EPJ Web of Conferences **321**, 02006 (2025) https://doi.org/10.1051/epjconf/202532102006
APITECH-VII-2025

Chaos engineering in cloud platforms

Sheinman Vered*

University of Haifa, Abba Khoushy Ave 199, Haifa, 3498838, Israel

Abstract. This article explores the implementation and impact of chaos engineering (CE) on cloud platforms, emphasizing its role in enhancing system resilience and reliability. Various CE methodologies, such as fault injection and passive observation, are discussed, alongside case studies from major cloud service providers (AWS, Azure, GCP). The integration of artificial intelligence and machine learning in automating and refining chaos experiments is analyzed, highlighting its growing importance as cloud infrastructures become more complex. The potential of CE for proactive prediction and management of system vulnerabilities is underscored, illustrating its capability to preemptively identify and address weaknesses. Through a comprehensive review of existing research and practical case studies, this article aims to demonstrate how CE can effectively bolster cloud platform resilience. The market for CE tools, valued at two billion dollars in 2022, is projected to grow significantly, driven by increasing investment and the need for robust security models. The historical development of CE, from Netflix's Chaos Monkey to contemporary practices, is also covered, showcasing the evolution of CE and its adoption by tech giants to maintain system robustness and reliability.

1 Introduction

Chaos engineering (CE) is a pioneering discipline that focuses on proactively testing and improving the resilience of software systems by deliberately injecting controlled failures into them. Originating from the innovative practices of tech giants like Amazon, Netflix, and Google in the early 2000s, this approach challenges conventional wisdom by prioritizing proactive experimentation over reactive mitigation. This proactive methodology is particularly pivotal in the context of cloud computing, where dynamic, distributed environments are vulnerable to a range of unpredictable failures that can significantly impact service delivery and customer experience.

The goal of this article is to explore the principles of CE application within cloud platforms (CP). Through a comprehensive review and integration of existing research and case studies, this article will explore how CE can effectively bolster the resilience of cloud platforms against potential failures.

In cloud infrastructure (CI) issues are often caused by misconfigured resources. In the 2020 report by Information Technology Intelligence Consulting (ITIC), it was revealed that 98% of organizations report that an hour of downtime costs them over 150 thousand dollars

* Corresponding author: bringomun@rambler.ru

[1]. Furthermore, 40% of enterprises experience hourly downtime costs ranging from one million dollars to five million dollars. To address these issues, there is a pressing need for new security models.

The market for CE tools was valued at two billion dollars in 2022, with expectations to grow at a compound annual growth rate (CAGR) of over 8,5% from 2023 to 2032 (Figure 1).

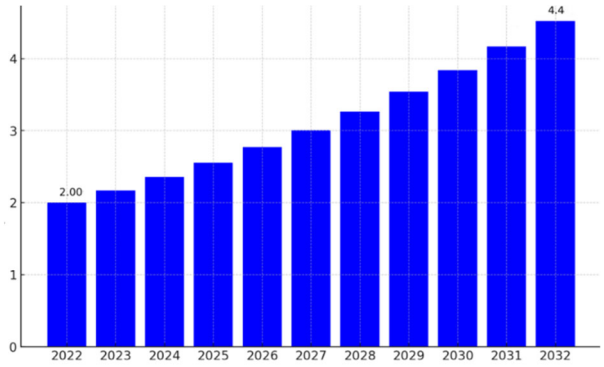

Fig. 1. CE market size, billion dollars [2]

This growth is driven by increased investment in the sector through financing innovative startups. For example, in September 2022, the CE startup Steadybit raised 7.8 million dollars. CE tools are most often used in the IT & telecom sector: 24% of the total market share.

CE started in 2010 when Netflix developed the Chaos Monkey tool to enhance system resilience during their transition to CI. The company deliberately disrupted Amazon Web Services (AWS) to identify potential risks. Since the concept proved effective, Netflix expanded it with the Simian Army, introducing more complex failure modes. This suite included tools to disable entire AWS regions, simulate network outages, and expose security vulnerabilities [3]. In 2012, Netflix open-sourced Chaos Monkey, facilitating widespread adoption of the method across various industries. Organizations could then stress-test their infrastructures and proactively enhance their resilience. Since then, the practice has evolved, with companies like Gremlin launching managed corporate solutions in 2016, further institutionalizing CE across different sectors. Over the years, CE has been employed by technology giants including Amazon, Microsoft, and Google, underscoring its importance in maintaining system resilience and reliability.

