

Institut Francophone International

Université Nationale du Vietnam

Dockerisation du MaaS

Pour une architecture Micro-services

Perrault André¹

¹Orange Lab Networks

Encadrant: Luiz-Otávio Gonçalves BURITY

23 Octobre 2018



Sommaire

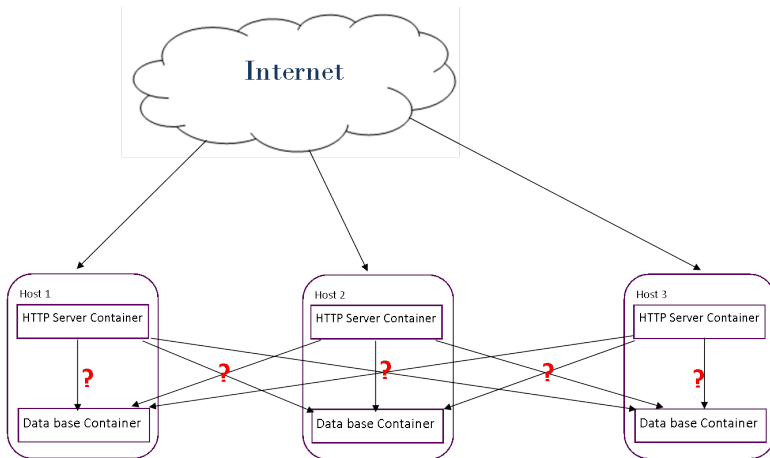
- 1 Introduction
 - Contexte
 - Problématique
- 2 Etat de l'art
 - Tableau de comparaison des Orchestrateurs
 - Tableau de comparaison des GUIs
 - Tableau de comparaison des Overlay
 - Tableau de comparaison des outils d'automatisation
- 3 Solution Proposée
- 4 Implémentation et résultats
- 5 Conclusion et perspectives
- 6 Références

Introduction

Département et équipe

- Orange Labs Networks (OLN) : définit et pilote la stratégie et la politique réseau du groupe.
- Routing IP VPN (RIV) : missions principales, d'être le spécialiste du plan de contrôle et des protocoles de routage des réseaux IP/MPLS, de son monitoring, des mécanismes de sécurité associés et de préparer les évolutions de son automatisation

Contexte général du MaaS¹



1. **Monitoring as a Service (MaaS)** : permet de déployer sur vos postes de travail une application de supervision en ligne.

Problématique

Comment :

- 1 Créer un réseau privé et sécurisé situées sur des réseaux hétérogènes ?
- 2 Mettre en place une architecture d'orchestration open source, sécurisé et utilisable en entreprise ?
- 3 Permettre aux applications de monitoring dockerisées de fonctionner sur le même principe des micro-services ?

Orchestration

Orchestration

- L'orchestration de conteneur désigne le processus d'organisation du travail des composants individuels et des niveaux d'application.

Différents composant de l'orchestration

- Orchestrateur
- GUI
- Overlay
- Outils d'automatisation

Tableau de comparaison des Orchestrateurs

	Nomad	Mesos Marathon	Swarm	Kubernetes
Crée par	Hashicorp	Mesosphere, Microsoft	Docker	Linux Fondations, Google
Année	2015	2011	2013	2014
Type	Conteneur d'orchestration et planification	Conteneur d'orchestration et gestion d'infrastructure	Conteneur d'orchestration et planification	Conteneur d'orchestration et planification
Réseau virtuel	Calico, weave, flannel, etc.	Calico, weave, flannel, etc.	Calico, weave, flannel, etc.	Calico, weave, flannel, etc.
Critères importants				
Niveau de complexité d'installation et de configuration	Moyen	Moyen	Simple	Complexe
Contributeurs	251	259	165	1798
Support pour entreprise	Oui	Oui	Oui	Oui
Plusieurs Datacenter	Oui	Oui	Docker Cloud, Docker Machine Drivers, Docker for AWS/ Azure	Oui
Support Bare Metal	Oui	Oui	Oui	Oui
Gestion des secrets	vault	Oui	Oui	Oui
Montage de volume	-	Oui	Oui	Oui
On-premise	-	Oui	Oui	Oui
Clusters	Jusqu'à 5000 nodes	Jusqu'à 10000 nodes	Jusqu'à 4700 nodes	Jusqu'à 5000 nodes
Pods	-	-	-	Jusqu'à 150000
Conteneurs	Jusqu'à 1000000	-	Jusqu'à 50000	Jusqu'à 300000
Self-healing	Non	Non	Non	Oui

