TRAINING THE NEXT GENERATION OF EUROPEAN FOG COMPUTING EXPERTS

mck8s: A container orchestration platform for geo-distributed multi-cluster environments

Mulugeta Ayalew Tamiru
PhD candidate at FogGuru project, University of Rennes 1
Cloud-native engineer at Elastisys AB

KubeCon + CloudNativeCon North America 2021
October 13, 2021

About me and FogGuru

- Mulugeta Ayalew Tamiru
- PhD candidate at FogGuru / University of Rennes 1
- Cloud native engineer at Elastisys AB

moule3053

@moulougueta

https://www.linkedin.com/in/mulugeta-ayalew-tamiru-a0a2581b/

- Funded by EU
- 6 European organizations
- 8 PhD students
- +20 publications

http://www.fogguru.eu/

FogGuru

@thefogguru

Outline

1. Evolution of cloud deployments
2. Challenges in multi-cluster management
3. Kubernetes Federation
4. Mck8s architecture
5. Mck8s controllers
6. Demonstration

Evolution of cloud deployments

- Cloud environments are increasingly geographically distributed
- Geo-distributed deployments
  - Hybrid-cloud
  - Multi-cloud
  - Fog computing
- Non-functional requirements to address:
  - Performance (low latency)
  - High bandwidth and reliable connectivity
  - HA and disaster recovery
  - Scalability
  - Security
  - Compliance

Hybrid cloud

Combine resources from private and public clouds (Eg. Enterprise applications)

Why?: Low latency, high availability, scalability, data locality, security, privacy, legal restrictions

Multi-cloud

Combine resources from multiple public clouds (Eg. Consumer applications)

Why?: Low latency, high availability, scalability, best-of-breed services, cost, vendor neutrality

Fog computing

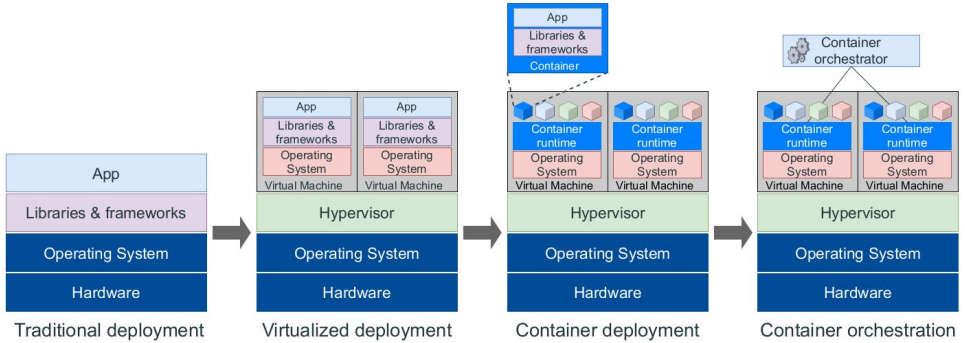
Combine resources from private and public clouds as well as micro data centers with vast geographical distribution (Eg. Telco clouds, IoT)

Why?: Low latency, high bandwidth, reliable connectivity

Challenges in multi-cluster management

- Potentially 100's or 1000's of clusters → automate deployment of applications & resource management
- Several challenges
  - Resilience
  - Placement
  - Autoscaling
  - User traffic routing and load balancing