2 Chaos engineering principles

CE is predicated on the idea that the best way to ensure high availability and robustness in systems is by deliberately introducing disturbances to study how these systems fail and how they recover. Originating from practices developed by companies operating at scale, such as Netflix, CE has evolved into a disciplined approach to improving system resilience. This methodology involves a series of principles that guide the proactive disruption and subsequent recovery, making it possible to anticipate and mitigate potential issues before they affect the end-users adversely.

Principle of controlled experimentation: one of the foundational principles of CE is controlled experimentation. This involves setting clear boundaries for each test to isolate variables and limit the impact. The goal is to simulate realistic scenarios under controlled conditions, which helps in pinpointing vulnerabilities without risking the entire system's

stability [4]. This principle ensures that the experiments are reproducible, and their effects are predictable, thereby providing reliable data for improving system resilience.

Principle of hypothesis-driven testing: central to CE is the formulation of hypotheses based on system behaviors. Before any experiment is conducted, a hypothesis that predicts the outcome of the test is established. This prediction might pertain to the system's endurance under a specific fault condition or its recovery path following a disruption. This structured approach helps in aligning the chaos experiments with specific reliability goals, making the outcomes actionable and directly tied to enhancing system performance.

Principle of real-world conditions: the experiments are designed to mimic conditions that could realistically occur in production. This includes scenarios like server failures, spikes in traffic, network latencies, or database and service degradation. By testing the system's response to these conditions, engineers can assess and enhance the fault tolerance mechanisms that are critical during actual operational challenges [5]. The principle ensures that the findings of chaos experiments are applicable and beneficial in real-world settings.

Principle of automation given the complexity and potential frequency of tests, automation is a key principle of CE. Automating the setup, deployment, and analysis of chaos experiments enables continuous integration and continuous development (CI/CD) pipelines to include resilience testing [6]. Automation ensures that the chaos tests are conducted systematically and without bias, allowing for consistent testing over time, which is essential for iterative improvement.

Principle of continuous learning: the iterative nature of CE means that each test provides data that refine understanding and improve system design. Continuous learning from each experiment helps in building a culture of resilience where insights gained are used to fortify the system incrementally. This principle encourages ongoing adjustments and updates to the system architecture, which are crucial for maintaining reliability in an evolving technological landscape.

A CE experiment typically progresses through eight distinct phases, each critical to the comprehensive evaluation and understanding of system resilience [7]:

- Experiment design: define the scope, objectives, and methods of the chaos experiment, including the selection of target systems and fault types to inject.
- Preparation: set up the necessary tools and environment, ensuring all systems are ready and stakeholders are informed about the planned disruptions.
- Baseline measurement: establish a performance and behaviour baseline to compare against the impact of the chaos interventions.
- Fault injection: introduce faults or failures into the system to simulate real-world disruptions and observe how the system reacts.
- Observation: monitor the system’s response to the injected faults in real-time, using predefined metrics and logging systems.
- Analysis: evaluate the data collected during the experiment to understand the impact of the faults and identify weaknesses or failure points in the system.
- Conclusion: draw conclusions from the analysis to determine whether the system’s resilience goals were met and if the existing defenses were effective.
- Documentation and communication: document the process, findings, and conclusions; communicate the results and recommendations to relevant stakeholders to guide future improvements.

CE represents a proactive approach to identification of vulnerabilities in infrastructure platforms. It is not merely about causing disruptions but about learning from them to create more resilient systems. By adhering to these principles, organizations can better prepare for the unexpected, ensuring that their systems can not only withstand crises but also recover from them swiftly and efficiently.

3 Chaos engineering techniques

CE employs various techniques to enhance system resilience. These include fault injection, passive observation and metrics-driven analysis.