FIGURE – Tableau comparatif entre les orchestrateurs

Tableau de comparaison des GUIs

	Panamax	Kitematic	Shipyard	DockStation	Portainer	Kubernetes Dashboard	Rancher
Gérer les conteneurs docker	Oui	Oui	Oui	Oui	Oui	Oui	Oui
Accéder à la console des conteneurs	Non	Oui	Oui	Oui	Oui	Oui	Oui
Gérer les images de docker	Oui	Oui	Oui	Oui	Oui	Oui	Oui
Tag et push les images docker	Oui	Non	Oui	Oui	Oui	Oui	Oui
Critères importants							
Gérer le réseau de docker	Non	Non	Non	Non	Oui	Oui	Oui
Gérer les volumes de docker	Non	Non	Non	Non	Oui	Oui	Oui
Regarder les événements de docker	Non	Non	Oui	Oui	Oui	Oui	Oui
Préconfigurer les modèles de conteneur	Oui	Non	Non	Non	Oui	Oui	Oui
Vue d'ensemble du cluster	Non	Non	Oui	Non	Oui	Oui	Oui
Niveau de complexité d'installation et/ou configuration	Simple	Simple	Simple	Simple	Simple	Moyen	Moyen
Type de conteneurs et/ou orchestrateurs	Docker	Docker	Docker	Docker	Docker / Swarm	Docker / Kubernetes	Tout
Gérer les Stacks	Non	Non	Non	Non	Oui	Oui	Oui
Gérer les secrets	Non	Non	Non	Non	Oui	Oui	Oui
Contributeurs	14	74	45	2	106	141	46
Gérer les services	Non	Non	Non	Non	Oui	Oui	Oui
Catalogue	Oui	Oui	Non	Oui	Oui	Non	Oui

FIGURE – Tableau comparatif entre les GUIs

Tableau de comparaison des Overlay

	Docker Overlay Network	Flannel	Weave	Calico	Canal
Modèle réseau	VxLAN	VxLAN or UDP Channel	VxLAN or UDP Channel	Pure Layer-3 Solution	Flannel + Calico
Isolation d'application	CIDR Schéma	CIDR Schéma	CIDR Schéma	Profile Schéma	
Protocol Support	Tout	Tout	Tout	TCP, UDP, ICMP & IC-MPV6	
Nom de service	Non	Non	Oui	Nom	
Exigences de stockage distribué	Oui	Oui	Non	Oui	
Chaîne de cryptage	Non	TLS	NaCl Library	Oui	
Support réseau partiellement connecté	Non	Non	Oui	Non	
vNIC séparé pour le conteneur	Non	Non	Oui	Non	
Prise en charge du chevauchement IP	Oui	Oui	Oui	Non	
Restriction de sous-réseau de conteneur	Oui, configurable après démarrage	Non	Oui, configurable après démarrage.	Non	
Approche	Overlay	Routing, overlay	Overlay	Routing	
Spécification	CNM	CNI, CNM	CNI, CNM	CNI, CNM	
Sauvegarde de données externe	Aucun	Aucun	Aucun	Etd	

