# Killer-Bots – KI Monkeys für Widerstandsfähige Systeme

„*Der Betrieb muss aufrecht erhalten bleiben, auch wenn Teile nicht mehr richtig funktionieren. Wenn beispielsweise der Paymentservice eines Online Shops nicht mehr funktioniert, muss der Kunde dennoch in der Lage sein, sich einzuloggen, Produkte zu suchen, sie in seinen Warenkorb zu legen und nachher zu Bestellen.*“

*Jeder CTO / CIO*

## Resilienz – Eine sichere Fahrt auch im Sturm

Diese Forderung an Softwaresysteme wird als **Resilienz,** als Widerstandsfähigkeit und Fehlertoleranz gegen Störungen oder Teilausfälle bezeichnet.

Beim Betrieb von Softwaresystemen, wie zum Beispiel einem großen Onlineportal, geht man davon aus, dass es keine fehlerfreie Software und keinen störungsfreien Betrieb gibt. Jede Software hat Codepassagen in sich, die in bestimmten Konstellationen unerwünscht reagiert. Zudem kann kein Betrieb von Serversystemen einen absolut unterbrechungsfreien Betrieb aller Komponenten ohne Fluktuationen oder Ausfällen garantieren.

Um diesen widerstandsfähigen Betrieb sicherzustellen, gibt es eine Reihe von Maßnahmen und Werkzeugen. Es gibt zum Beispiel Softwarehüllen, mit denen man Softwarekomponenten (Microservices) ummantelt, um sie vor Ausfällen anderer Komponenten zu schützen. Es gibt fachliche Kompensationen für nicht funktionierende Softwarekomponenten. Man kann zum Beispiel eine Bestellung auch ohne Bezahlung auslösen und den Bezahlvorgang nachlagern, sobald der Paymentservice wieder verfügbar ist.

Dennoch weiß man nie, was für Probleme im System noch schlummern nur um auf eine unglückliche Kombination von Zufällen zu wartet, um Chaos zu stiften. Um möglichst viele dieser Situationen geregelt und sicher an die Oberfläche zu befördern gibt es das von Netflix initiierte Vorgehensmodell des Chaos Engineerings.

## Chaos Engineering – Die Ausnahme zur Regel machen

Das von Netflix entwickelte Vorgehensmodell und die dazugehörigen Tools dienen dazu, in gesicherter Art und Weise ein System derart zu stören, dass ein potentielles Chaos zu Tage tritt, ohne in einer Katastrophe zu enden. Im Kern geht es darum, einzelne Teile des Systems zufällig auszuwählen und hart zu beenden. Je nachdem, was beendet werden soll, gibt es sogenannte Monkeys. Der einfache Chaos Monkey wählt zum Beispiel Softwarekomponenten aus (Instanzen von Microservices) und beendet sie. Chaos Gorillas fahren hingegen ein komplettes Rechenzentrum herunter. Eine komplette Liste aller Monkeys findet man in folgendem Wikipedia Artikel: <https://en.wikipedia.org/wiki/Chaos_engineering>. Natürlich wird hierbei der Impact Radius so geregelt, dass der laufende Betrieb weiterhin sichergestellt wird.

## Killer-Bot – der KI Monkey unter den Chaos Monkeys

Das Problem mit den Chaos Monkeys ist, dass sie ihre Ziele zufällig und isoliert auswählen. Man weiß also immer noch nicht, ob eine geschickt gesetzte Kombination von Ausfällen oder Schwächungen sich nicht zu einer Katastrophe aufschaukeln können. Was geschieht zum Beispiel, wenn der Order Service eines Online Shops mehr CPU und Speicher bekommt, damit er noch schneller arbeiten kann, der Paymentservice aber gleichzeitig ausgebremst wird? Läuft der Order Service voll? Sprengt er Batch – Queues? Solche „Chaos Vektoren“ findet man nicht zufällig. Die muss man suchen. Hier kommt der Killer-Bot ins Spiel. Er wählt seine Ziele nicht zufällig aus, sondern sucht und bohrt sich genau in die Schwachstellen des Systems. Damit er diese Schwachstellen und unguten Konstellationen findet, hat er eine KI in sich, die kontinuierlich dazulernt. Hierbei wird die Theorie rund um die Bellman Gleichung und das Reinforcement Learning genutzt, also die gleiche Mathematik, die AlphaGo nutzt um die weltbesten Go Spieler zu schlagen (<https://de.wikipedia.org/wiki/AlphaGo>).

