# Chaos-Driven Development: TDD for Distributed Systems

Bloomberg

Engineering

SRECon Americas 2023
March 23, 2023

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

TechAtBloomberg.com

# 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



Bloomberg

Engineering

# 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]

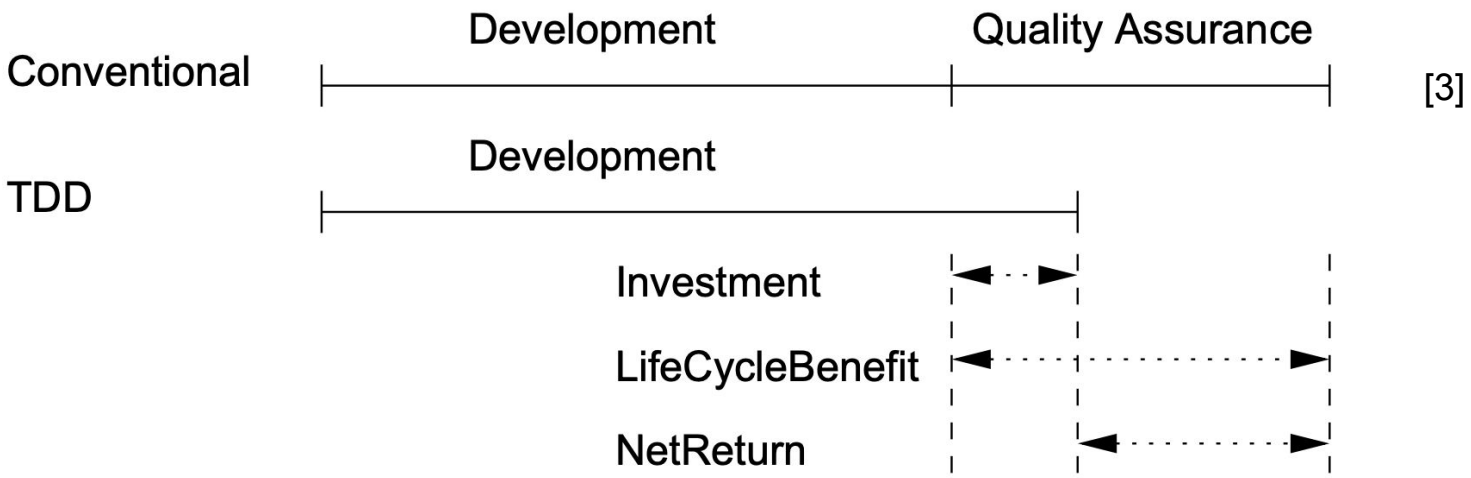
- Effectiveness is dependent on accuracy of the tests…
  - …but chaos experiments are highly accurate!

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?

