Review SME Continous Call (F4Fp – SME) CHAOS@FIRE

Miguel Robredo

Mandarine Juice

Virtual FEC 10

January 27th 2022

Outline

- **Experiment description**
  - Concept and Objectives
  - Background and motivation
  - Experiment set-up
- **Project results**
  - Measurements
  - Lessons learned
- **Business impact**
  - Impact on our business… How did Fed4Fire helped us?
  - Value perceived… Why did we come to Fed4Fire?
- **Feedback**
  - Used resources and tools
  - Added value of Fed4Fire

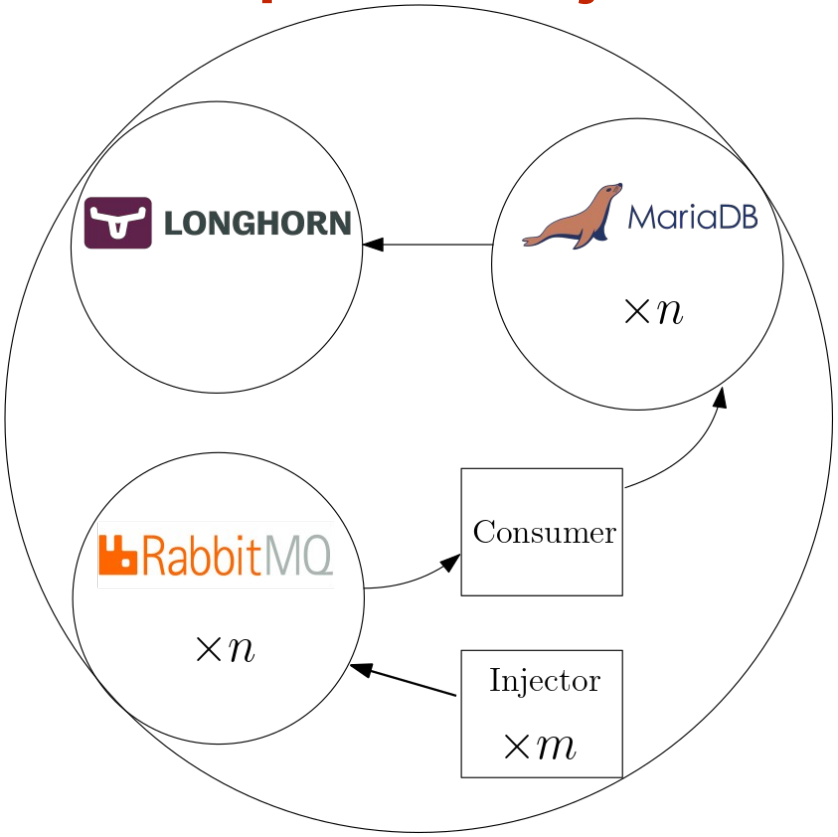
Experiment Description

Background and Motivation

Mandarine Juice is a Venture Builder, whose mission is to help startups to develop innovative solutions to the market. We do so by providing seed investment to entrepreneurs but also by providing consulting services on different areas as strategy, financial or technology, empowering entrepreneurs to focus on the innovative aspects of their proposal.

Current technology trends addresses the construction of technology platforms based on, usually, a large number of services that are replicated to provide high availability. In this context, in 2012 came up the concept of Chaos Engineering: the idea of killing random instances to test a redundant architecture to validate that any failure did not impact noticeably on the overall service.

