codecentric

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

Agenda

Gamification of **Chaos Engineering** with **K8s**

3. 1. 2.

4. **Live Demos**
5. **Summary**

Gamification of **Chaos Engineering** with **K8s**

Butters as “Professor Chaos” from South Park

Why do we need chaos engineering?

Software incidents are not a question of *if* they will occur, but rather *when* they will happen

Netflix goes to the cloud in 2010

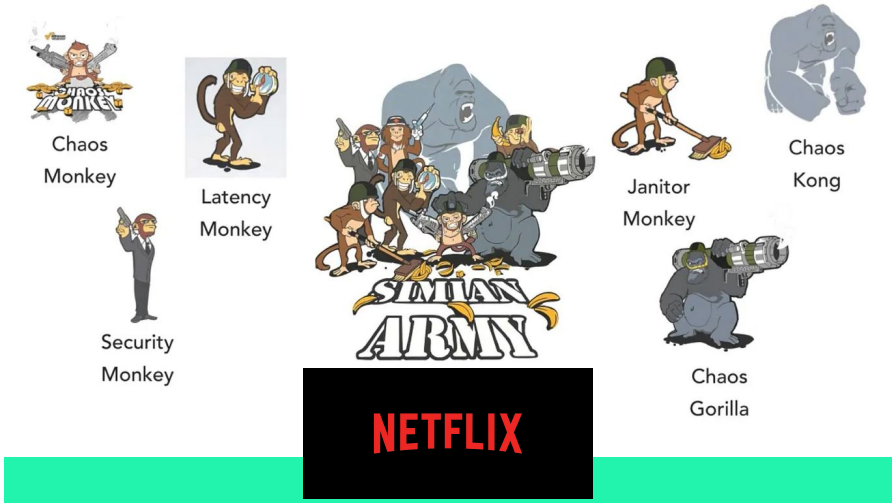
> "The best way to avoid failure is to fail constantly"
>
> Netflix (2010)

https://netflixtechblog.com/5-lessons-weve-learned-using-aws-1f2a28588e4c

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



https://medium.com/@alaison377/from-chaos-comes-order-a-journey-into-chaos-engineering-a3a8bc94d807

Chaos Engineering Definition

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/).

From "Chaos Engineering GameDay" by Manuel Wessner

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. Brainstorm 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

From “Chaos Engineering GameDay” by Manuel Wessner

How would chaos engineering be applied to this problem?

How would chaos engineering be applied to this problem?

1. Controlled Environment

2. The whole team is present

3. Cause Incident

4. Repair Incident

5. Write down findings and tasks

Resilience 🥋

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

The system **steps out** of the **Steady State temporarily.**

Examples:

- Kubernetes Managed 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.

It **avoids stepping out** of the **Steady State**.

Examples:

- Software Development in Java:

  - Error handling
  - Input Validation
  - Fail-Safe

Gamification of **Chaos Engineering** with **K8s**

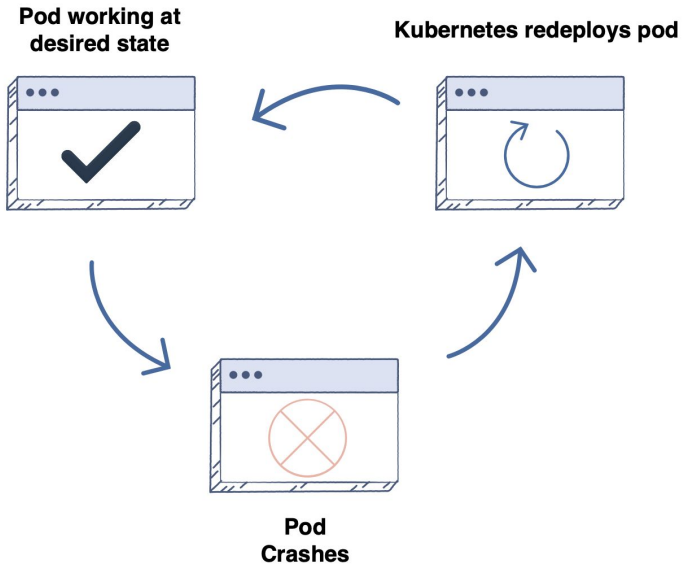
kubernetes

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#

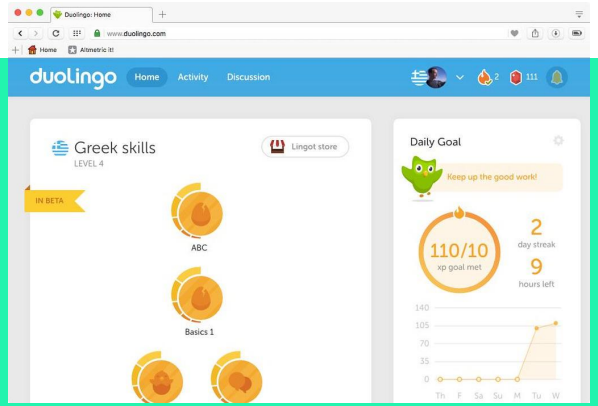
Gamification of **Chaos Engineering** with **K8s**