TechAtBloomberg.com



Bloomberg

Engineering

# 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

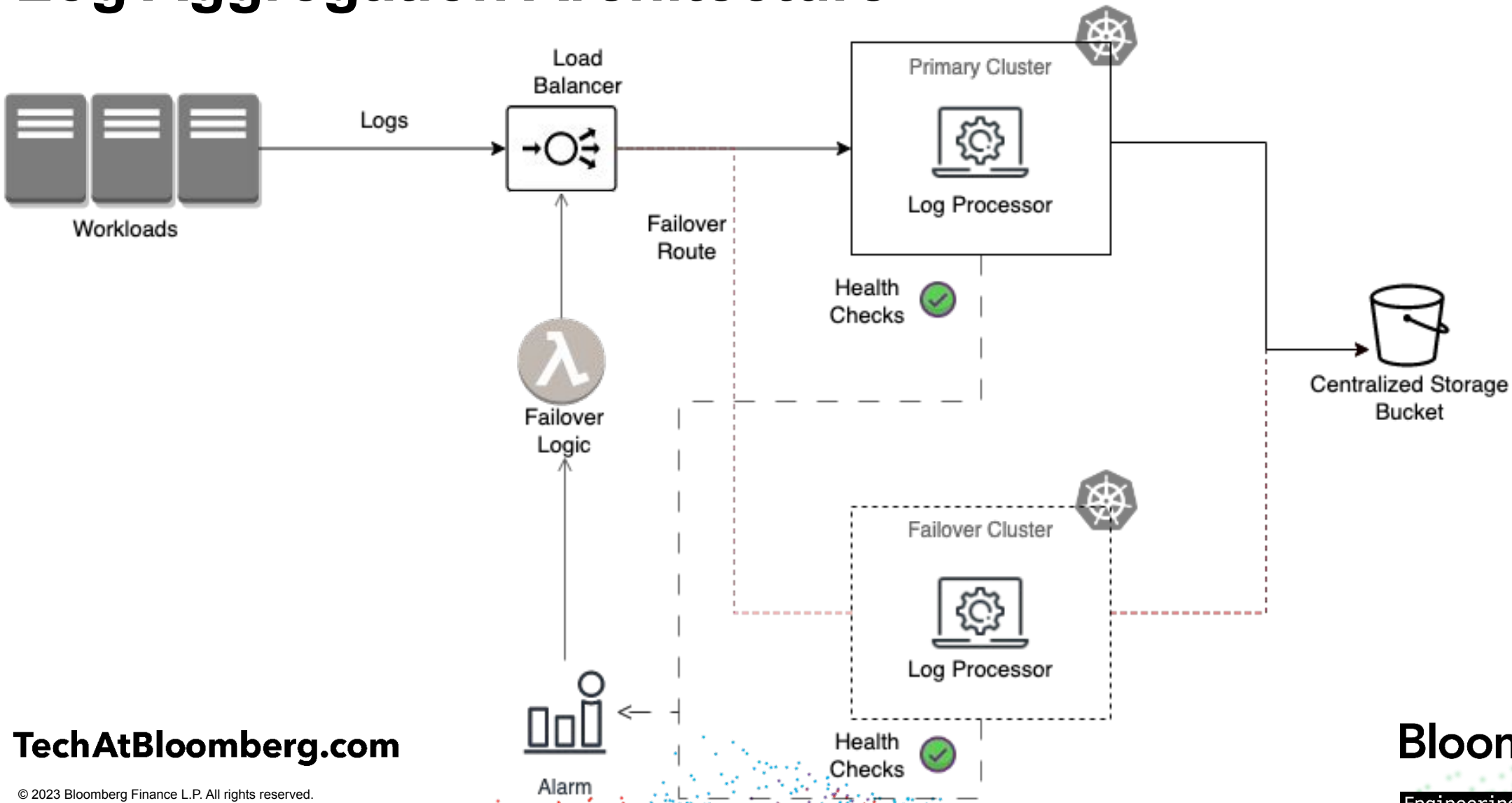
TechAtBloomberg.com



Bloomberg

Engineering

# Log Aggregation Architecture




TechAtBloomberg.com



Bloomberg

Engineering