Concept and Objectives



Objectives

Main goals

O1. Chaos engineering test a typical production environment

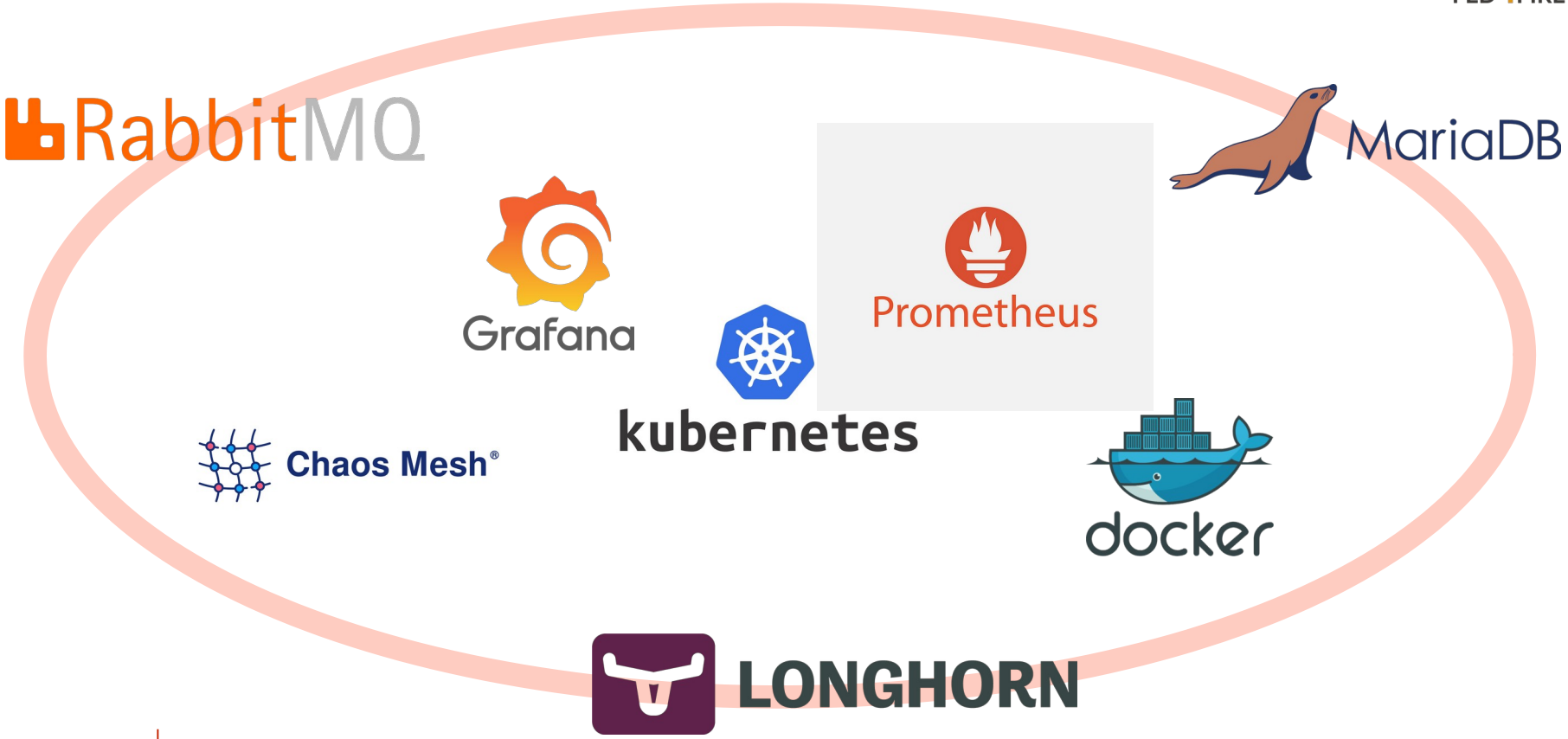
- Test **broker and database performance** while scaling up the volume of messages
- Test **replication persistence** resistance

Side effects

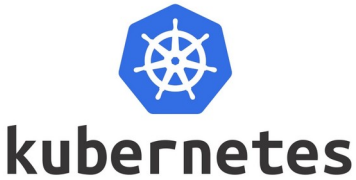
O2. Dimension the infrastructure required for scaling up

O3. Develop a **chaos engineering workflow**

Experiment Setup



Experiment Setup



- Officially, it is a container orchestrator
- In real life is a game changer on how applications are designed, deployed and operated
- Need to rethink existing aplications to convert to a microservice architecture
- The promise is: *«Just throw me more resources (CPU, RAM and Storage), and I will handle the scalability and reliability of the application»*
- This is ok with stateless services, but how to deal with persistent storage to be distibuted and reliable?