FIGURE – Tableau comparatif entre les overlays

Tableau de comparaison des outils d'automatisation

	Puppet	Chef	Ansible	Salt
Support	Puppet Labs	Opscode	Ansible Works	saltStack
Année	2005	2009	2012	2011
Références	Google, eBay, Twitter	Facebook, Ancestry, Splunk	Rackspace, Evernote	LinkedIn, HP Cloud
Interface de contrôle	Manifest (DSL)	Recipes (DSL: Ruby)	Playbook (YAML, JSON)	SLS (SoLt Sate/YAML)
Code	Open source	Open source	Open source	Open source
Cloud	Tout	Tout	Tout	Tout
Critères importants				
Sécurité		Chef Vault	Elevé avec SSH	
Prix (Node/année)	Std : \$88 / Prem : \$152	Std : \$72 / Prem : \$137	Up to 100 Nodes Std:\$10000/ Prem:\$14000	Contacter l'entreprise
Communication	http, SSH/SSL	STOMP, rest/SSL	SSH/JSON	ZeroMQ
Contrôleur d'exécution	Mcollective, challenging	Knife, challenging	Built-in, easy	Built-in
Contributeurs	496	557	3616	2105
Type	Gestion de configuration	Gestion de configuration	Gestion de configuration	Gestion de configuration
Langage	Puppet DSL, Ruby	Ruby	Python	Python
Architecture	Client/Server	Client/Server	Client seulement	Client/Server
Difficulté de configuration	Difficile	Difficile	Facile	Facile
Évolutivité	Elevé	Elevé	Elevé	Elevé
Licence	Apache license v2	Apache license v2	GNU Public license	Apache license v2
Plus utilisé	Grande entreprise avec un environnement hétérogène.	Adapter pour le développement.	Adapter facilement pour l'automatisation.	-

FIGURE – Tableau comparatif entre les outils d'automatisation

Solution Proposée

Pour ce travail nous proposons :

- 1 Orchestrateur : Kubernetes
- 2 GUI : Kubernetes Dashboard
- 3 Overlay : Calico
- 4 Outil d'automatisation : Ansible
- 5 Métrique : Prometheus et Grafana

Architecture Orchestration avec Kubernetes

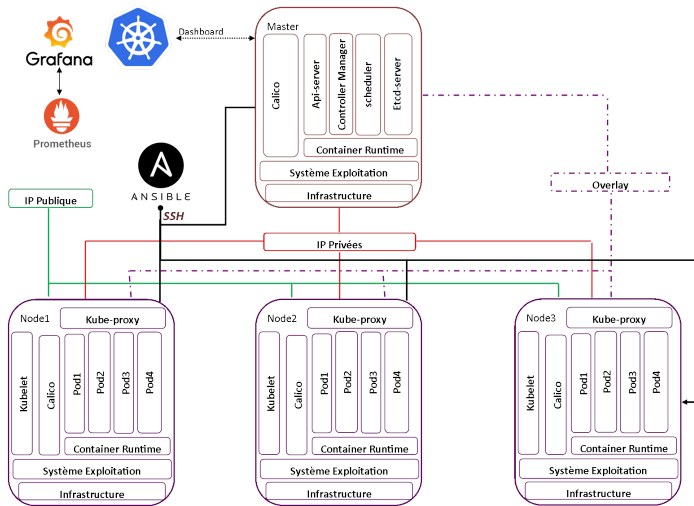


FIGURE – Architecture d'orchestration

Implémentation et résultats

- 1 Installation et configuration des matériels
- 2 Création du réseau privé entre les machines (OpenVPN)
- 3 Installation et configuration des outils nécessaires pour l'orchestration
- 4 Configuration et déploiement automatique de l'orchestration
- 5 Installation et configuration des métriques
- 6 Dockerisation de notre application (Gestion Restaurant)

Représentation de l'espace de travail



FIGURE – Représentation de la connexion entre les machines

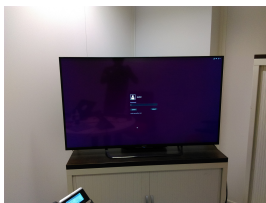
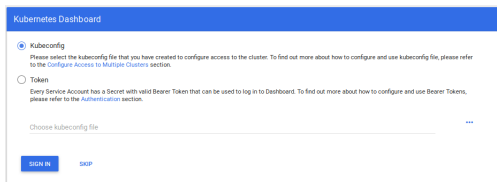