Fault injection involves introducing errors into the system to assess its robustness, potentially by stopping processes, simulating server failures, or creating component delays. Passive observation incorporates monitoring the availability and behavior of the service and measuring service KPIs (key performance indicators) before, during and after the chaos experiment. Metrics-driven analysis evaluates system behavior and performance. Organizations define the RTO (Recovery Time Objective) and RPO (Recovery Point Objective) for their systems. Those metrics are used to assess the impact of failures on these objectives thus providing insights into system behavior under stress, which allows organizations to make data-driven decisions to improve resilience. This approach allows both proactive and reactive resilience enhancements.

Table 1 shows a comparison of these techniques.

Table 1. Chaos engineering techniques.

Technique	Description	Advantages	Disadvantages
Fault injection	Introducing faults to test how systems endure under stress	Directly identifies system vulnerabilities	Can cause system disruption if not carefully managed
Passive observation	Before, during and after the experiment the behavior of the system is observed	Real-world insights into system performance, reliability, and failure modes	May result in a delayed response to incidents or failures
Metrics driven analysis	Assess system behavior, performance, and resilience.	Provides valuable data for root cause analysis and troubleshooting when incidents occur	The selection of metrics used for analysis may introduce bias or overlook critical aspects of system behavior or performance

The use of these techniques enables organizations to proactively address potential issues by continuously refining these methods, particularly with advancements in AI, aims to enhance their effectiveness and applicability in increasingly complex IT environments.

4 CE implemenation at cloud service models

Cloud architectures commonly implement and integrate IaaS (Infrastructure as a Service), PaaS (Platform as a Service), and SaaS (Software as a Service) models, each providing different levels of abstraction and management [8]. IaaS offers base-level infrastructure with flexibility but requires significant management of operating systems and applications. PaaS provides a further abstraction that includes managed operating platforms, while SaaS delivers complete application services managed by the provider, minimizing customer involvement in underlying operations.

CE plays a crucial role in these environments by controlled faults introducing faults to test and improve the resilience of CP. For example, in an IaaS setup, engineers might use CE

to simulate hardware failures or network disruptions to test the robustness of virtual machines and storage services.

Here's a short Python script that uses the «boto3» library to simulate a network disruption on an AWS EC2 instance (Figure 2). The script represents a common form of chaos experiment in an IaaS setup:

```
import boto3

# Initialize a boto3 client
ec2 = boto3.client('ec2', region_name='us-west-2')

# Function to simulate a network disruption by
detaching a network interface
def simulate_network_disruption(instance_id):
    # Get the network interface attached to the instance
    Interfaces = ec2.describe_network_interfaces(Filters
    =[{'Name': 'attachment.instance-id', 'Values': [
    instance_id]}])
    network_interface_id = interfaces[
    'NetworkInterfaces'][0]['NetworkInterfaceId']

    # Detach the network interface
    ec2.detach_network_interface(AttachmentId=
    network_interface_id)

# Replace 'your_instance_id' with your actual instance
ID
simulate_network_disruption('your_instance_id')
```

Fig. 2. Simulating Network Disruption on AWS EC2 Instance: A Chaos Experiment.

This script demonstrates a way to detach a network interface from an EC2 instance, which can simulate a network failure, affecting the instance's ability to communicate.

In PaaS and SaaS models, CE could focus more on application-level disruptions such as forced crashes or throttled databases to ensure that applications can gracefully handle failures. In such cases, a Python script based on the request's library can be used (Figure 3):

```
import requests
import time

# Function to simulate database throttling
def throttle_database(api_url):
    print("Starting API request...")
    response = requests.get(api_url)
    print("Simulating database load...")
    time.sleep(5)  # Introduce a 5-second delay to
    simulate throttling
    print("Response:", response.status_code)

# Replace 'your_api_url' with your actual API URL
throttle_database('your_api_url')
```

Fig. 3. Application-Level Chaos Engineering in PaaS/SaaS: Handling Failures with Python.

This script makes a GET request to a specified API and introduces a deliberate delay before processing the response, which can mimic the effect of a database under heavy load.

Implementing CE in cloud architectures involves techniques like fault injection, stress tests, performance degradation:

- Fault injection might include, for example, shutting down virtual servers or blocking network traffic, to assess the resilience of load balancers and auto-scaling systems.
- Stress tests could simulate sudden spikes in user demand to evaluate the performance of web servers and database systems under pressure.

