

@codecentric

# **Gamification of Chaos Engineering with K8s: Press Start to Play!**



## Agenda

# Gamification of Chaos Engineering with K8s

3.

1.

2.

## 4. Live Demos

# Why do we need chaos engineering?

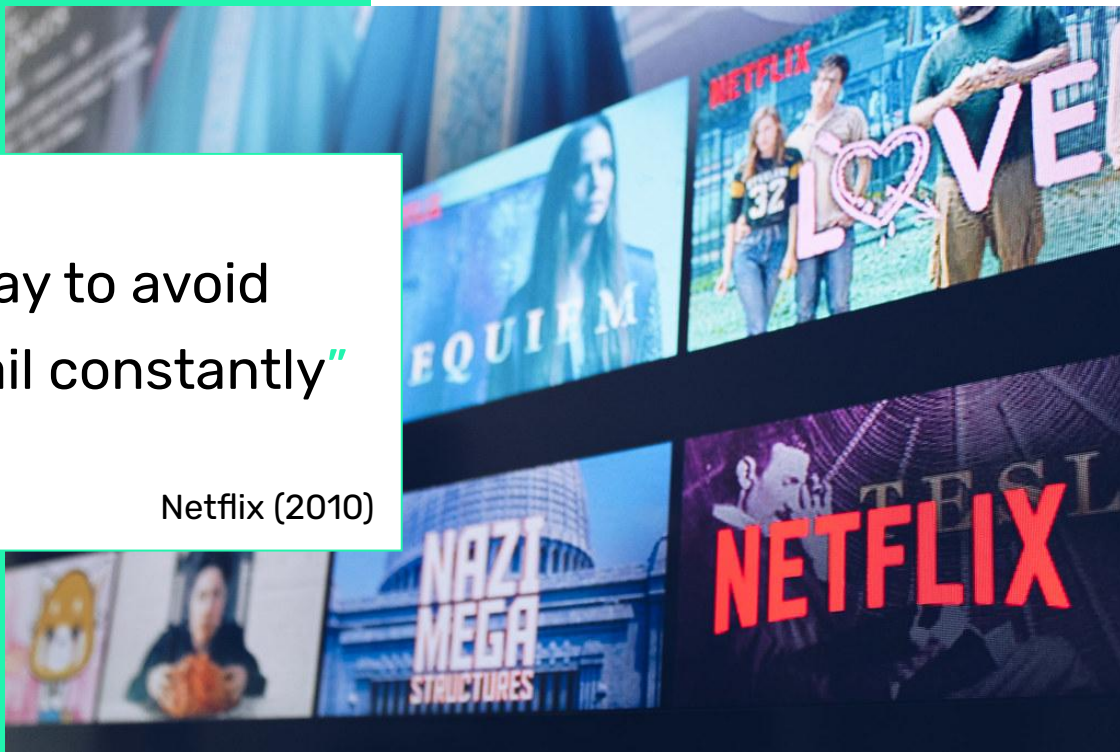


## Netflix goes to the cloud in 2010

“

The best way to avoid failure is to fail constantly”

Netflix (2010)

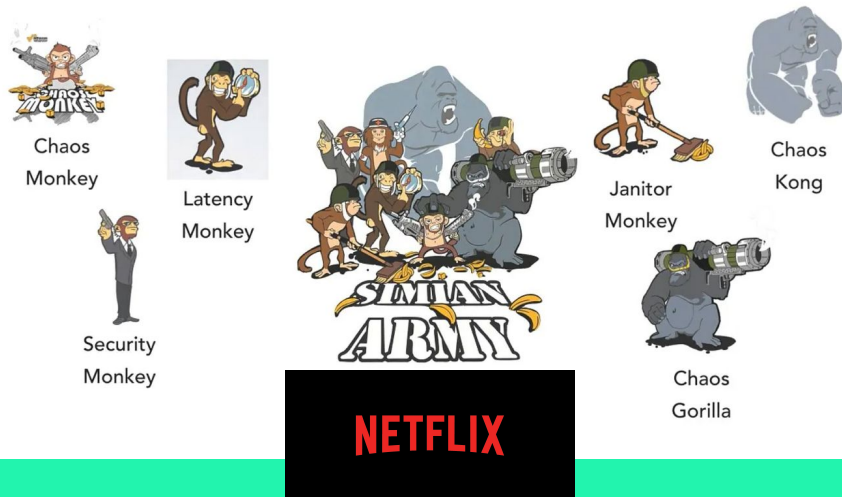


# Netflix Chaos Monkey



# The Netflix Simian Army (2011)

- **Chaos Monkey:** Disables production instances randomly
- **Latency Monkey:** Introduces API response delays
- **Security Monkey:** Monitors Policy changes and alerts on insecure configurations
- **Janitor Monkey:** Cleans out unused resources
- **Chaos Kong:** Simulates AWS region outages
- **Chaos Gorilla:** Simulates entire AWS Availability Zone outage