FIGURE – Écran de visualisation

Dashboard login



Kubernetes Dashboard

☒ Kubeconfig

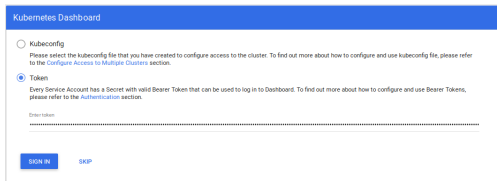
Please select the kubeconfig file that you have created to configure access to the cluster. To find out more about how to configure and use kubeconfig file, please refer to the [Configure Access to Multiple Clusters](#) section.

☐ Token

Every Service Account has a Secret with valid Bearer Token that can be used to log in to Dashboard. To find out more about how to configure and use Bearer Tokens, please refer to the [Authentication](#) section.

Choose kubeconfig file

FIGURE – Dashboard login avec kubeconfig



Kubernetes Dashboard

☐ Kubeconfig

Please select the kubeconfig file that you have created to configure access to the cluster. To find out more about how to configure and use kubeconfig file, please refer to the [Configure Access to Multiple Clusters](#) section.

☒ Token

Every Service Account has a Secret with valid Bearer Token that can be used to log in to Dashboard. To find out more about how to configure and use Bearer Tokens, please refer to the [Authentication](#) section.

Enter token

FIGURE – Dashboard login avec token

Représentation du cluster par le dashboard de Kubernetes

Cluster

Namespaces

Nodes

Persistent Volumes

Roles

Storage Classes

Namespaces

default

Overview

Workloads

Cron Jobs

Daemon Sets

Deployments

Jobs

Pods

Replica Sets

Replication Controllers

Stateful Sets

Discovery and Load Balancing

Ingresses

Services

Config and Storage

Config Maps

Persistent Volume Claims

Secrets

Settings

About

Namespaces

Name	Labels	Status	Age
monitoring	name: monitoring	Active	a day
kube-public	-	Active	a day
default	-	Active	a day
kube-system	-	Active	a day

Nodes

Name	Labels	Ready	CPU requests (cores)	CPU limits (cores)	Memory requests (bytes)	Memory limits (bytes)	Age
node1	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux kubernetes.io/hostname: node1 node-role.kubernetes.io/node: true	True	0.648 (16.20%)	1.523 (38.07%)	618,588 Mi (7.86%)	3,084 Gi (40.15%)	a day
node2	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux kubernetes.io/hostname: node2 node-role.kubernetes.io/node: true	True	0.705 (17.63%)	1.4 (35.00%)	492,500 Mi (6.26%)	3,508 Gi (45.66%)	a day
master	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux kubernetes.io/hostname: master node-role.kubernetes.io/master: true	True	1.04 (26.00%)	2.5 (62.50%)	895,773 Mi (11.39%)	5,836 Gi (76.97%)	a day

Roles

Name	Role Type	Namespaces	Age
kube-prcmethuon-exporter-kube-state	Role	monitoring	a day
pop-kube-prometheus	Cluster Role	All Namespaces	a day
kube-prcmethuon-exporter-kube-state	Cluster Role	All Namespaces	a day