- Performance degradation, such as increased latency or reduced throughput, targets to meter the system's responsiveness and resilience to degraded performance.

These methodologies help uncover potential points of failure that could impact service continuity and user experience.

The integration of CE into CP development cycles encourages a proactive approach to disaster recovery planning and operational resilience. This not only helps in adhering to SLAs but also enhances customer trust by providing highly reliable and robust services. By continuously testing and improving the architecture's ability to handle unexpected disruptions, organizations can ensure that their CP remain reliable and efficient, even under adverse conditions.

5 Chaos engineering in practice

Chaos experiments in CP are designed to methodically assess the resilience and robustness of systems by simulating realistic disruptions. These tests include simulating server failures to test redundancy protocols and failover mechanisms, introducing network latency to evaluate the performance under suboptimal internet conditions, high traffic load simulation by increasing the volume of incoming requests to assess the system's scalability and performance under heavy load conditions [9].

Table 2 presents test scenarios that assist in identifying potential vulnerabilities of CP.

Table 2. Chaos test scenarios in CP: system resilience assessment.

Fault type	Test scenario	Purpose	Impact Assessment
Server failure	Intentional shutdown of virtual machines	To assess failover and redundancy effectiveness	Determine downtime minimization capabilities
Network latency	Throttling network speed	To evaluate response times under delay	Test application performance during slow connectivity
High traffic load	Stress testing	To uncover memory leaks or service degradation	Assess robustness of the system over a time

These chaos tests are crucial not just for proving the system's resilience but also for planning future improvements. Chaos experiments can reveal opportunities to automate recovery and mitigation actions. Companies can leverage this knowledge to implement self-healing mechanisms and redundancy architecture to minimize the impact of failures [10].

6 CE native tools by major cloud providers

CE has been employed across various industries to enhance system resilience, with some notable case studies from major cloud service providers.

AWS has developed its own CE tool called AWS Fault Injection Simulator, which integrates closely with Amazon CloudWatch, monitoring and observability service. which allows users to conduct controlled chaos experiments, monitor the impact of these experiments in real-time, automate experiment workflows, thus following best practices for enhancing system reliability and resilience on AWS [11].

Microsoft Azure has its own dedicated CE service called Azure Chaos Studio, that enables users to conduct controlled experiments to identify vulnerabilities across Azure resources, supports experiments automation and scheduling. This tool integrates closely with

Azure Monitor, Microsoft's monitoring and observability service, which allows users to monitor the impact of chaos experiments in real-time [12].

Google's approach to CE implementation in GCP revolves around leveraging third-party tools, emphasizing monitoring and observability, embracing reliability engineering principles, and fostering a collaborative culture of experimentation and learning.

Alibaba Cloud employs a rigorous approach to CE through their proprietary CE tool, ChaosBlade, which allows to conduct meticulously designed chaos experiments to simulate real-world failure scenarios, systematically assessing the resilience and reliability of their cloud services [13]. Complementing these efforts, AliCloud integrates sophisticated monitoring and observability services, including CloudMonitor and Application Real-Time Monitoring Service (ARMS), to provide real-time insights into the impact of chaos experiments and facilitate continuous improvement. This disciplined approach reflects Alibaba's commitment to scientific rigor and technological excellence.

Table 3 provides a comparative analysis of the CE tools used in Microsoft Azure, Amazon Web Services (AWS) and Google Cloud Platform (GCP).

Table 3. Comparative analysis of CE tools in AWS, Azure, GCP.