# Designing the experiments provided half of the learning

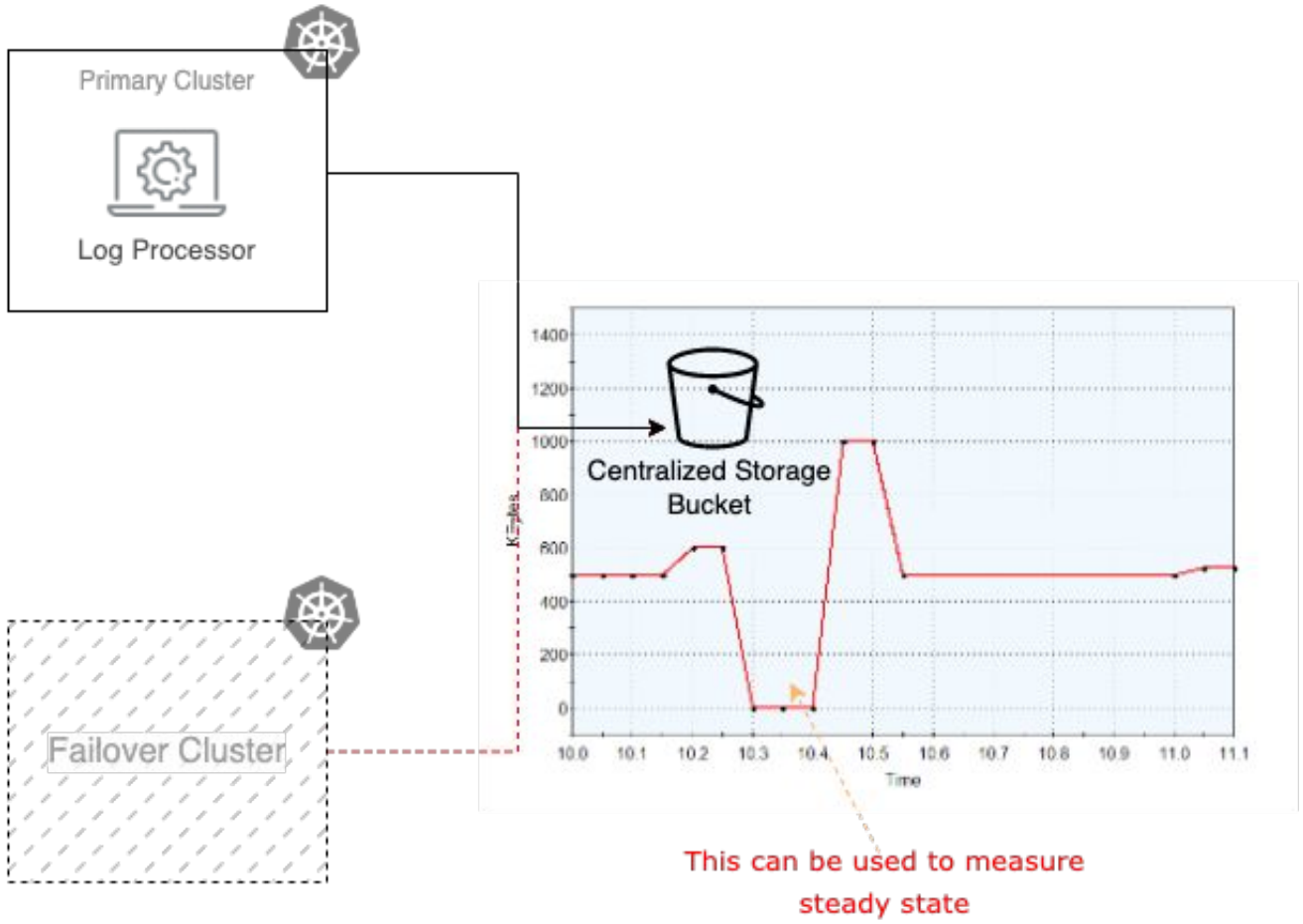
TechAtBloomberg.com



Bloomberg

Engineering

# How do we measure steady state?



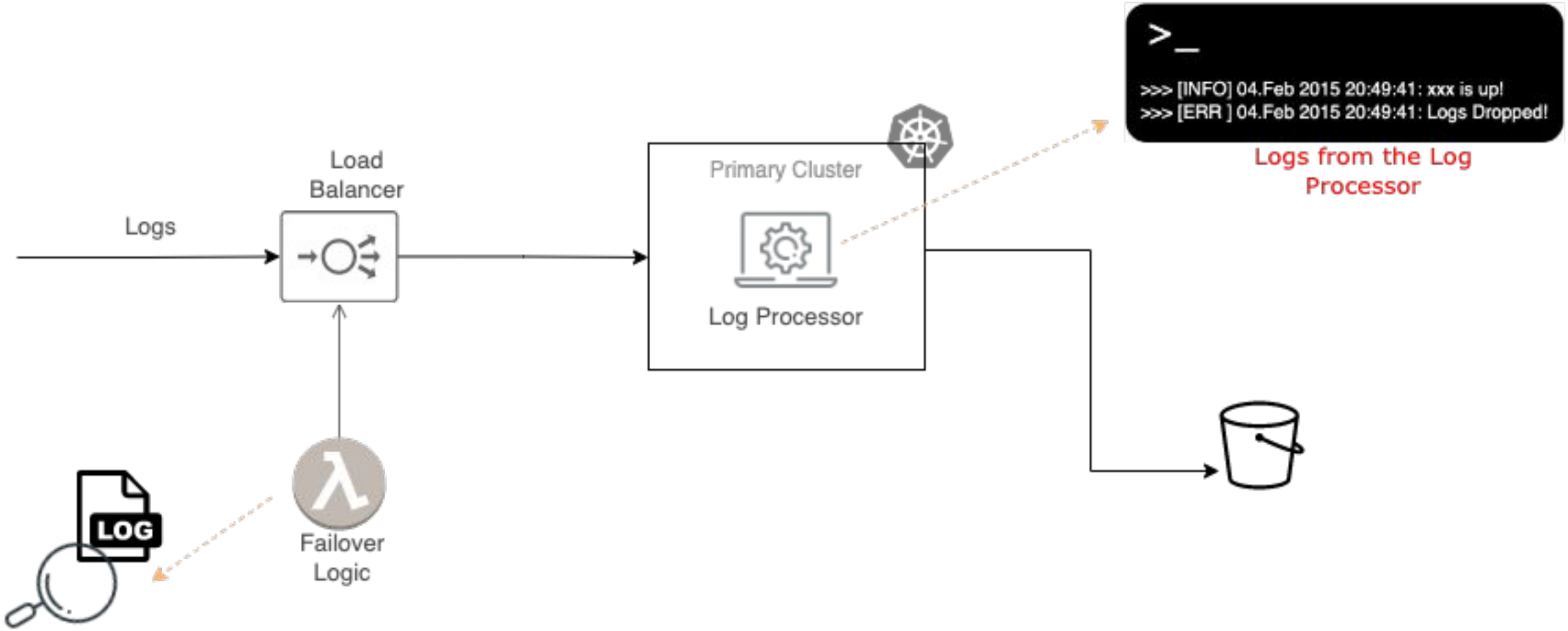

TechAtBloomberg.com



Bloomberg

Engineering