Für uns stehen die unguten Kombinationen von „Kleinigkeiten“ im Fokus: „Hier ein wenig mehr CPU dort ein bisschen schlechtere Netzwerkverbindung“. Die Aktionen, die der Killer-Bot ausführt, sind also keine Totalausfälle einzelner Komponenten, sondern nur das Schwächen und Stärken von Ressourcen innerhalb normaler Parameter. Dazu halten wir das System unter kontinuierlicher Last, indem wir die üblichen Geschäftsvorfälle (Use Cases) automatisch und kontinuierlich ausführen. Der Killer-Bot verändert einzelne Ressourcen und misst die Reaktionen. Dadurch lernt er die Schwächen des Systems kennen und findet so die möglichen Wege in problematische Situationen. Dabei erzeugt er im Hintergrund ein Storybook, das dazu genutzt werden kann, dass die Schwachstellen später geschlossen werden können.

Wenn man den Killer-Bot kontinuierlich laufen lässt, findet man schon zur Entwicklungszeit die unbeachteten Schwachstellen und vergrößert die Resilienz der gesamten Applikation.

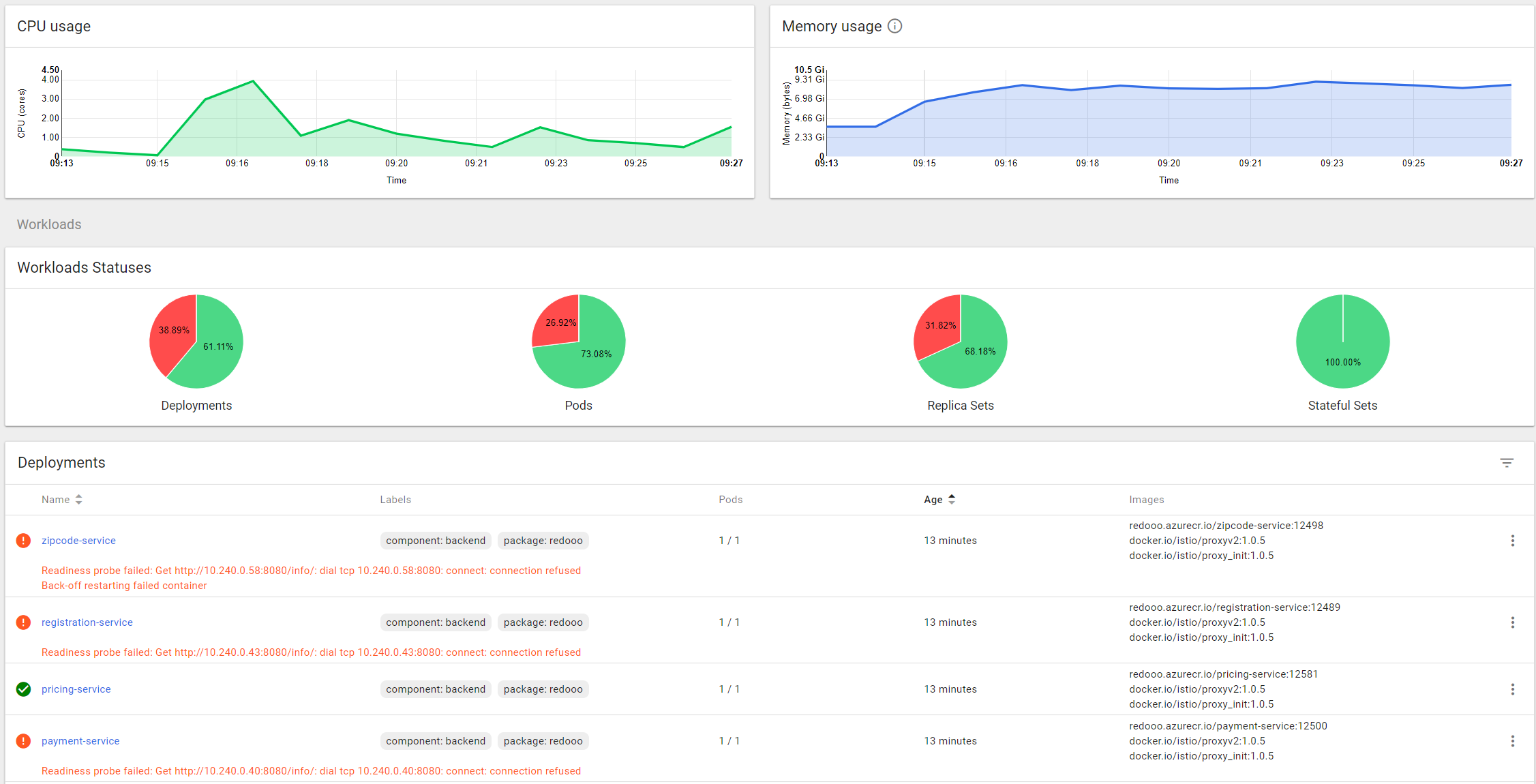


Abbildung 1: Ein mögliches „Werk“ des Killer-Bot für ein zu klein dimensioniertes Kubernetes Cluster