Features/Tools	**AWS**	**Azure**	**GCP**
Experiment Design tool	AWS FIS (Fault Injection Simulator)	Azure Chaos Studio	No GCP dedicated tool, while offering usage of third-party tools, such as Chaos Mesh and LitmusChaos
Creation and scheduling of experiments	AWS Management Console, AWS CLI, or SDKs	Azure portal or Azure CLI	Chaos Mesh and LitmusChaos APIs and CLI tools
Fault Injection	Broad AWS-specific faults, such as EC2 instance termination, S3 bucket deletion, and RDS instance failure and more	Diverse faults across Azure, such as VM restart, network latency injection, DNS outage, service disruption, and more.	Faults on Compute Engine instances, Kubernetes Engine clusters, Cloud Storage buckets, and more, through third-party tools.
Observability and Monitoring	Integrated CloudWatch	Uses Azure Monitor	Offers native GCP's monitoring and logging service

The analysis table above reveals differences and similarities in functionality and features. Each cloud provider offers its suite of tools tailored to their platform's requirements and ecosystem. AWS provides Fault Injection Simulator (FIS) for chaos experiments, Azure offers Chaos Studio, and GCP does not have a native CE tool but supports third-party solutions. While AWS FIS integrates tightly with CloudWatch, Azure Chaos Studio integrates with Azure Monitor. GCP users may rely on third-party tools like Chaos Mesh. Despite differences, all three cloud providers prioritize resilience testing, demonstrating a commitment to improving system reliability in cloud environments.

7 Challenges and considerations

The implementation of CE presents numerous technical and organizational challenges that can impede its integration into CP. Understanding these challenges is essential for effectively leveraging CE to enhance system resilience and reliability.

7.1 Technical challenges

A primary hurdle in deploying CE is the complexity of these architectures. They often feature multiple layers and dependencies that complicate the simulation of failures and prediction of their effects. Achieving precise control to avoid cascading failures requires in-depth understanding and strategic planning.

Integrating CE practices into existing infrastructures, particularly those not originally designed to accommodate such testing, presents considerable difficulties. These challenges necessitate robust compatibility strategies and might require extensive modifications to both infrastructure and application design.

Another significant trend is the development of more granular and sophisticated testing tools that can simulate highly specific scenarios across diverse CE. These tools aim to create a more comprehensive understanding of potential failures at both the application and infrastructure levels. As CP continue to expand, the ability to conduct precise, targeted chaos experiments becomes crucial for maintaining service integrity and customer trust.

An example of a sophisticated open-source Chaos Engineering tool is Chaos Mesh, developed by the Cloud Native Computing Foundation (CNCF) and specifically designed for Kubernetes environments. Its capabilities include a diverse set of chaos experiments, such as network, filesystem, and time-based chaos, allowing users to simulate various failure scenarios at scale and assess system resilience [14]. For example, a complex chaos experiment in Chaos Mesh could involve injecting network latency and packet loss to multiple pods across different Kubernetes nodes simultaneously, while concurrently introducing disk I/O errors to specific containers, providing a comprehensive assessment of the system's fault tolerance and recovery mechanisms.

Advanced monitoring and alerting systems represent another significant challenge. Effective chaos experiments demand sophisticated tools to track impacts and the propagation of induced failures in real-time.

Several tools provide alerting systems for service degradation. Some popular open-source tools include:

- Prometheus, a monitoring and alerting toolkit, that collects metrics from monitored targets, stores them in a time-series database, and provides a querying language for analysis. It supports alerting based on predefined rules and integrates with email, Slack, PagerDuty, and more.
- Grafana, a visualization and analytics platform that integrates with Prometheus and other data sources to create dashboards for monitoring and alerting. Grafana allows users to define alert rules based on metrics thresholds and send alerts via email, webhook, or other channels.

Establishing these systems typically involves enhancing existing logging and monitoring solutions, which can be resource-intensive and complex.

7.2 Organizational challenges

Cultural resistance to CE practices, which involve intentional fault injection, is often rooted in concerns over inducing critical failures that could affect service delivery and customer

trust. Transforming this mindset requires educating stakeholders about CE's preventive benefits, highlighting that controlled disruptions can avert more severe unplanned outages.

Securing resources for CE initiatives is a major organizational hurdle. Effective implementation requires personnel with specialized skills and significant investments in technological tools and training programs. Obtaining executive support usually depends on demonstrating potential ROI through comprehensive risk assessments and benefit analyses, challenging without concrete pilot results.

The global expansion of CP is influencing the adoption of CE across different regions. As companies deploy services in multiple geographic locations, the need to ensure these services can withstand regional outages or disruptions is critical. This geographical diversification demands that CE tools adapt to a variety of regulatory and operational environments, making flexibility and customization key features of future CE solutions.