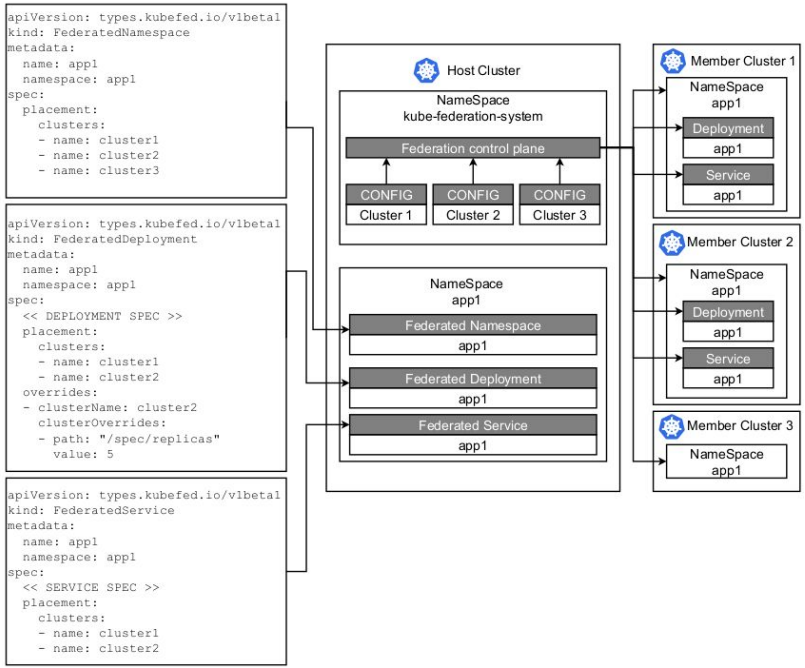
- Container orchestrators are basic foundation for building orchestrators for geo-distributed environments
  - Portable
  - Inter-operable
  - Extensible



Evolution of application deployment from VMs to container orchestration.

Kubernetes Federation (KubeFed)

- Sub-project of Kubernetes SIG Multicluster
- Provides control plane, concepts and abstraction for managing multiple Kubernetes clusters
- Provides manual placement
- Provides fully load-balanced or weight-based placement (Replica Scheduling Preferences)
- More automated policies needed in geo-distributed environments

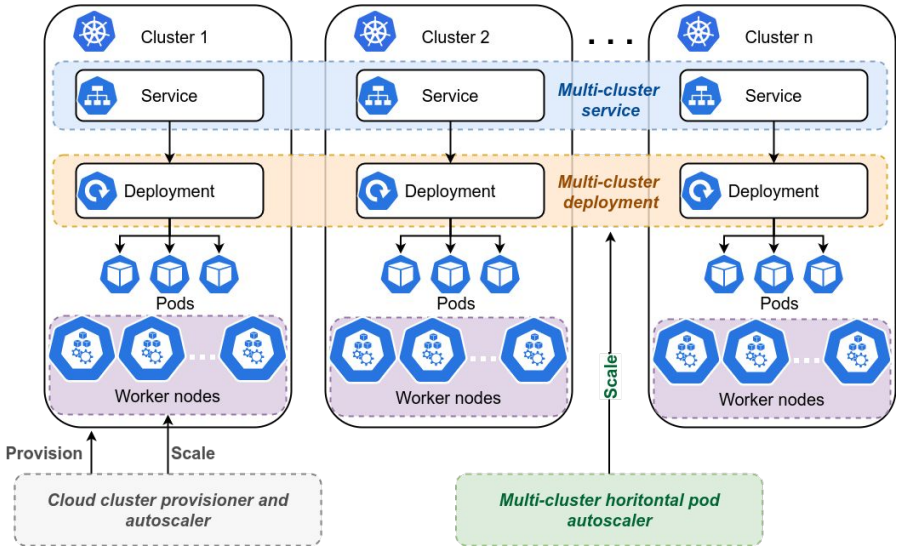


High-level KubeFed architecture with manifest files.

Multi-cluster Kubernetes (*mck8s*)

