0... .... = Opcode: Standard query (0)
    .....0. .... = Truncated: Message is not truncated
    .....1 .... = Recursion desired: Do query recursively
    .... .... .0.. = Z: reserved (0)
    .... .... ...0 = Non-authenticated data: Unacceptable
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 1
Queries
  foo.bar.bloomberg.com: type A, class IN
  Name: foo.bar.bloomberg.com
Example 2 – What happens when you scale

Domain Name System (response)
Flags: 0x8380 Standard query response, No error
1... .... .... .... = Response: Message is a response
.000 0... .... .... = Opcode: Standard query (0)
.... .0... .... .... = Authoritative: Server is not an authority for domain
.... ..1... .... .... = Truncated: Message is truncated
.... ...1 .... .... = Recursion desired: Do query recursively
.... ...1... .... .... = Recursion available: Server can do recursive queries
.... .... .0... .... = Z: reserved (0)
.... .... .0... .... = Answer authenticated
.... .... ...0 .... = Non-authenticated data: Unacceptable
.... .... .... 0000 = Reply code: No error (0)

Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
Example 2 – What happens when you scale

- RFC 791 – IPv4 requires a minimum MTU of 576 bytes
- RFC 883 – DNS max packet size 512 bytes
- RFC 5966 – DNS over TCP
  - Stub resolver … MUST support TCP …
  - Stub resolver … MAY omit … TCP when designed for deployment in restricted environments where truncation can never occur
Example 3 – Can you reproduce bugs?

• [https://github.com/kubernetes/kubernetes/issues/87615](https://github.com/kubernetes/kubernetes/issues/87615)

• Bug reported on January 28, 2020
• Lots of reports, but not reliably reproducible for a whole year
• Or 1 week of chaos engineering
Example 4 – Proactively find bugs

0. Validate that all instances have the same CA fingerprint and a recent heartbeat

1. In the prepare step, each ECM instance writes the new CA with a .prepared prefix

2. On receiving a prepared message from all nodes, send a commit message

3. In the commit step, each ECM instance removes the .prepared suffix and restarts etcd

The committed message can be omitted, the heartbeat will take care of things
Example 4 – Proactively find bugs

Deliberately introduce crashes with probability $p$ at specific points in the code

1. A prepared message is **not** received from all nodes. After timeout the lock is freed and an abort message is sent

0. Validate that all instances have the same CA fingerprint and a recent heartbeat

2. The .prepared cert is deleted. No rotation occurs (and no etcd restarts) and another instance will try again at the next opportunity.
Classes of failures

- Node failures
- Network Partitions
- Dropped packets
- Limits
- Blast Radius
- Increased Latency
- Noisy neighbors
- Thundering herds
- Bad configurations
Lessons learned

• Boring is good
• Experiments tend to be cheap
• It’s just code
• A mindset shift
• “Good” <= ROI <= “Awesome”
• Retrofitting is expensive
Getting started in 2021

• **Tools readily available**
  - Open Source & also commercial

• **Books and training material also available**
  - I should know, I wrote one of these 😊

• **Cost of entry is low**
Tools

https://github.com/powerfulseal/powerfulseal
https://chaostoolkit.org
https://github.com/alexei-led/pumba
https://github.com/Shopify/toxiproxy
https://github.com/Netflix/chaosmonkey
https://byteman.jboss.org/
https://github.com/storax/kubedoom
https://litmuschaos.io/

More: https://github.com/dastergon/awesome-chaos-engineering

TechAtBloomberg.com
Chaos Engineering: Site Reliability Through Controlled Disruption

Manning

https://www.manning.com/books/chaos-engineering
Let’s connect!

Mikolaj Pawlikowski
Author "Chaos Engineering: Crash test your applications" | Engineering Lead at B...

Sachin Kamboj
Bloomberg Engineering - Software Infrastructure
Photo Credits

Photos from unsplash.com:

• Eirik Solheim - https://unsplash.com/@eirikso (old phones)
• Meritt Thomas - https://unsplash.com/@merittthomas (unicorn)
Thank you!

https://www.bloomberg.com/careers