

Chaos Engineering

Die Verfügbarkeit von IT-Systemen ist heute wichtiger denn je, weshalb Administratoren gefordert sind, die Stabilität ihrer Systeme bewerten zu können. Das Hauptkonzept des Chaos Engineerings besteht daher darin, ein System absichtlich zu zerstören, um darüber Informationen zu sammeln, die zur Verbesserung der Widerstandsfähigkeit des Systems beitragen.

Es ist ein Ansatz für Softwaretests und für die Qualitätssicherung. Chaos Engineering eignet sich besonders gut für moderne verteilte Systeme und Prozesse. Denn bei solchen Systemen haben die Komponenten oft komplexe und unvorhersehbare Abhängigkeiten. Es ist oft schwierig, Fehler zu beheben oder vorherzusagen, wann ein Fehler auftreten wird.

Chaos Engineering nutzt kontrollierte Experimente, um Schwachstellen zu identifizieren, die zu direkten Systemausfällen oder zu durch Hacker initiierten Downtime führen könnten. Es wird zufälliges und unvorhersehbares Verhalten erzeugt, um die Reaktion des Systems auf diese extremen Bedingungen zu analysieren. Typische Probleme, die sich so finden lassen, sind z. B. Blind Spots, also Stellen, die von der Monitoring Software nicht ausreichend erfasst werden, versteckte Bugs oder Performance-Engpässe.

In diesem Chaos Engineering Training lernen Sie, Kubernetes-basierte Systeme mittels Chaos Engineering möglichst ausfallsicher zu betreiben.

Kursinhalt

- Grundlagen, Konzept und Ziele des Chaos Engineerings für Kubernetes-basierte Lösungen
- Typische Tools für Chaos Engineering
- Arbeiten mit dem LitmusChaos Toolkit
- Steady State Hypothesis (SSH) im Detail
- Abbruch von Microservices und Diensten sowie Injektion anderer Fehler mittels Chaos-Experimenten
- Typische Auswirkungen der Ausführung von Zufallsexperimenten auf die Anwendung
- Umgang mit der Fehlersituation
- Zusammenfassung und Abschlussdiskussion

E-Book Das ausführliche deutschsprachige digitale Unterlagenpaket, bestehend aus PDF und E-Book, ist im Kurspreis enthalten.

Zielgruppe

Dieser Chaos Engineering Kurs richtet sich an Personen, die mit Microservices-Lösungen auf Kubernetes-Plattformen arbeiten und diese mittels Chaos Engineering möglichst ausfallsicher gestalten möchten.

Voraussetzungen

Sie sollten einen technischen Background haben und mit Kubernetes- und Microservices-Lösungen arbeiten.

Dieser Kurs im Web



Alle tagesaktuellen Informationen und Möglichkeiten zur Bestellung finden Sie unter dem folgenden Link: www.experteach.ch/go/CHEN

Vormerkung

Sie können auf unserer Website einen Platz kostenlos und unverbindlich für 7 Tage reservieren. Dies geht auch telefonisch unter 06074 4868-0.

Garantierte Kurstermine

Für Ihre Planungssicherheit bieten wir stets eine große Auswahl garantierter Kurstermine an.

Ihr Kurs maßgeschneidert

Diesen Kurs können wir für Ihr Projekt exakt an Ihre Anforderungen anpassen.

Training		Preise zzgl. MwSt.
Termine in Deutschland		2 Tage CHF 1.975,-
Online Training		2 Tage CHF 1.975,-
Termin/Kursort		Kurssprache Deutsch
09.10.-10.10.25	München	27.04.-28.04.26
09.10.-10.10.25	Online	27.04.-28.04.26

Stand 30.07.2025



Inhaltsverzeichnis

Chaos Engineering

- 1 Einstieg in das Chaos Engineering**
 - 1.1 Grundidee des Chaos Engineering
 - 1.2 Ziele des Chaos Engineering
 - 1.3 Warum braucht man Chaos Engineering?
 - 1.3.1 Monolithische Software
 - 1.3.2 Microservices
 - 1.4 Prinzipien des Chaos Engineering
 - 1.4.1 Steady State Hypothese (SSH)
 - 1.5 Realistische Bedingungen und Wirkungen
 - 1.5.1 Kontinuierliches Chaos
 - 1.5.2 Vergleich zu herkömmlichen Testmethoden
 - 1.6 Übersicht typischer Tools
 - 1.6.1 Chaos Toolkit
 - 1.6.2 Chaos Mesh
 - 1.6.3 Litmus
 - 1.6.4 Kommerzielle Tools
 - 1.6.5 Und die Entscheidung?
- 2 Planung und Durchführung von Experimenten**
 - 2.1 Voraussetzungen
 - 2.2 Entwicklung von Steady State Hypothesen
 - 2.2.1 Key Performance Indicator
 - 2.3 Mental Model
 - 2.4 Entwicklung von Experimenten
 - 2.4.1 Explosionsradius
 - 2.4.2 Umgebung und Targeting
 - 2.4.3 Redundante Experimente
 - 2.4.4 Kontrolliertes Chaos
 - 2.4.5 Unterschiedliche Mental Models
 - 2.4.6 Die Richtigen Fragen Stellen
 - 2.4.7 Kultur der Resilienz
 - 2.4.8 Beispielszenarien
 - 2.5 Durchführung von Experimenten
 - 2.5.1 Debriefing
 - 2.6 Beispiel: Amazon
 - 2.6.1 Emergency Levers und Metriken
 - 2.6.2 Chaos Grundsätze
 - 2.6.3 Standard-Experimente
 - 2.6.4 SLO Report
 - 2.7 Correction of Error
 - 2.8 Die 5 Warums
- 3 Praktische Voraussetzungen für Experimente**
 - 3.1 Überblick
 - 3.2 Container
 - 3.2.1 Images und Registry
 - 3.2.2 Simpler Microservice
 - 3.2.3 Dockerfile
 - 3.3 Kubernetes
 - 3.3.1 Deployment
 - 3.3.2 Services
 - 3.4 Prometheus
 - 3.4.1 Metrik Typen
 - 3.4.2 Installation über Kube-Prometheus-Stack
 - 3.4.3 Services und GUI
 - 3.4.4 Anwendungsspezifische Metriken
 - 3.4.5 Rules & Alertmanager
- 4 Chaos mit Litmus**
 - 4.1 Architektur
 - 4.2 Installation
 - 4.2.1 Installationsverifikation
 - 4.3 Chaos Environment & Infrastruktur
 - 4.3.1 Chaos Infrastruktur
 - 4.4 Simple Fehlerinjektion
 - 4.4.1 Chaos Engine
 - 4.4.2 Chaos Pods
 - 4.5 Chaos Experiment
 - 4.5.1 Chaos Experiment - Infrastruktur
 - 4.5.2 Chaos Experiment - Faults
 - 4.5.3 Chaos Experiment - Fault Target
 - 4.5.4 Chaos Experiment - Explosionsradius
 - 4.5.5 Chaos Experiment - Fault Details
 - 4.5.6 Chaos Experiment - Fault Probes
 - 4.5.7 Chaos Experiment - Visual und YAML
 - 4.5.8 Chaos Experiment - Run
 - 4.6 Resilience Score
 - 4.7 Resilience Probes
 - 4.8 Argo Workflow
 - 4.8.1 Argo Workflow Ressourcen
 - 4.8.2 Argo Workflow - Probe Reference