FIGURE – Représentation du cluster par le dashboard de Kubernetes

Représentation des capacité du cluster par Grafana

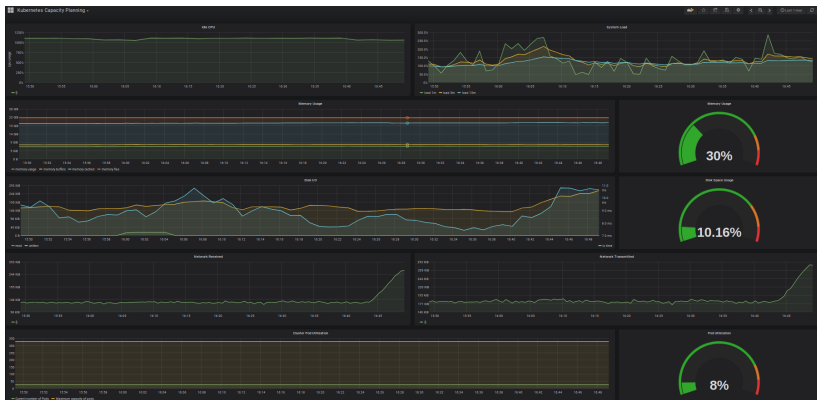


FIGURE – Représentation des capacité du cluster par Grafana

Représentation de l'application

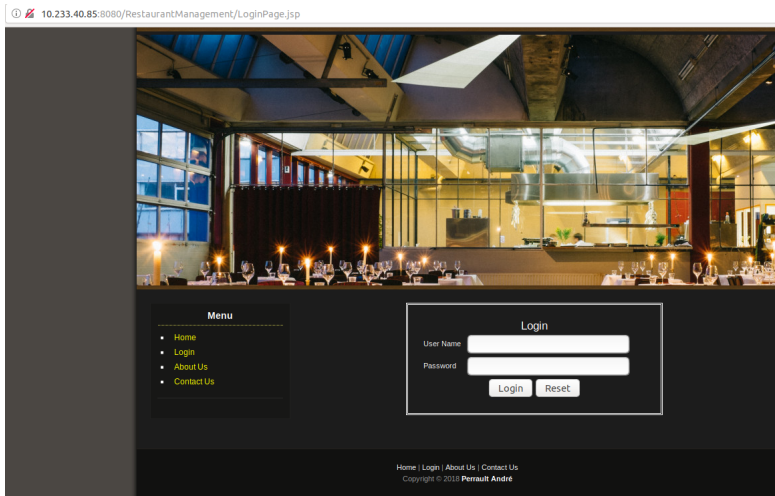


FIGURE – Représentation de l'application

Représentation du déploiement de l'application

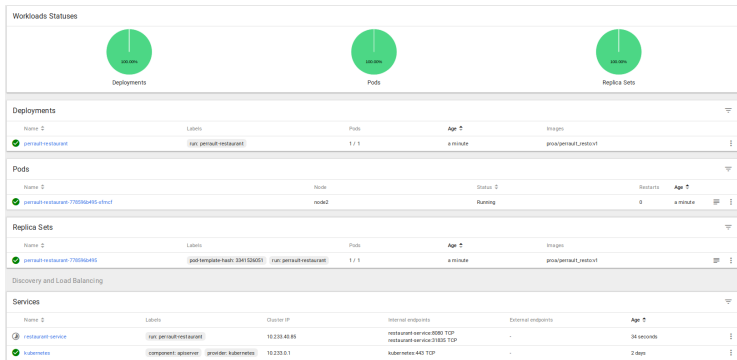


FIGURE – Représentation du déploiement de l'application

Représentation de la mise en échelle de l'application 1

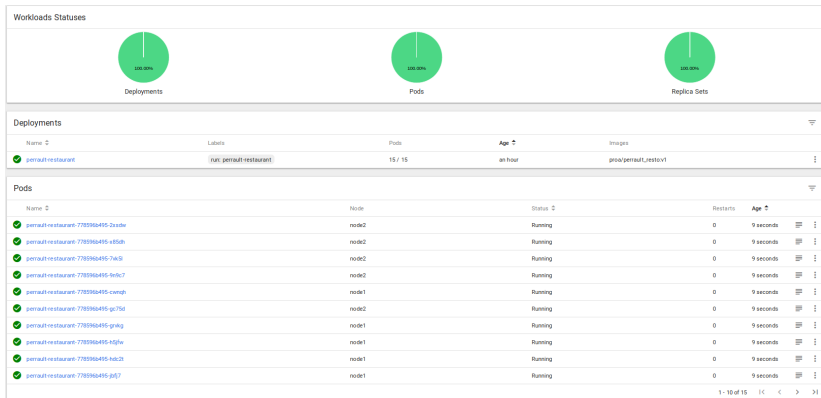


FIGURE – Représentation de la mise en échelle de l'application 1