Oswaldo and Manuel playing “Donkey Kong Country” at codecentric Summer Event 2023, has nothing to do with “gamification” but its a cool picture

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

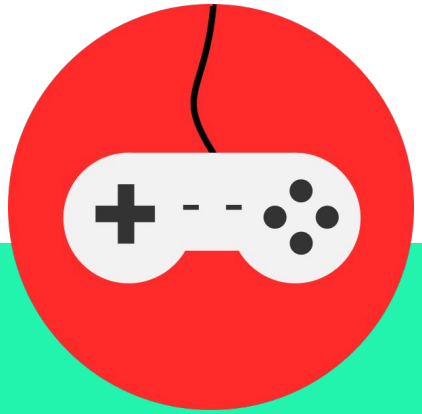


Based on “The Landlord's Game” in 1902 to teach “Law of rent” to children.

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
```

Chaos Experiment: Kubeinvaders

Type of attack	Continuous unexpected Pod deletion
Target	NGINX Deployment running under http://nginx.codecentric.link
Hypothesis	The NGINX application should be **accessible** at least every **5 seconds** as long as I do not delete all the pods at the same time
Blast Radius	Kubernetes Pods in “nginx” **namespace. One pod at a time.**
Details of Attack	Nginx Pods will be randomly deleted using “Kubeinvaders”.
Rollback	Stopping shooting Pods should recover the replica count
Steady-State	Grafana Dashboard shows “Green Dots” on Panel. “Green Dots” means a **200 HTTP** Response for the http://nginx.codecentric.link application every **5 seconds**.

LIVE DEMO

http://kubeinvaders.codecentric.link

http://grafana.codecentric.link/d/EmUBHUFGh/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

Chaos Experiment: Shotgun

Type of attack	Continuous unexpected Pod deletions
Target	NGINX Deployment running under http://nginx.codecentric.link
Hypothesis	The NGINX application should be **accessible** at least every **5 seconds** as long as I do not delete all the pods at the same time
Blast Radius	Kubernetes Pods in “nginx” **namespace. One or Two Pods at a time.**
Details of Attack	Nginx Pods will be randomly deleted using **Shotgun**.
Rollback	Stopping shooting Pods should recover the replica count
Steady-State	Grafana Dashboard shows “Green Dots” on Panel. “Green Dots” means a **200 HTTP** Response for the http://nginx.codecentric.link application every **5 seconds**.

Chaos Experiment: BFG9000

Type of attack	Multiple unexpected Pod deletion
Target	NGINX Deployment running under http://nginx.codecentric.link
Hypothesis	The NGINX application should be **accessible** at least every **5 seconds** as long as I do not delete all the pods at the same time
Blast Radius	Kubernetes Pods in “nginx” **namespace. Many pods may be deleted at the same time.**
Details of Attack	Nginx Pods will be randomly deleted using **BFG9000 (Big F*** Gun 9000)**.
Rollback	Stopping shooting Pods should recover the replica count
Steady-State	Grafana Dashboard shows “Green Dots” on Panel. “Green Dots” means a **200 HTTP** Response for the http://nginx.codecentric.link application every **5 seconds**.

Chaos Experiment: Rocket Launcher

Type of attack	Multiple unexpected Pod deletion
Target	NGINX Deployment running under http://nginx.codecentric.link
Hypothesis	The NGINX application should be **accessible** at least every **5 seconds** as long as I do not delete all the pods at the same time
Blast Radius	Kubernetes Pods in “nginx” **namespace. All pods may be deleted at the same time.**
Details of Attack	Nginx Pods will be randomly deleted using **Rocket Launcher**.
Rollback	Stopping shooting Pods should recover the replica count
Steady-State	Grafana Dashboard shows “Green Dots” on Panel. “Green Dots” means a **200 HTTP** Response for the http://nginx.codecentric.link application every **5 seconds**.

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 ToolKIT:** CLI Tool that helps drive and run chaos experiments.
- **Chaos monkey for Spring Boot (by codecentric):** Simulates Runtime Exceptions, High Latency, Memory and CPU assaults on JVM

Summary

- Ongoing Chaos Engineering “Game Days” are a great addition on top of automatic testing
- Chaos Engineering is a great way to understand your system better and find technical debt before the end user does
- Kubernetes **Self Healing** is great but it has its limits. Make sure to tailor your deployment and application in order to meet your **specific requirements**.
- Gamification is a great way to encourage people but it also has its limits
- **Kubeinvaders** and **KubeDOOM** should **not** be taken too seriously, but don’t underestimate their potential to inspire people.

codecentric

codecentric AG
Gartenstraße 69a
76137 Karlsruhe

Oswaldo Montenegro
IT - Consultant | Software Engineer
oswaldo.montenegro@codecentric.de
www.codecentric.de

Telefon: +49 (0) 151 51 53 7762

SCAN TO GET THE SLIDES!

...with us!

karriere codecentric