RESILIENCE ENGINEERING: AN INTRODUCTION

Hanging out under the same umbrella as chaos engineering, resilience engineering is a way of building your systems to fail. Let's take a look.

What is resilience engineering?

Let's start with resilience—the ability to keep on keeping on in the face of failure. In the words of Bob Dylan, "There's no success like failure, and failure is no success at all." And in my own, "Failure sucks." In terms of technology and IT systems,

Resilience is a system's ability to recover from a fault and maintain persistency of service dependability in the face of faults.

Resilience engineering, then, starts from accepting the reality that failures happen, and, through engineering, builds a way for the system to continue despite those failures. Good resilience engineering produces a system **that can adapt**. Resiliency can be built into any system, and it offers a lens to look at critical areas like cybersecurity and operations.

Here are some examples:

- If the system adapts by acquiring servers from a different region when all the servers in its zone fail, it has been successfully engineered.
- If the system adapts by taking the next best CPU when the cloud provider cancels providing the present CPU, the system has been successfully engineered.

- If the system fails to scale its number of servers when, suddenly, its number of users skyrockets, then the system has *not* been successfully engineered.

How adaptability supports resilience

Adaptability is the defining trait of resilience. If you are to provide a SaaS product, and your systems go down, there is no product.

Humans have long been the primary agent in making systems adapt. It has been people who are on-the-ready to investigate and get the software back up and running as quickly as possible—to make a system resilient to failure. Human attention has been required to ensure system resiliency.

But human labor is old-school in the age of software. *How Complex Systems Fail* by Richard Cook is a short document that covers common ways that systems fail. 50% of the choices have to do with human error or the necessity of human intervention.

Software, now, is being designed to help make systems adapt. With the dawn of cloud computing, and infrastructure parts like containers and Kubernetes orchestration, software is doing the work instead of people.

How to build in resilience

There are a few good ways to build resilience into your systems.

React to failures. When errors occur, teams respond to them. When a failure occurs and there is no response, you are not adapting. (Not responding to failures is one characteristic of the organizational death spiral.)

Log correctly. It is easiest to treat failures when their cause is known. Building good logging reports into the application can help identify errors quickly, allowing tech/support staff to easily handle and treat the errors. Good logging is critical to root cause analysis.

Check your metrics. Building resiliency should consider important metrics like mean time to failure (MTTF) and mean time to recovery (MTTR) in order isolate impacted components and restore optimal performance.

Know your options. Backup plans are illustrative of preparedness, not paranoia. When plan A fails, and your company already has plans B, C, and D in place, your ability to respond to the failure increases greatly.

Finding resilience in cloud computing

If a document were to suddenly disappear from a computer in a hard drive crash, that disappearance would be a failure of the system. That failure would strike a user as odd—something that should never occur. Resiliency in systems has become something we all expect.

The cloud infrastructure for storage was developed in the 2000s with products like:

- Microsoft OneDrive
- Dropbox
- Apple iCloud

Users could expect their files to remain in existence in the event of a computer failure. An added benefit to this cloud storage, which has become a key feature: we can now access files from any computer or device.

The infrastructure for the other parts of computing, memory and compute power, was developed in the 2010s. By 2018, these were expected components of software products. Now, the value of cloud computing, with regards to building resiliency, comes when computation-heavy features require greater resources than what the user's or edge device provisions. Examples of these compute-heavy features are:

- Image processing
- Certain machine learning models
- Browser-based microservices

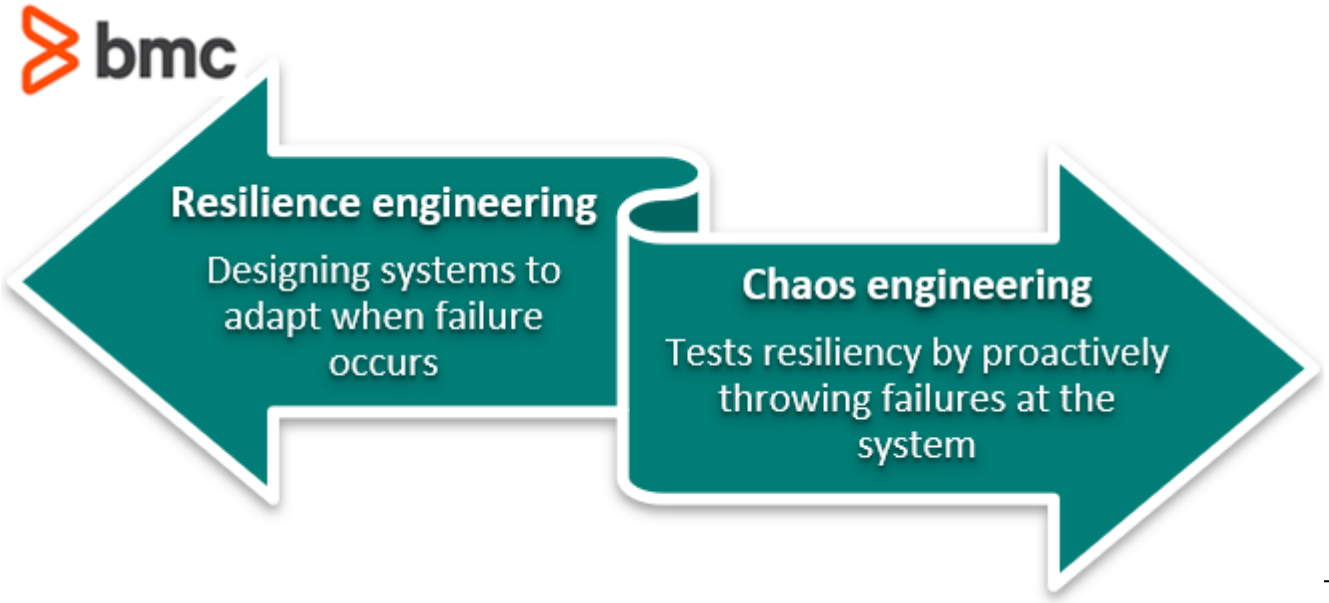
A compute-heavy task could fail in the event the edge device doesn't have appropriate resources to handle the task. The system can avoid failure by using cloud compute to process the task, and simply return the "42" value back to the user via a network connection.

Cloud computing is an easy way to increase the resilience of a software system. When the cloud doesn't solve the resilience problem, then it is building fault awareness and fault tolerance directly into the applications.

Resilience engineering vs chaos engineering

In the practice of resilience engineering, a method known as chaos engineering is one way to test resiliency:

- The goal of resilience engineering is to design systems to adapt in the event of failure.
- Chaos engineering helps test the resiliency of the system by proactively throwing common failures at the system.



The practice of chaos engineering was a practice developed by Netflix. The idea was an experiment in improving system resilience: how can engineers build the system to be more resilient *before* bad things happen, instead of waiting until after the event?

This led the Netflix team to create Chaos Monkey, a popular tool that simulated common failures in

the system's infrastructure. Like its namesake, the tool acts like a monkey rampaging inside a data center, unplugging and cutting cords wherever it goes.

Keep systems working

Resilience engineering is all about adaptability. When someone builds their system to be resilient, it means the system can encounter failures, and find a way to keep on keeping on.

Additional resources

For more on this topic, explore our BMC DevOps Blog or browse these articles:

- What Is a Cyber Resilience Strategy?
- Operational Resiliency in Financial Services
- Reliability vs Availability: What's the Difference?
- Impact of Redundancy on Availability