# Do we have the right logs and metrics collected?



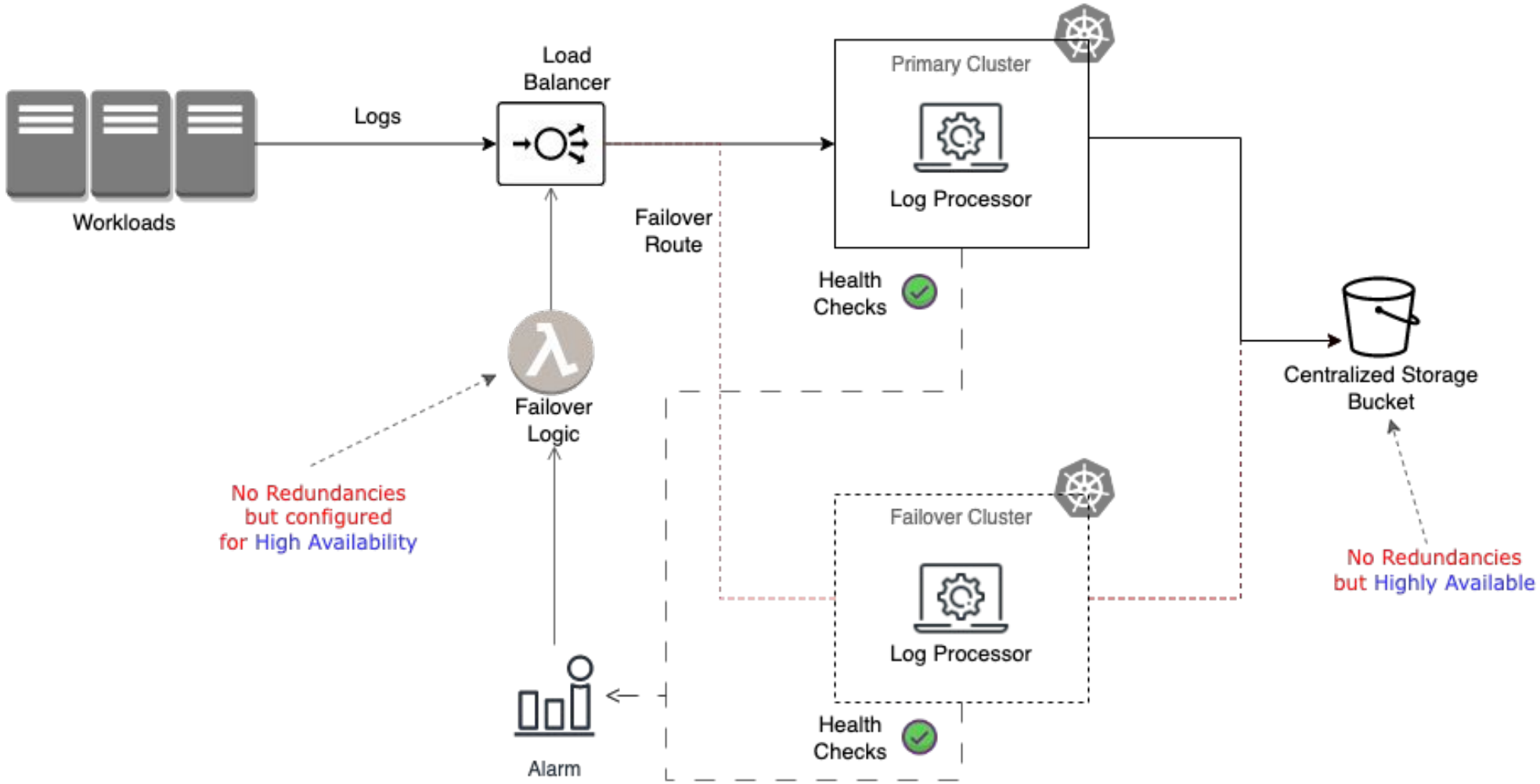

TechAtBloomberg.com



Bloomberg

Engineering

# Do we have redundancies for critical components?



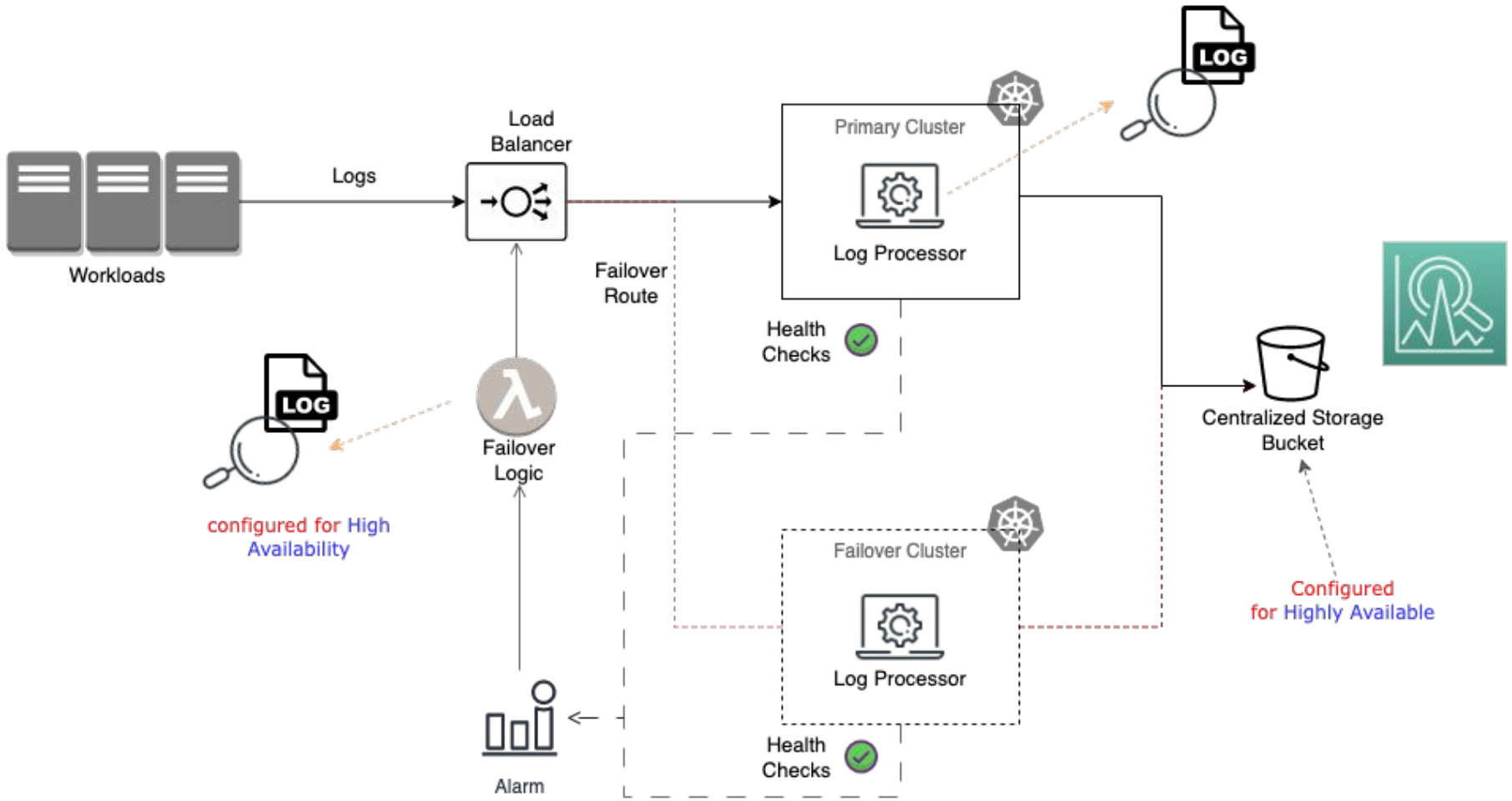

TechAtBloomberg.com



Bloomberg

Engineering

