Chaos-Driven Development: TDD for Distributed Systems

SRECon Americas 2023
March 23, 2023

Dhishan Amaranath, Cloud SRE
Tucker Vento, Resilience Engineering Team Lead

TechAtBloomberg.com

© 2023 Bloomberg Finance L.P. All rights reserved.
Who We Are

Dhishan Amaranath
Public Cloud SRE, Resilience Guild Contributor
Bloomberg

Tucker Vento
Resilience Engineering Team Lead
Bloomberg
Bloomberg: An Emphasis on Reliability

- 7,000+ software engineers
- One of the largest private networks in the world
- Over 300 billion pieces of real-time market data each day; peak of more than 15 million messages/sec (ticks)
- 2 million news stories ingested / published each day (500+ news stories ingested/sec from 125K+ sources)
- Over 2 billion messages and Instant Bloomberg (IB) chats handled daily

TechAtBloomberg.com
What have we learned from Chaos Engineering?

● Shorten the time to having a resilient system

● Designing your experiments provides half of the learning

● Codify your expectations as “unit tests” for reliability and resilience

…but why stop there?
Your Least Favorite, But Very Useful Development Practice

● Test-driven development has been “proven”\(^1\)[\(^2\)][\(^3\)] to be beneficial…
  ○ …and also not\(^4\)[\(^5\)][\(^6\)]
  ○ “It turns out that test-first does not accelerate the implementation, and the resulting programs are not more reliable, but test-first seems to support better program understanding.”\(^2\)

● Effectiveness is dependent on accuracy of the tests…
  ○ …but chaos experiments are highly accurate!

What happens when this model is applied to resilience?
Your Least Favorite, But Very Useful Development Practice

What happens when this model is applied to resilience?
Resilience-First Development

- Shorten the time to having a resilient system
  - EVEN SHORTER!

- Designing your experiments provides half of the learning
  - LEARNING SOONER!

- Codify your expectations as “unit tests” for reliability and resilience
  - RESILIENCE IS A FEATURE!
Our Journey

Building a Highly Resilient and Highly Available Log aggregation architecture on Public Cloud
Log Aggregation Architecture

- Workloads
- Logs
- Load Balancer
- Failover Route
- Health Checks
- Primary Cluster
- Log Processor
- Centralized Storage Bucket
- Failover Logic
- Failover Cluster
- Log Processor
- Health Checks
- Alarm
Designing the experiments provided half of the learning
How do we measure steady state?
Do we have the right logs and metrics collected?
Do we have redundancies for critical components?
Architecture with all the bells and whistles
“Unit Tested” Failover Logic
“Unit Tested” Failover Logic
Put Your Designs to the Test
Confronting Our Design Choices

● Are we too hasty to failover?
  ○ Node Failure and Pod Failure experiments

● Are timestamps a reliable file name?
  ○ Time Offset and Time Skew experiments

● Are we using the right defaults?
  ○ Application Failures
Knowing the (Service) Limits
SLOs & Application Limitations

Retry Logic

Memory Buffer
SLOs & Platform Limitations

Nodes goes down

Alarm Triggers

Delays
Observability: A Force Multiplier!

How can you make informed decisions about a distributed system without seeing inside that system?

- Validate your early choices in metrics and service-level indicators
- Determine initial alerting decisions based on results, not guesses
- Peer inside the “black box” components in your system
Observability: A Force Multiplier!

How can you make informed decisions about a distributed system without seeing inside that system?

Better Observability × Earlier in Development

More Time Spent Making Better Decisions
Setting Up Future Success

● Early understanding of SLO factors

● A scientific method for change management
  ○ Failure “units” are now reliability regression checks for the system
  ○ A/B validation for design changes, alerting changes, configuration changes…
  ○ Automation is your friend!

● Prepared for production gamedays
Where do we go from here?

Resilience checklist for new system design:

➔ Pod / Node failure
➔ Regional failure
➔ Network black holes
➔ Network latency
➔ Time offset / clock skew
➔ CPU / Memory / I/O stress
➔ API access denial
Where do you go from here?

- Resilience stress tests should be part of certifying your distributed systems’ designs

- Day-1 chaos accelerates identification of:
  - Factors that influence your SLIs
  - Design defects
  - Missing metrics or other health indicators

- Break things, learn more, and have fun!
Thank you!

We are hiring: bloomberg.com/engineering
Sources


TechAtBloomberg.com