Experiment Setup

- Cloud-native distributed persistent block storage for Kubernetes
  - Define persistent volumes to be used by K8 pods, and it will take care of replication
  - Backups to a S3 storage
  - Relying on Longhorn to store tablespaces for MariaDB

Experiment Setup

Fork of the MySQL relational database management system

- Deployed in a HA configuration (Statefulset)
- Two replication mechanism are going on here at the same time:
  - MariaDB cluster storage replication
  - Longhorn block storage replication

Experiment Setup

RabbitMQ

- Open source message broker
  - Easily deployed high availability
  - It allows delivery acknowledgements
  - Easily configurable cluster
  - Compatible with Python through Pika

Experiment Setup

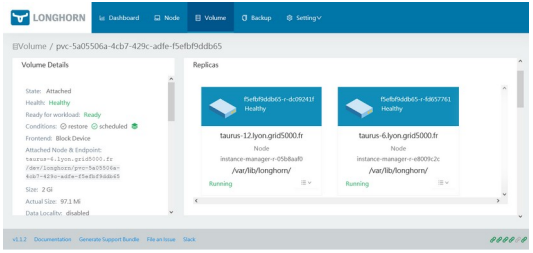
Chaos Mesh®

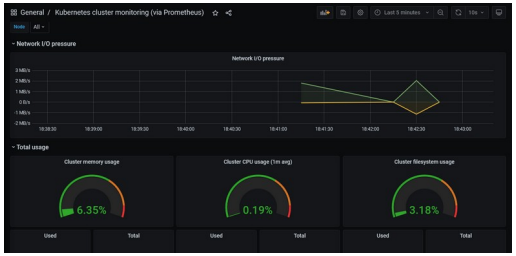
- Open source cloud-native Chaos engineering platform
- Easily deployed without reconfiguring the Kubernetes cluster
- Failure simulation: container, pod, network, system time failures

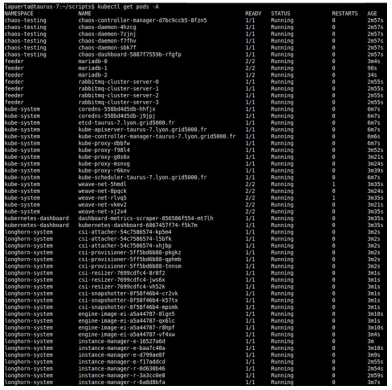
Project Results

Measurements

- Written family of scripts for creating a Kubernetes cluster with HA microservices in Grid5000
- Deployed tools: weave net, nginx, Longhorn, mariaDB (x4), rabbitmq (x4), ChaosMesh
- Deployed monitoring tools: Kubernetes dashboard, Prometheus, Grafana



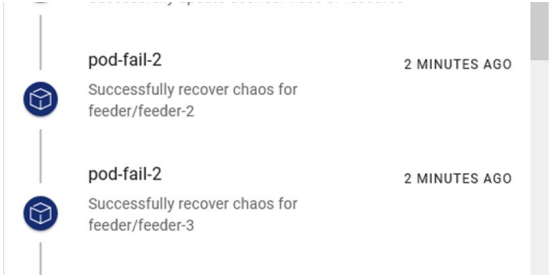


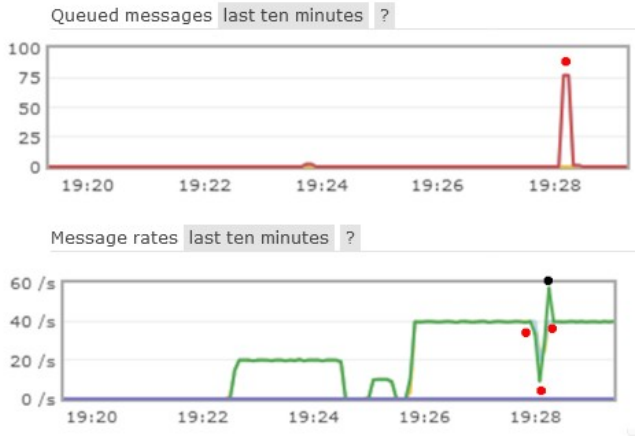


Measurements

Chaos mesh:

Pod fail on feeder → feeder automatically restarts pods → Drop of feeded messages and automatic recovery