# What is Chaos Engineering?

Chaos Engineering is the discipline of performing experiments on a distributed system under a controlled environment in order to build confidence in the system's capability to withstand turbulent conditions in production (<https://principlesofchaos.org/>).

Chaos Engineering may give insights to improve the **reliability** of a distributed system. This is done by increasing both, **resilience** and **robustness**.



# How is Chaos Engineering done currently?

- A Development Team agrees on having a “Game Day”
- The Team should execute one or more chaos experiments on the “Game Day”
  - a. Hypothesis
  - b. ‘Steady State’ definition
  - c. Incident Simulation
  - d. Hypothesis and Steady State verification



[https://commons.wikimedia.org/wiki/File:Wikimedia\\_Foundation\\_SOPA\\_War\\_Room\\_Meeting\\_AFTER\\_BLACKOUT\\_1-17-2012-1-3.jpg](https://commons.wikimedia.org/wiki/File:Wikimedia_Foundation_SOPA_War_Room_Meeting_AFTER_BLACKOUT_1-17-2012-1-3.jpg)



## Resilience

*The ability of a system to absorb a failure and dynamically adapt in order to continue working as intended.*

*Examples:*

- *Kubernetes Infrastructure:*
  - *Self-Healing*
  - *Autoscaling*
  - *Rolling Updates and Rollbacks*

## Robustness

*The ability of a system to avoid or withstand failures without changing or adapting dynamically to continue working properly*

*Examples:*

- *Software Development in Java:*
  - *Error handling*
  - *Input Validation*
  - *Fail-Safe*

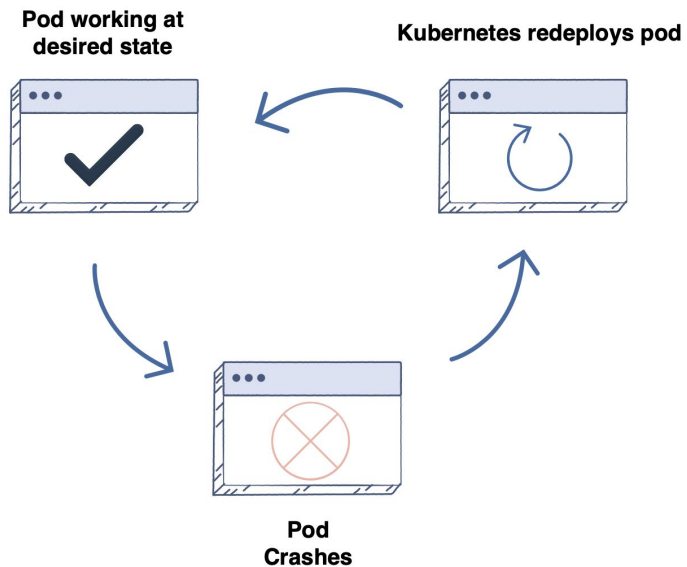
## Kubernetes

Kubernetes is an open source orchestration system that manages containerized applications within a cluster. It has **self healing** features for broken application instances.

The self healing example for kubernetes is a good example for a system **resilience**.



# How does self-healing in Kubernetes work?



The self-healing cycle in Kubernetes

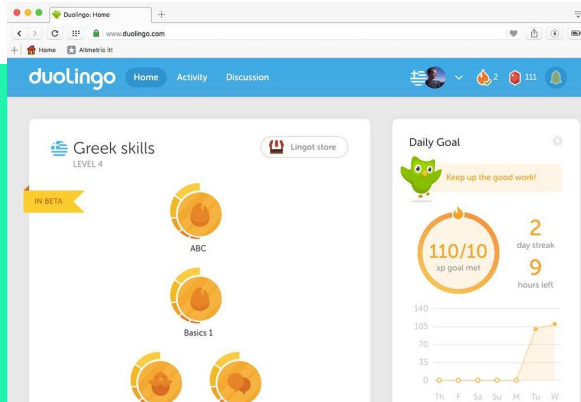
<https://www.educative.io/answers/what-is-self-healing-in-kubernetes#>



# What is gamification?

Gamification is the use of gaming elements and mechanics in non gaming environments. These elements can be leaderboards, points and badges. Gamification can be used to introduce sensations of fun, pleasure and surprise, causing motivation and engagement for certain activities.