Représentation de la mise en échelle de l'application 2

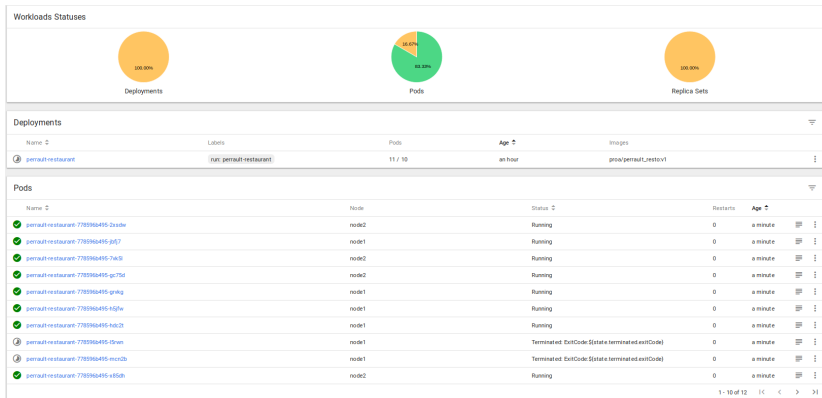


FIGURE – Représentation de la mise en échelle de l'application 2

Représentation de la mise en échelle de l'application 3

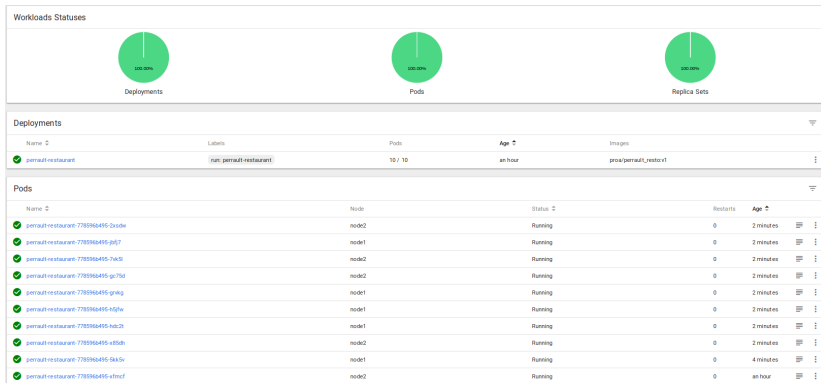


FIGURE – Représentation de la mise en échelle de l'application 3

Représentation des capacité de l'application par le terminal

The screenshot shows a terminal window with a table of application capacity metrics. The table has columns for NAME, REFERENCE, TARGETS, MINPODS, MAXPODS, REPLICAS, and AGE. The data is organized into groups for 'perrault-restaurant' and 'root@master:/home/master'.

