ChaOps

Chaos Engineering in a world of DevOps
Who am I?
DevOps* minded open, Cloud focused Open Source enthusiast with an increasingly growing passion for Containers, Orchestrators, Microservices and Chaos Engineering.

*(a lot) more Ops than Dev

Red Hat Premier Business Partner
Chaos Engineering in a world of DevOps
DevOps?
IMMIGRATION IS A LOT LIKE DEVOPS
YOU NEED CONTAINERS

ONE DOES NOT SIMPLY
DO DEVOPS WITHOUT CULTURE CHANGE

BACK IN MY DAY,
DEVOPS WERE JUST DEVS

THE 1ST RULE OF DEVOPS CLUB
IS TO CONTINUOUSLY DEPLOY INTO PRODUCTION

AUTOMATIC BUILD, CONTINUOUS TEST AND CONTINUOUS DEPLOY ISSUES?
WELCOME TO DEVOPS !!!

"DEVOPS"
YOU KEEP USING THAT WORD, I DO NOT THINK IT MEANS WHAT YOU THINK IT MEANS
What is DevOps **NOT**?
DevOps is **NOT** just tools...
YOU FOOL!
It’s not about the tools, it’s about the CULTURE!
DevOps is **NOT** just a project...
WHO ARE THESE GUYS?

THEY ARE DEVOPS CONSULTANTS AND THEY'RE HELPING US WITH OUR DEVOPS PRACTICES.
THEY LOOK LIKE A BAND FROM THE EARLY 80'S.

WE'RE HERE TO WHIP YOUR DEVOPS PRACTICES INTO SHAPE.
CRACK THAT WHIP!
DevOps is **NOT** development + operations...
It's not my problem, the hole is in their side of the boat!
DevOps is **NOT** a separate team...
you come across a bit snobby in some of these...

Speak up - the acoustics in my ivory tower are TERRIBLE
DevOps is **NOT** just copy/paste...
WE ARE NOT NETFLIX
DevOps is **NOT** foolproof...
i wonder if this will bounce

maybe that one will
ChaOps

Chaos Engineering in a world of DevOps
Chaos Engineering?
“Chaos Engineering is the discipline of experimenting on a distributed system in order to build confidence in the system’s capability to withstand turbulent conditions in production.”

- Netflix -

http://principlesofchaos.org/
Why tempt fate?
“If it can go wrong, it will...”

- Murphy -
“Anything that can go wrong, will - at the worst possible moment.”

- Finagle -
"I find that if you want to reach your goal sooner, double your failure rate."
TESTING IS FOR WIMPS

REAL MEN TEST IN PRODUCTION
How to start with Chaos Engineering?
Step 1: Define your “steady state”
Step 2: Design for failure
Step 3: Start small
Step 4: Learn & Improve
Chaos Engineering tools?
ChaOps

Designing for failure
Immutable Infrastructure
Infrastructure as Code
Infrastructure as Code
Infrastructure as Code

MacSteve: scalable-openshift tescst$ ansible-playbook -i inventory/ec2.py destroy.yml -e @vars/eu.yml

PLAY [tag_master_eu[20]] *********************************************************************************************************************

TASK [Delete DNS record] *********************************************************************************************************************
changed: [35.159.0.118]

PLAY [localhost] ****************************************************************************************************************************************

TASK [Get all health checks] ***************************************************************************************************************************
ok: [localhost]

TASK [Remove demo app DNS route] ***********************************************************************************************************************
changed: [localhost] => (item="["HealthCheckConfig": {"FullyQualifiedDomainName": "autoscaler.eu.gluo.io", "ResourcePath": "/", "Inverted": False, "MeasureLatency": False, "RequestInterval": 30, "Type": "HTTP", "Port": 80, "FailureThreshold": 3}, "CallerReference": "88e605d4-45cb-4631-925b-875e5240a463", "HealthCheckVersion": 1, "Id": "4753c64c-54bc-4959-8407-3604cf6fc730"]")

skipping: [localhost] => (item="["HealthCheckConfig": {"FullyQualifiedDomainName": "autoscaler.us.gluo.io", "Inverted": False, "MeasureLatency": False, "RequestInterval": 30, "Type": "HTTP", "Port": 80, "FailureThreshold": 3}, "CallerReference": "bf5ff847-8b6c2-437a-8e9f-3a7febf6c93b", "HealthCheckVersion": 2, "Id": "bf5ff847-8b6c2-437a-8e9f-3a7febf6c93b"]")

TASK [Remove demo app health check] ************************************************************************************************************************
changed: [localhost]

TASK [Remove node lifecycle hooks] *********************************************************************************************************************
changed: [localhost] => (item="["group": "asg-nodes", "name": "scaleup-nodes"]")
changed: [localhost] => (item="["group": "asg-nodes", "name": "scaledown-nodes"]")

changed: [localhost] => (item="["group": "asg-master", "name": "scaleup-masters"]")
changed: [localhost] => (item="["group": "asg-master", "name": "scaledown-masters"]")

TASK [Delete auto scaling groups] **************************************************************************************************************************
changed: [localhost] => (item=asg-master)
changed: [localhost] => (item=asg-nodes)
Build resilience into your applications
Build resilience into your applications
Readiness Checks
Readiness Checks

Service (Load Balancer)

Pod (Application replica) ready

Pod (Application replica) not ready (starting)

Without readiness check
Readiness Checks

Service
(Load Balancer)

Pod
(Application replica)
ready

Pod
(Application replica)
not ready (starting)

Without readiness check: 50% failure (when 2nd replica starting)
Readiness Checks

Service (Load Balancer)

Pod (Application replica) ready

Pod (Application replica) not ready (starting)

With readiness check
Readiness Checks

Service
(Load Balancer)

Pod
(Application replica)
ready

Pod
(Application replica)
ready

With readiness check
Liveness Checks
Liveness Checks

Pod
(Application replica)
ready
Liveness Checks

Pod (Application replica) ready
Liveness Checks

Pod
(Application replica)
frozen

Without liveness check: nothing happens… (your custom monitoring might pick it up at some point, or if it doesn’t your users will start complaining)
Liveness Checks

With liveness check: frozen pod is killed, and new one automatically started (self healing without manual intervention)
Self-healing systems (orchestrator)
Self-healing systems (orchestrator)
Canary Releases
Canary Releases

Service (Load Balancer)

Pod v1 Pod v1 Pod v1 Pod v1 Pod v1 Pod v1 Pod v1 Pod v2
ChaOps
Unleash the chaos monkeys!
Availability Zone
http://demo.gluo.io
http://demo.eu.gluo.io

node 1
node 2
node 3

eu-central-1a

http://demo.gluo.io
http://demo.eu.gluo.io
Container
CHAOS MONKEY

DEMO

EXPECT FAILURE
ChaOps

Takeaways
1. Design for failure (in all layers: network, infrastructure, application,...)
2. Embrace failure (learn from it!)
3. DevOps tools/mindset enable Chaos Engineering
4. Start small, grow gradually (not everybody is Netflix and that is OK!)
5. Don’t fight the chaos zombies by yourself, use tools that help you fight them!