XP and Scores to learn languages in Duolingo



Based on “The Landlords Game” in 1902 to teach “Law of rent” to children.



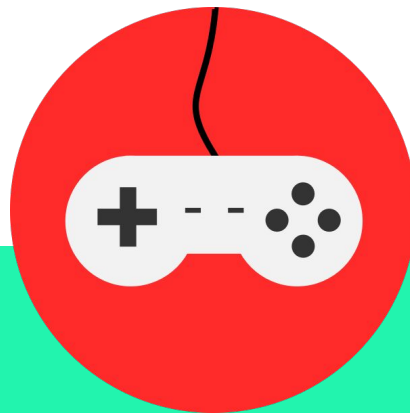
# Is there any gamification tool for all these topics?



Chaos Engineering



Kubernetes



Gamification

# Kubeinvaders

- Resembles the retro game "Space Invaders"
- Keeps in-game scores after execution
- Represents Kubernetes Pods as aliens
- Pods are killed if an alien is shot
- Easy to install using either Helm Charts or docker containers
- Access through browser
- Full Automatic mode to let the Chaos Experiment loosely
- **Use Cases:**
  - Test how resilient Kubernetes clusters are on unexpected pod deletion
  - Collect metrics like pod restart time
  - Tune readiness probes

<https://github.com/lucky-sideburn/kubeinvaders>



Actual (boring) code that could do the same

```
#!/bin/bash
while true
do
  echo "Choosing a pod to kill..."

  PODS=$(oc get pods | grep -v NAME | awk '{print $1}')
  POD_COUNT=$(oc get pods | grep -v NAME | wc -l)

  K=$(( ( RANDOM % $POD_COUNT ) + 1 ))

  TARGET_POD=$(oc get pods | grep -v NAME | awk '{print $1}' | head -n ${K} | tail -n 1)

  echo "Killing pod ${TARGET_POD}"
  oc delete pod $TARGET_POD

  sleep 1
done
```

# DEMO

<http://kubeinvaders.codecentric.link>



<http://grafana.codecentric.link/d/EmUBHUFCh/website-monitoring?orgId=1&refresh=5s>





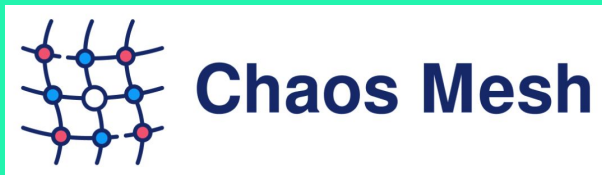


# KubeDOOM

<https://github.com/storax/kubedoom>

- Resembles the classic video game DOOM from Id Software
- Pods are represented by monsters
- Pods are killed when the monster dies
- Different Chaos Experiments can be done by choosing different weapons
- Runs mainly as docker container
- Needs a VNC (Virtual Network Computing) Client to access the video game

# Should I use Kubeinvaders and KubeDOOM in production?





Both projects are well suited for presentations and an introduction to the chaos engineering concepts.


Currently, there are a lot of chaos engineering tools for different technologies and use cases.

- **Chaos Mesh:** Simulates faults in Kubernetes
- **Gremlin SaaS:** Tool to create Chaos Experiments
- **Chaos monkey for Spring Boot (by codecentric)**




 **codecentric AG**  
Gartenstraße 69a  
76137 Karlsruhe

 **Oswaldo Montenegro**  
IT - Consultant | Software Engineer  
[oswaldo.montenegro@codecentric.de](mailto:oswaldo.montenegro@codecentric.de)  
[www.codecentric.de](http://www.codecentric.de)

 **Telefon: +49 (0) 151 51 53 7762**





**THINK  
OUTSIDE  
THE BOX**

**...with us!**

**karriere @codecentric**