Chaos Engineering Framework

Amit Sengupta (Independent Research)
Email – amits2913@gmail.com
Independent Researcher

Abstract

Chaos engineering is the process of experimenting with a distributed computer system by introducing unexpected disruptions in order to gauge the system's resiliency and to identify potential points of failure. Chaos engineering is often confused as a way of testing your system. However, chaos engineering is not testing since tests are simply statements about a system's known properties. It makes assertions based on existing knowledge instead of verifying properties of the current system. On the other hand, experimentation is all about exploring the unknown. It creates new knowledge by proposing a hypothesis. If it's proven, then the confidence grows and if it's disproved, then we learn something new. Chaos engineering is all about experimentation. No amount of testing can match the insights gained through experimentation as experimentation can explore scenarios that cannot occur during testing.

Keywords: - *Chaos Engineering, Chaos Experiments, Blameless Postmortem, Chaos Monkey, Simian Army, Enterprise Resiliency, Fault Tolerance, Blast Radius, Controlled Chaos.*

Introduction: -

Chaos Engineering as an engineering discipline highlights experimentation with production systems to build confidence in its capability to respond and sustain in turbulent conditions. It helps to identify faults and gaps in the systems. One of the basic principles of chaos engineering is to introduce hypothesis and experiments. The nature and scale of the hypothesis are small while closer to the live systems in terms of functionalities. The primary objective of chaos engineering is to generate new unknown information about a system and its behavior pattern while reacting to a catastrophe.

Principles of Chaos Engineering: -

Unlike what the name may suggest, chaos engineering follows a systematic approach. The following principles describe an ideal way to run experiments on your system.

1. ***Build a Hypothesis -*** The first principle of chaos engineering is to focus on the measurable output of a system instead of its internal attributes. Measuring that output over a short period will show you the system's steady state. Some metrics that represent a system's steady state include latency percentile, error rates, system throughput, etc. Then you can hypothesize how that output will change when an incident is introduced. By observing systemic patterns while experimenting, chaos engineering verifies how outputs are based on your system health.
2. ***Introduce New Real-world Events -*** Chaos variables reflect real-world events, specifically hardware and software failure, and non-failure events like a traffic spike. Each event is prioritized by its frequency or potential impact. Any event that has the potential to disrupt the system's steady state is a potential variable in chaos experiments.
3. ***Run Experiments in Environments Matching Production*** - Systems, in general, behave differently depending on the traffic patterns and environment. Due to its varying nature, the only way to capture the request path in a reliable manner is by modeling your testing environment accurately by mirroring real traffic conditions and usage.

Chaos experiments are usually run in a simulation of the production environment that is as accurate as possible. This is a protective measure to prevent a worst-case scenario that could occur when testing in production. You also need to have proper control over the system environment in case the experiment goes sideways.

4. ***Automate Experiments* -** Running experiments on a system is usually an interesting job and requires a lot of creativity. However, running the experiments manually will take engineers away from more meaningful work. Once an experiment has been established, with its outputs logged and its range of conditions set, you should automate it to explore a wider range of conditions. To continuously run the experiments, automate the process and let the software take care of the rest.
5. ***Understand Blast Radius*** - Experimenting in production is a bold move and can potentially cause unnecessary pain to the customer. While small incidents can be handled quickly, the chaos engineer must ensure that any fallout is contained and minimized. Additionally, you should also have the incident response team on-call to handle incident management.

How Does Chaos Engineering Work?

In chaos engineering, everything is an experiment, and each experiment starts with a specific fault injected into the system. Later, the admins observe what actually happened and compare it to what they thought would happen. Chaos engineering experiments generally involve two groups of engineers. The first group generally controls the failure injection and the second one deals with the effects. Here's the step-by-step flow of chaos engineering experiments in practice:

1. Define the system's steady state as a measurable output that indicates normal behavior.
2. Hypothesize how the system's output will change in the experimental groups compared to the control group.
3. Introduce variables that reflect real-life events ranging from hardware and software failure to non-failure events like traffic spikes.
4. Work on disproving the hypothesis by comparing the system's steady state in both control and experimental groups.

The confidence in the system's behavior grows if it turns out to be harder to disrupt the system's steady state. Alternatively, if you discover a weakness, then work to improve it before the behavior manifests in the system at large.