# Architecture with all the bells and whistles



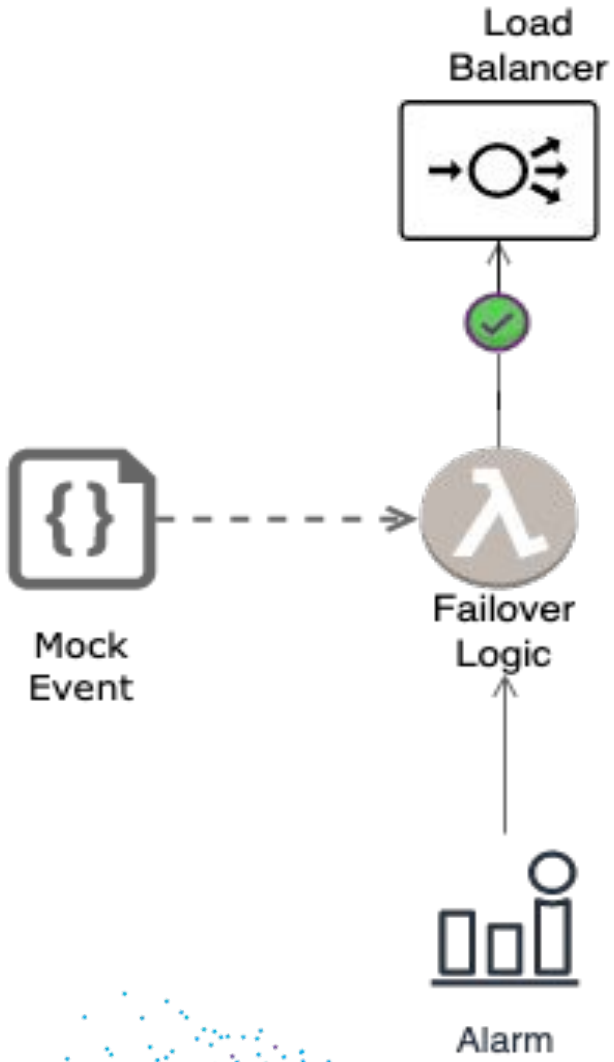

TechAtBloomberg.com



Bloomberg

Engineering

# “Unit Tested” Failover Logic



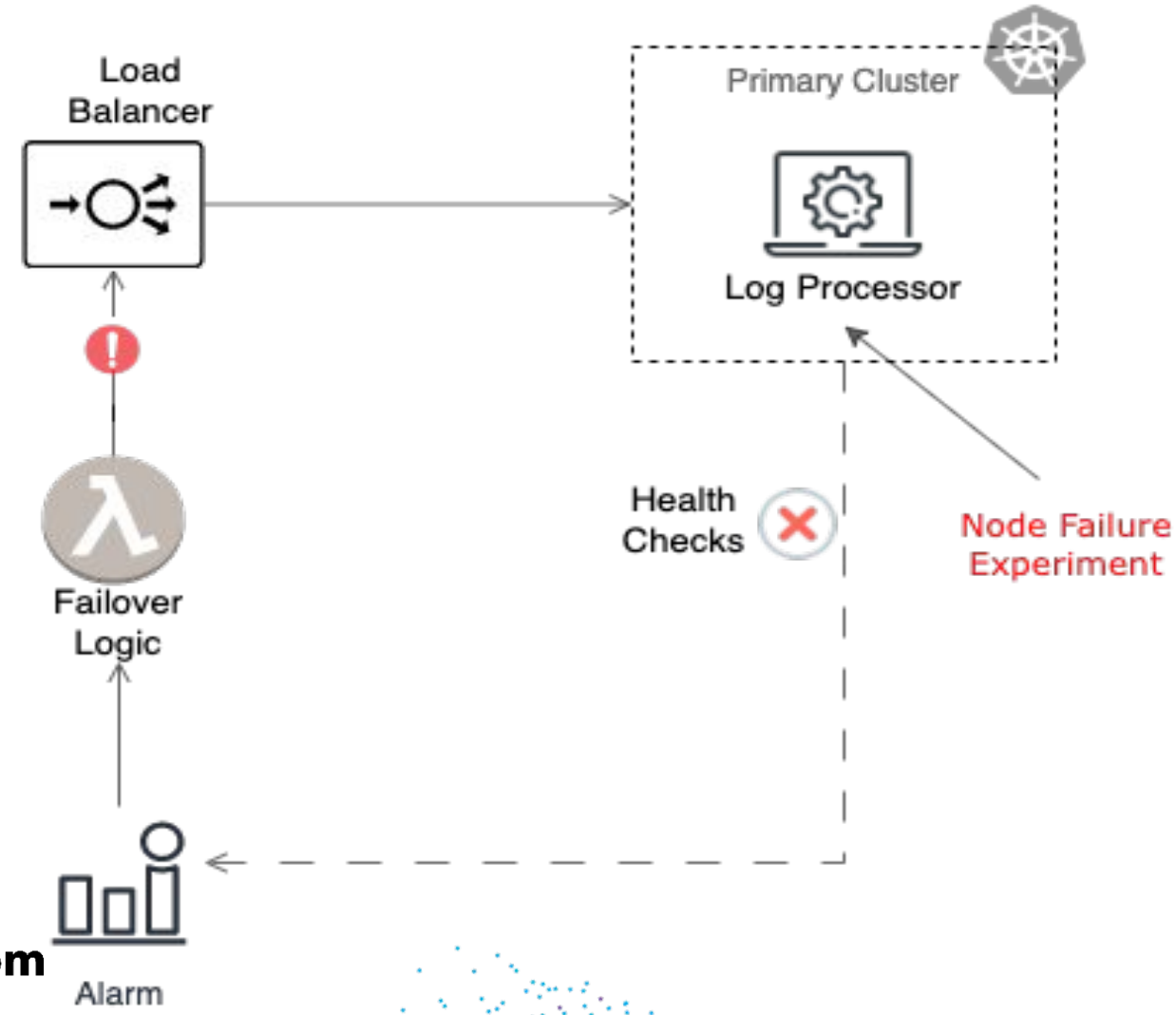

TechAtBloomberg.com



Bloomberg

Engineering

# “Unit Tested” Failover Logic




TechAtBloomberg.com



Bloomberg