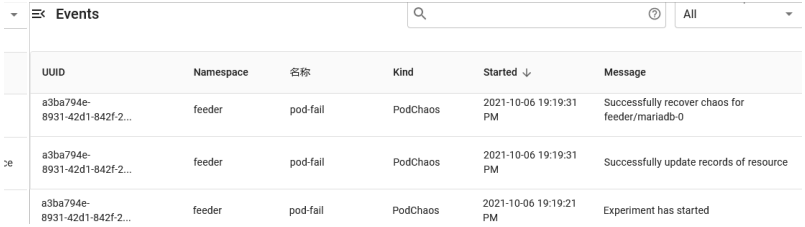


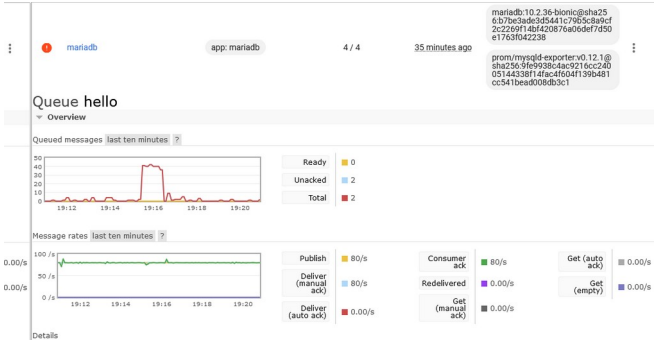


Measurements

Chaos mesh:

mariadb Pod fail → mariadb cluster malfunctioning and little message queueing → Need of manual restoration

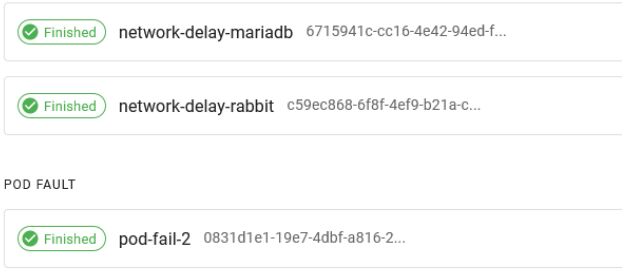


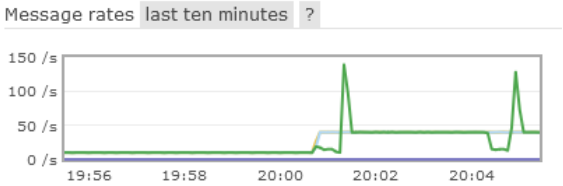


Measurements

Chaos mesh:

force injection latency → acummulation of messages → dump rate drop during the experiment → Need to configure autoscale of the consumer





Lessons Learned

- Fully testing distributed storage requires specific harware config (i.e. min. 4 disks in min. 4 nodes)
- Need to sort out message queueing and latency when system is stressed
- High availability is a must
- Every single app deployed has its own complexity
- Produce highly available software
- Mantain a chaos engineering cycle is crucial
- Monitoring tools are a necessity

Business Impact

Impact on business

- Hands on experience on how to automate Kubernetes cluster creation
- Funding helps to lower the cost of learning curve of the platform
- Knowledge on chaos engineering techniques, stress test development and reliability evaluation
- Practice making HA clusters (Helm, operators, CRD)

Impact on business

- First contact with this paradigm has taught some lessons
  - High availability is a must
  - Push platform to stressful limits
- Need to implement a full chaos engineering cycle
  - Ensure platform responsiveness / availability
  - Ensure resilience / persistence

Value Perceived

- Availability of a large amount of resources:
- Grid5000 is a great infrastructure that can be easily used to test different software possibilities
- Funding helps to lower the cost of learning curve of the platform:
- But furthermore, it allows to dedicate time to experiment, which is important but not urgent in an SME, where costs requiere to be assigned to billable projects

Value Perceived

Being backed by Fed4FIRE, it is possible to target more ambitious projects

We have spent quite a lot of time experimenting with technologies that otherwise would result very difficult (both because the resources but also for the time needed)