Chaos Engineering Maturity Model

As organizations mature and expand, chaos engineering offers more opportunities and techniques to enhance the resiliency and reliability of the enterprise. We listed down the various stages of maturity based on capabilities to provide a clear understand of the overall journey.

Stage Of Maturity	Maturity Definition
Beginner	1. Non-Availability of Non-Functional requirements 2. Stake holder have no to little knowledge about the resiliency parameters 3. Basic awareness about what could go wrong in terms of application or infrastructure reliability 4. Leverage Dev environment for Chaos experiments to perform Basic disruption 5. Adhoc Chaos experiments. 6. Use of Enterprise Chaos engineering tools or Home-grown scripts
Intermediate	1. Non-Functional requirements related to Chaos testing are defined for some of the use cases

	2. Stakeholders are aware about the resiliency parameters 3. Teams involved understand their roles and responsibility to make each component of the application. with dedicated team members working on Chaos Engineering 4. Some of the tests are run in Production environment manually while most of the tests are run in non-production environment 5. Test results have to be manually curated 6. Experiments are integrated with Continuous delivery and business impact is measured 7. Use of Enterprise Tools to perform Chaos engineering experiments
Emerging	1. Well defined Non-Functional requirements related to Chaos testing are defined for most of the use cases 2. Stakeholders are well aware about the resiliency parameters 3. Teams involved understand their roles and responsibility to make each component of the application resilient and work towards it in all project phases 4. Most of the tests are run in Production environment with automated setup and result analysis capabilities 5. Experiments are integrated with Continuous delivery and business impact is measured. 6. Use of Enterprise Tools to perform Chaos engineering experiments
Advanced	1. Well defined Non-Functional requirements 2. Stakeholders participate in Chaos experiments by default as they contribute to enhance the application resiliency 3. Teams involved considers Chaos experimentation as mandatory in onboarding plan 4. Chaos Experiments is carried out in each step of development and in production environment. 5. Chaos test design, execution, and early termination are fully automated and are dynamic in nature 6. Use of Enterprise Tools to perform Chaos engineering experiments

It's important to note that chaos engineering offers many benefits regardless of your organization's maturity model. Starting sooner can give you more time to develop your expertise in the area.

Conclusion: -

The goal of chaos engineering is to improve a system's reliability and resilience, which makes it an essential part of any mature SRE (site reliability engineering) solution. The chaos engineering mindset also helps DevOps teams work with unpredictability. Many SRE practices such as SLOs (service level objectives), retrospectives, and runbooks can integrate with chaos engineering to improve efficiency.

The impact of SLOs on chaos engineering is important to determine the impact of a hypothetical failure, which is not exactly easy. Chaos engineering also helps SRE teams improve their runbooks by giving them more opportunities to evaluate them. It also helps them build a library of incident retrospectives as teams write retrospectives for chaos experiments as they would for a real event enhancing reliability and net promoter index.

Reference:

1. https://linearb.io/dev-interrupted/blog/chaos-engineering-the-practice-behind-controlling-chaos?_bt=&_bk=&_bm=&_bn=x&_bg=&utm_term=&utm_medium=cpc&utm_campaign=PMax&utm_source=google&gad_source=1&gclid=CjwKCAjwuMC2BhA7EiwAmJKRrKMNzOt3q3KfGHfP83CcNt5lwoIE_9htWd9qkJRf_rJ4mdFEzdStUxoCrtkQAvD_BwE
2. http://techblog.netflix.com/2012/07/chaosmonkeyreleasedintowild.html
3. http://queue.acm.org/detail.cfm?id=2371297
4. https://azure.microsoft.com/enus/blog/insideazuresearchchaosengineering/
5. http://www.datacenterknowledge.com/archives/2014/09/15/facebookturnedoffentiredatacent ertotest-resiliency/
6. http://techblog.netflix.com/2014/10/fitfailureinjectiontesting.html
7. http://techblog.netflix.com/2015/02/spspulseofnetflixstreaming.html
8. https://www.usenix.org/system/files/conference/osdi14/osdi14paperyuan.pdf
9. Sam Newman: Building Microservices, O'Reilly Media, Feb. 2015.