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
perrault-restaurant	Deployment/perrault-restaurant	200/300	1	40	1	1m
root@master:/home/master#	kubectl get hpa					
NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
perrault-restaurant	Deployment/perrault-restaurant	300%/300	1	40	1	2h
root@master:/home/master#	kubectl get hpa					
NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
perrault-restaurant	Deployment/perrault-restaurant	300%/300	1	40	4	2h
root@master:/home/master#	kubectl get hpa					
NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
perrault-restaurant	Deployment/perrault-restaurant	200/300	1	40	4	3h
root@master:/home/master#	kubectl get hpa					
NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
perrault-restaurant	Deployment/perrault-restaurant	170/300	1	40	4	5h
root@master:/home/master#	kubectl get hpa					
NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
perrault-restaurant	Deployment/perrault-restaurant	170/300	1	40	4	6h
root@master:/home/master#	kubectl get hpa					
NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
perrault-restaurant	Deployment/perrault-restaurant	320/300	1	40	4	8h
root@master:/home/master#	kubectl get hpa					
NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
perrault-restaurant	Deployment/perrault-restaurant	150/300	1	40	2	9h
root@master:/home/master#	kubectl get hpa					
NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
perrault-restaurant	Deployment/perrault-restaurant	1800/300	1	40	2	9h
root@master:/home/master#	kubectl get hpa					
NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
perrault-restaurant	Deployment/perrault-restaurant	200/300	1	40	2	12h
root@master:/home/master#	kubectl get hpa					
NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
perrault-restaurant	Deployment/perrault-restaurant	200/300	1	40	2	12h
root@master:/home/master#	kubectl get hpa					
NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
perrault-restaurant	Deployment/perrault-restaurant	200/300	1	40	2	14h
root@master:/home/master#	kubectl get hpa					
NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
perrault-restaurant	Deployment/perrault-restaurant	700/300	1	40	4	16h
root@master:/home/master#	kubectl get hpa					
NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
perrault-restaurant	Deployment/perrault-restaurant	700/300	1	40	4	16h
root@master:/home/master#	kubectl get hpa					
NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
perrault-restaurant	Deployment/perrault-restaurant	200/300	1	40	4	17h
root@master:/home/master#	kubectl get hpa					
NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
perrault-restaurant	Deployment/perrault-restaurant	200/300	1	40	4	18h
root@master:/home/master#	kubectl get hpa					
NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
perrault-restaurant	Deployment/perrault-restaurant	170/300	1	40	4	19h
root@master:/home/master#	kubectl get hpa					
NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
perrault-restaurant	Deployment/perrault-restaurant	160/300	1	40	2	25h

FIGURE – Représentation des capacité de l'application par le terminal

Représentation des capacités de l'application par Grafana

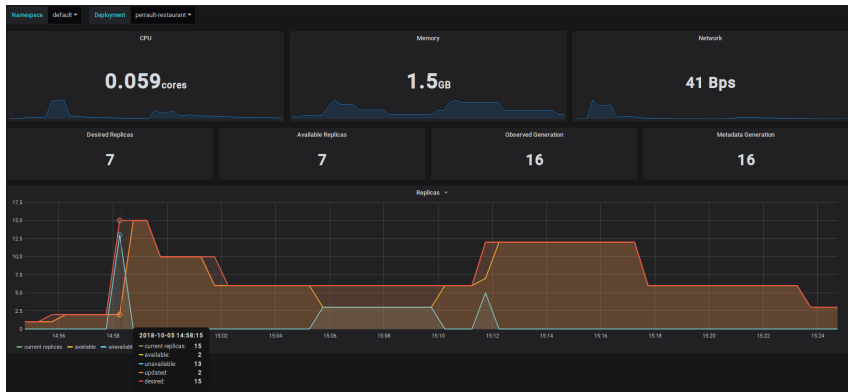


FIGURE – Représentation des capacités de l'application

Représentation du node2, hors service

Nodes							
Name	Labels	Ready	CPU requests (cores)	CPU limits (cores)	Memory requests (bytes)	Memory limits (bytes)	Age
node1	beta.kubernetes.io/arch: arm64 beta.kubernetes.io/os: linux kubernetes.io/hostname: node1 node-role.kubernetes.io/node: node	True	0.473 (14.00%)	1.018 (37.00%)	408,888 MB (3.71%)	3.875 Gi (40.00%)	2 days
node2	beta.kubernetes.io/arch: arm64 beta.kubernetes.io/os: linux kubernetes.io/hostname: node2 node-role.kubernetes.io/node: node	Unknown	0.363 (24.00%)	1.708 (42.70%)	1,044 Gi (13.00%)	4,168 Gi (34.20%)	2 days
master	beta.kubernetes.io/arch: arm64 beta.kubernetes.io/os: linux kubernetes.io/hostname: master node-role.kubernetes.io/master: true	True	1.84 (26.00%)	2.032 (50%)	895,773 MB (11.00%)	5.836 Gi (75.00%)	2 days

