Define and prepare the steady state

The steady state is the control state of a system, and can be defined with metrics such as error rate, response time, or resource utilization. The purpose of having a steady state is to validate the stability of the application before any chaos experiments are run.

With Infrastructure as code, this could include checking to make sure that all components are deployed.

Observe your system

Examine the architectural diagram and try to identify key components to test.

Baseline your metrics

Make sure you're monitoring the system as it is and that metrics are being collected to help you define that steady state quantitatively. You should also set SLOs/SLIs per service so that you know when the performance strays away from the steady state.

Build a hypothesis

A hypothesis should be formulated as a positive statement such as "When an instance is shut down, the application recovers and switches to another node within 10 minutes." It should be based on measurable output.

In testing terminology, the hypothesis is what would typically determine whether the test has passed or failed-- although chaos engineers usually speak in terms of experiments rather than tests.

Set abort conditions

Setting an abort condition means determining the situations beyond which it would not be meaningful to continue the experiment, or those beyond which the experiment would be detrimental to the application (especially if you're testing in production).

Set blast radius and magnitude

Plan in advance and try to limit blast radius and magnitude. Start small! For example, run *one* experiment with only a 10% increase in load to services.

Execute chaos experiments to disprove the hypothesis

Introduce variables in the forms of experiments (latency, node shutdown, etc.) to determine how they affect an application. Prioritize these variables in terms of severity of frequency, and only do them one at a time to start with.

Experiments are designed specifically to try to attack the application in different ways to produce an outcome other than the hypothesized one.

> The harder it is to disrupt the steady state, the more confidence we have in the behavior of the system. If a weakness is uncovered, we now have a target for improvement before that behavior manifests in the system at large.

Environment

Chaos experiments are best done on production.

Analyze results

Re-test

Tweak the experiment parameters, explore "what if" scenarios, and think of other experiments that might disprove the hypothesis.

Share your results

Automate

Incorporate some experiments into the CI CD Pipeline and make them run continuously, to minimize manual effort.

References

- In the kitchen - a sprinkle of fire and chaos
- Article/Using Chaos Engineering to Test Distributed Systems
- Presentation/Using Chaos Engineering to Test Distributed Systems
- PRINCIPLES OF CHAOS ENGINEERING - Principles of Chaos Engineering