Engineering

# Put Your Designs to the Test

TechAtBloomberg.com



Bloomberg

Engineering

# 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


TechAtBloomberg.com



Bloomberg

Engineering

# Knowing the (Service) Limits

TechAtBloomberg.com



Bloomberg

Engineering

# SLOs & Application Limitations

Retry Logic

Memory Buffer

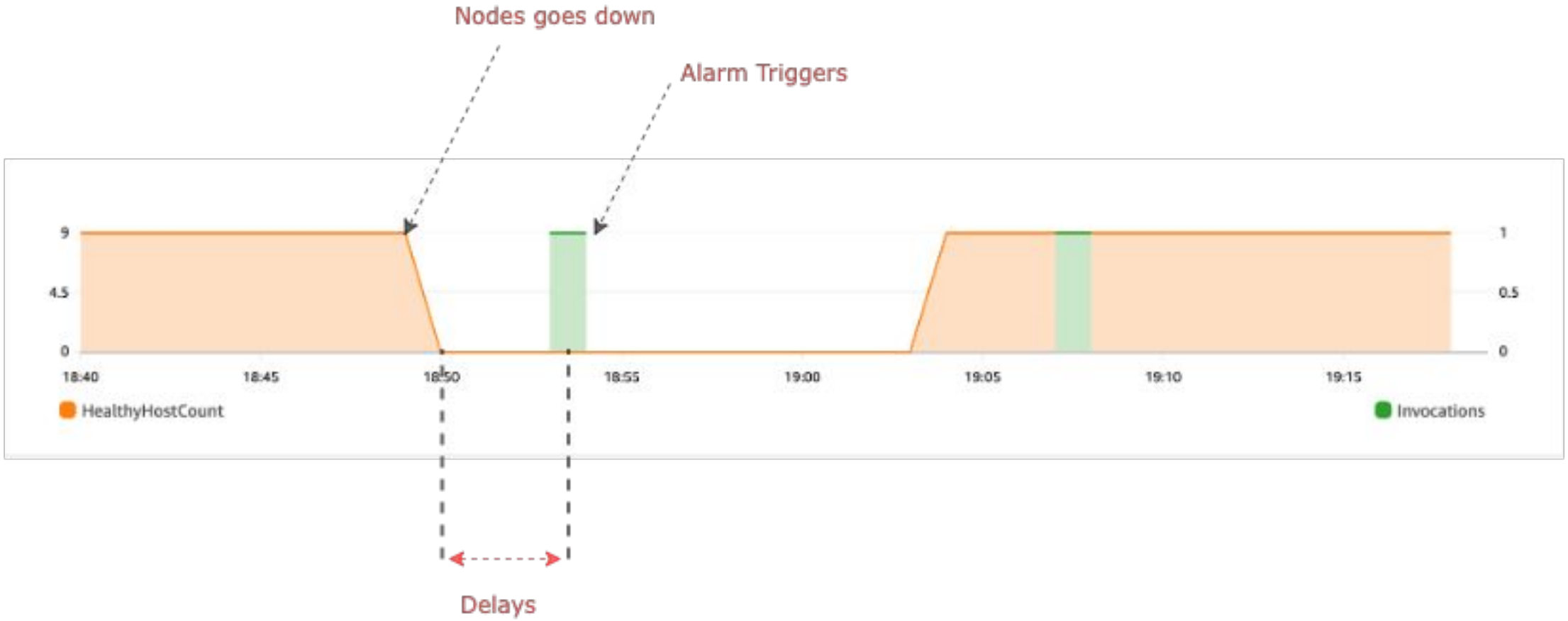
TechAtBloomberg.com



Bloomberg

Engineering

# SLOs & Platform Limitations




TechAtBloomberg.com



Bloomberg

Engineering

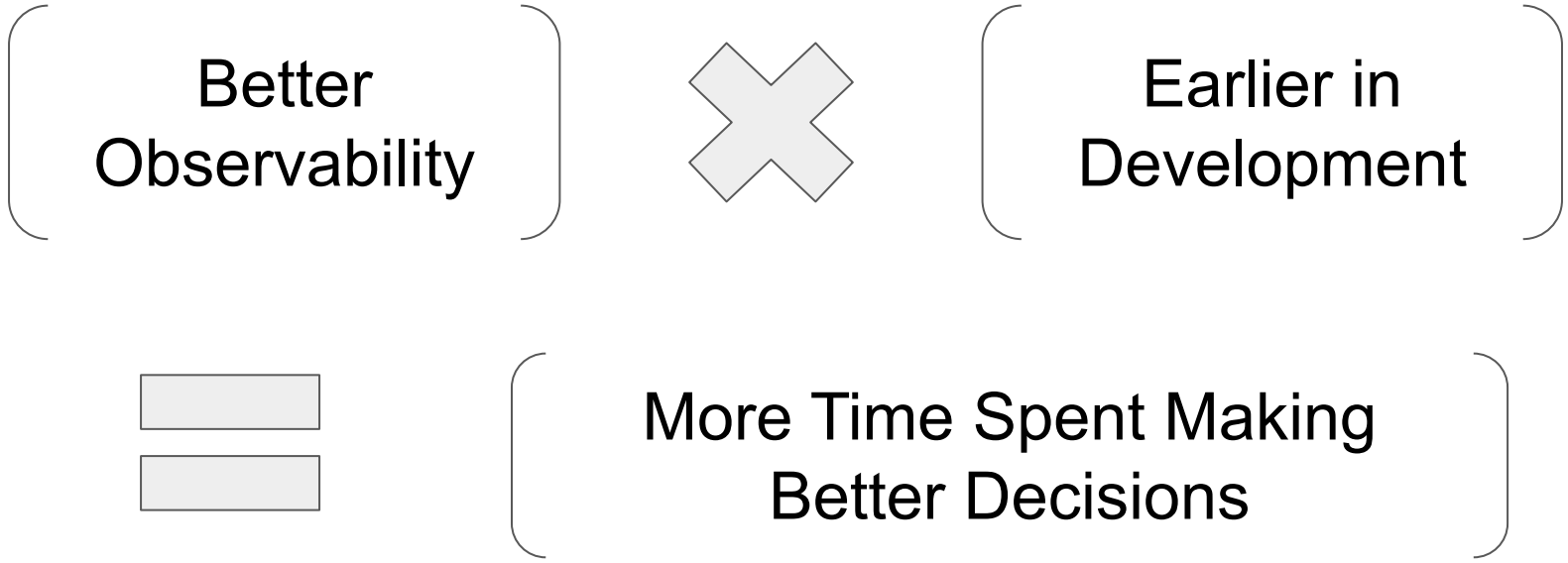
# Observability: A Force Multiplier!