FIGURE — Représentation du node2, hors service

Pods						
Name	Namespace	Node	Status	Restart	Age	
perault-reseau-17709b6498-zwolv	default	node2	Running	0	10 minutes	⌵ ⌶
perault-reseau-17709b6498-gt5d	default	node2	Running	0	10 minutes	⌵ ⌶
perault-reseau-17709b6498-gr8q	default	node1	Running	0	10 minutes	⌵ ⌶
perault-reseau-17709b6498-vghw	default	node1	Running	0	10 minutes	⌵ ⌶
perault-reseau-17709b6498-5kxlv	default	node1	Running	0	12 minutes	⌵ ⌶
prometheus-operator-598v49l-g8t8s	monitoring	node1	Running	0	15 minutes	⌵ ⌶
kube-prometheus-exporter-kube-statefulset-5f8vz8f	monitoring	node1	Running	0	15 minutes	⌵ ⌶
cluster-discovery-5k47y-gg9t5	kube-system	node1	Running	0	15 minutes	⌵ ⌶
kube-prometheus-grafana-589v738-vgh8t	monitoring	node1	Running	0	16 minutes	⌵ ⌶
prometheus-kube-prometheus-0	monitoring	node1	Running	1	34 minutes	⌵ ⌶

FIGURE — Représentation des pods du node2, hors service

Représentation de la réattribution des pods sur le node 1

Pods							
Name	Namespace	Node	Status	Restarts	Age		
✓ kube-proxy-node2	kube-system	node2	Running	0	a minute	☰	⋮
✓ nginx-proxy-node2	kube-system	node2	Running	0	a minute	☰	⋮
✓ perrault-restaurant-778596b495-nfzpz	default	node1	Running	0	2 minutes	☰	⋮
✓ perrault-restaurant-778596b495-cqfqn	default	node1	Running	0	2 minutes	☰	⋮
✓ perrault-restaurant-778596b495-fq3v4	default	node1	Running	0	2 minutes	☰	⋮
✓ perrault-restaurant-778596b495-4wdbk	default	node1	Running	0	2 minutes	☰	⋮
✓ perrault-restaurant-778596b495-nbh7s	default	node1	Running	0	2 minutes	☰	⋮
✓ perrault-restaurant-778596b495-qzqv	default	node1	Running	0	2 minutes	☰	⋮
✓ perrault-restaurant-778596b495-bd4f	default	node1	Running	0	4 minutes	☰	⋮
✓ prometheus-operator-85bc485-zxnhn	monitoring	node1	Running	0	4 minutes	☰	⋮
1 - 10 of 43							1 < > 43

FIGURE – Représentation de la réattribution des pods sur le node 1

Représentation des capacité du cluster par Grafana



FIGURE – Représentation des capacité du cluster par Grafana

Conclusion et perspectives

- Nous avons utilisé un ensemble d'outils (Kubernetes, Kubernetes Dashboard, Calico, Ansible, Prometheus et Grafana), sélectionnés à l'aide des critères que nous avons définis afin de mettre en place notre architecture d'orchestration.
- Objectif atteint.
- Projet future.

Références



HUNTER II, Thomas. Advanced Microservices : A Hands-on Approach to Microservice Infrastructure and Tooling. Apress, 2017.



KUTNER, Joe. Deploying with JRuby 9k : deliver scalable web apps using the JVM. Pragmatic Bookshelf, 2016.



VOHRA, Deepak. Kubernetes microservices with Docker. Apress, 2016.



Kubernetes Patterns Patterns, Principles, and Practices for Designing Cloud Native Applications.2017



HOLLA, Shrikrishna. Orchestrating docker. Packt Publishing Ltd, 2015.



FARCIC, Viktor. The DevOps 2.0 Toolkit. Packt Publishing Ltd, 2016.



GIRIDHAR, Chetan. Automate it !-Recipes to upskill your business. Packt Publishing Ltd, 2017.