Feedback

Used resources and tools

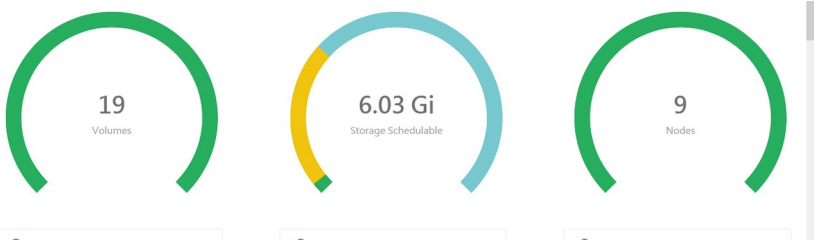
- Testbed: **Grid5000**
  - We used Nova and Taurus nodes in Lyon site to create the Kubernetes cluster
  - Although tutorials on using Terraform to create a Kubernetes cluster in Grid5000 were available, we did prefer to use our own scripts, so we will be able to replicate the infrastructure in our environment in the future

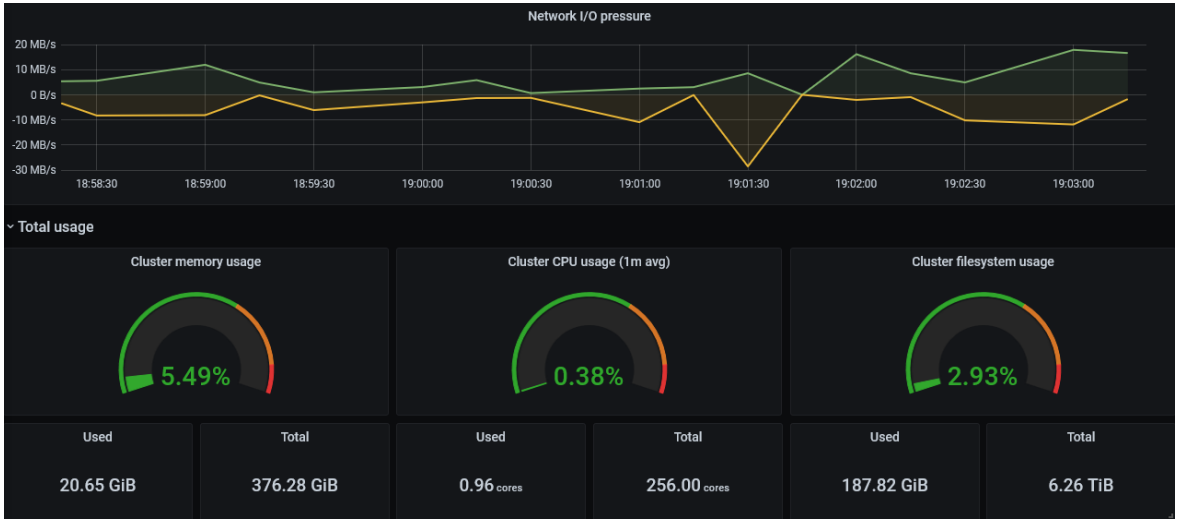
Used resources and tools

- Testbed: **Grid5000**
  - Having such amount of resources ready to be used is the most appreciated value. Also, the good documentation to get hands on quickly is a plus.
  - The header shell is astonishingly easy to use
  - oarsub is a powerful instantiation and automatization tool
  - Reliable nodes (none hurt during experimentation)

Used resources and tools

- Testbed: **Grid5000**





Added value of Fed4FIRE

- Diversity of available resources
  - Plenty of nodes
  - Plethora of hardware configurations
  - High bandwidth
- Documentation
  - Ease of use from day one
  - Different deployment configurations
- Easy setup
  - Our experiment was trivial to deploy once we decided how to deploy each tool

Open questions

- After completing the experiments, there are a bunch of open questions that we couldn't face this time like:
    - Stress test Kafka cluster (latency, availability)
    - Replication and stress test InfluxDB
    - Set a cluster-wide logging system (ECK)

We expect to have answers on this questions with new experiments on Fed4FIRE testbeds.


Co-funded by the European Union


Co-funded by the Swiss Confederation

This project has received funding from the European Union's Horizon 2020 research and innovation programme, which is co-funded by the European Commission and the Swiss State Secretariat for Education, Research and Innovation, under grant agreement No 732638.

WWW.FED4FIRE.EU