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?




TechAtBloomberg.com



Bloomberg

Engineering

# 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





Bloomberg

Engineering

# 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!





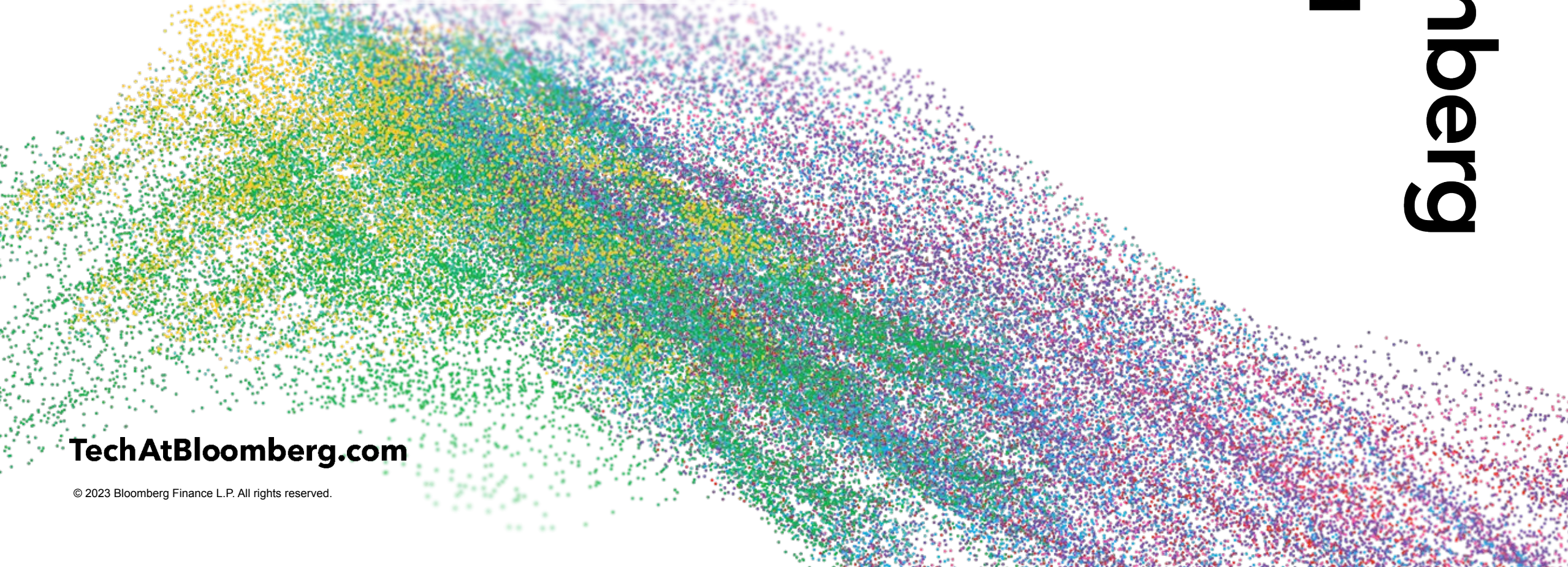


# Thank you!

## We are hiring: bloomberg.com/engineering

Bloomberg

Engineering



TechAtBloomberg.com

# Sources


[1] Huang, L., & Holcombe, M. (2009). Empirical investigation towards the effectiveness of Test First programming. Inf. Softw. Technol., 51, 182-194.

[2] Erdogmus, H., Morisio, M., & Torchiano, M. (2005). On the effectiveness of the test-first approach to programming. IEEE Transactions on Software Engineering, 31, 226-237.

[3] Padberg, F., & Müller, M.M. (2003). About the Return on Investment of Test-driven Development.

[4] Müller, M.M., & Hagner, O. (2002). Experiment about test-first programming. IEE Proc. Softw., 149, 131-136.

[5] Roman, A., & Mnich, M.A. (2020). Test-driven development with mutation testing – an experimental study. Software Quality Journal, 29, 1 - 38.

[6] Santos, A., Vegas, S., Dieste, Ó., Uyaguari, F.U., Tosun, A.N., Fucci, D., Turhan, B., Scanniello, G., Romano, S., Karac, I., Kuhrmann, M., Mandic, V., Ramač, R., Pfahl, D., Engblom, C., Kyykka, J., Rungi, K., Palomeque, C., Spisak, J., Oivo, M., & Juristo, N. (2020). A family of experiments on test-driven development. Empirical Software Engineering, 26.