Establishing strict governance for chaos experiments is crucial. Rigorous protocols must outline detailed plans for the initiation, execution, monitoring, and review of tests. Such measures help control risks and ensure that experiments remain within their designated boundaries without impacting critical operations.

Adopting CE in CP, though beneficial, is accompanied by substantial challenges related to technical complexities, cultural barriers, and resource allocation. Overcoming these obstacles requires careful planning, continuous stakeholder education, and clear governance frameworks. Addressing these areas effectively allows organizations to leverage CE to significantly improve the resilience and reliability of their CP.

8 Conclusion

CE has emerged as a vital practice in enhancing the resilience and reliability of CP. By intentionally introducing controlled disturbances, CE allows companies to proactively identify and mitigate potential failures before they impact actual system failure. This approach is especially crucial in today's digital ecosystem, where CI support a wide array of critical applications across various sectors. The adoption of CE in CP demonstrated by industry leaders like AWS, Microsoft Azure and GCP, underscores its significance. These companies have shown that through systematic and controlled disruption, it is possible to significantly improve system robustness, ensuring high availability and the continuous performance of services.

CE in CP is rapidly evolving, driven by the growing complexity of cloud architectures and the critical need for resilient systems. As businesses increasingly rely on cloud services, the demand for robust methodologies to ensure system reliability is greater than ever.

One emerging trend in CE is the integration of AI and ML techniques to automate and refine chaos experiments. AI and ML can predict potential points of failure more accurately than traditional methods, allowing for proactive rather than reactive management of system vulnerabilities. By leveraging AI, companies can automate the creation, execution, and analysis of simulated disruptions, optimizing the learning from chaos experiments.

References

1. ITIC 2020 Global Server Hardware, Server OS Reliability (2020). https://www.ibm.com/downloads/cas/DV0XZV6R?utm_source=xp&utm_medium=blog&utm_campaign=content
2. Global Market Insights 2023 Chaos Engineering Tools Market Size - By Application (Fault injection & testing, Resilience Testing & Disaster Recovery, Security Resilience Testing, Performance & Scalability Testing), Deployment Model, Component, Industry

Vertical & Forecast, 2023-2032 (2023). https://www.gminsights.com/industry-analysis/chaos-engineering-tools-market

3. M.A. Chang, B. Tschaen, T. Benson, L. Vanbever, SIGCOMM Computer Communication Review **45**, 371-372 (2015).
4. A. Basiri, L. Hochstein, N. Jones, H. Tucker, *Automating chaos experiments in production*, in 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), pp. 31-40 (2019).
5. K. Kaliuta, Computer-Integrated Technologies: Education, Science, Production **24**, 48-53 (2023).
6. M. Arsecularatne, R. Wickramarachchi, *Adoptability of Chaos Engineering with DevOps to Stimulate the Software Delivery Performance*, in 2023 International Research Conference on Smart Computing and Systems Engineering (SCSE) (2023).
7. I. Kudrenko, International Journal of Information Systems and Supply Chain Management **17**, 1-26 (2024).
8. A. Rostami, Cloud Service Models-IaaS, PaaS, and SaaS (University of Alberta, 2021).
9. S. Belyaeva, Y. Yanovich, XVIII International Symposium Problems of Redundancy in Information and Control Systems, pp. 100-105 (2023).
10. J. Hernández-Serrato, A. Velasco, Y. Nifio, M. Linares-Vásquez, *Applying Machine Learning with Chaos Engineering,* in 2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), (2020). https://doi.org/10.1109/ISSREW51248.2020.00057
11. A.A. Ahmad, P. Andras, Journal of Cloud Computing **11**, 1 (2022).
12. M. Russinovich, Advancing Microsoft Azure resilience with Chaos Studio (2024). https://azure.microsoft.com/en-us/blog/advancing-microsoft-azure-resilience-with-chaos-studio/
13. Z. Yang, X. Changjun, 2019 Available at: https://www.alibabacloud.com/blog/chaosblade---an-open-source-chaos-engineering-tool-by-alibaba_594850
14. C. Camacho, P.C. Cañizares, L. Llana, A. Núñez, Software: Practice and Experience **52**, 1581-1614 (2022).