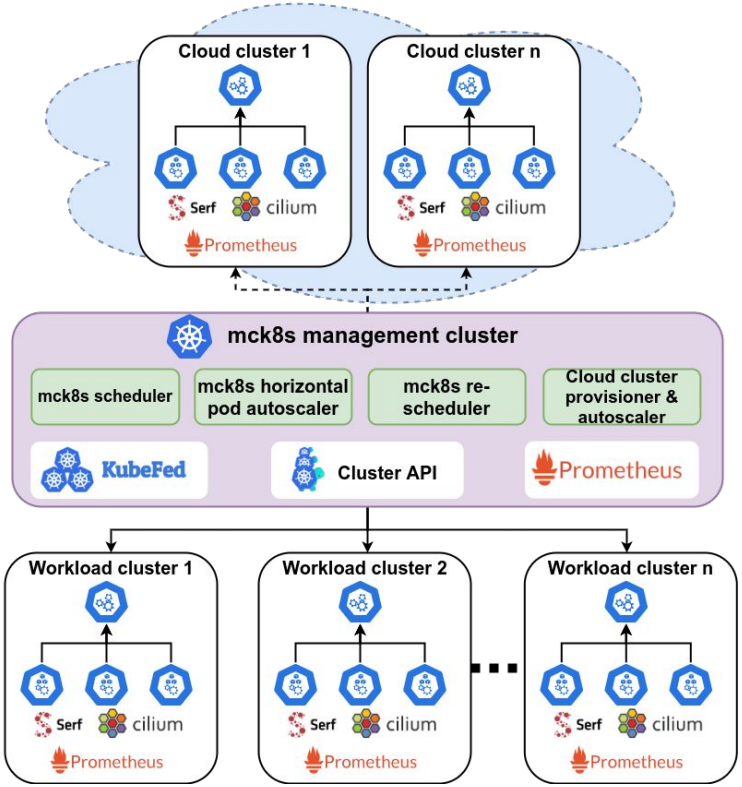
Goal:

- Provide automated placement, offloading and bursting
- Autoscaling at three levels:
  - Multi-cluster (federation)
  - Cluster (worker nodes)
  - Pods
- Inter-cluster network routing and load balancing



mck8s architecture

- One management cluster where applications are deployed
- Multiple workload clusters (potentially from different providers) that run applications
- Builds upon KubeFed, Cluster API, Prometheus, Serf and Cilium
- Four new controllers:
  - Multi-cluster scheduler
  - Multi-cluster horizontal pod autoscaler
  - Mck8s de-scheduler
  - Cloud cluster provisioner and autoscaler



mck8s custom resources

- Multi-cluster deployment (MCD)
- Multi-cluster job (MCJ)
- Multi-cluster service (MCS)
- Multi-cluster horizontal pod autoscaler (MCHPA)
- Cloud cluster provisioner and autoscaler
- Multi-cluster re-scheduler

```yaml
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: multiclusterdeployments.fogguru.eu
spec:
  scope: Namespaced
  group: fogguru.eu
  versions:
    - name: v1
      served: true
      storage: true
  names:
    kind: MultiClusterDeployment
    plural: multiclusterdeployments
    singular: multiclusterdeployment
    shortNames:
      - mcd
      - mcds
  versions:
    - name: v1
      served: true
      storage: true
      schema:
        openAPIV3Schema:
          type: object
          properties:
            spec:
              type: object
              x-kubernetes-preserve-unknown-fields: true
            status:
              type: object
              x-kubernetes-preserve-unknown-fields: true
```

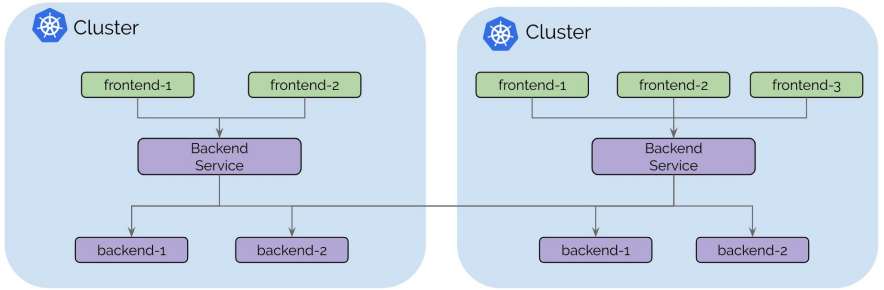
Multi-cluster scheduler

- Responsible for the lifecycle the following resources
  - MCD
  - MCS
  - MCJ
- Manual and policy-based automated scheduling / placement
  - Cluster-affinity: on selected clusters
  - Resource-based: worst-fit, best-fit
  - Network traffic-based: traffic-aware
- Horizontal offloading to neighboring clusters
- Bursting to neighboring clusters

```yaml
apiVersion: fogguru.eu/v1
kind: MultiClusterDeployment
metadata:
  name: hello
spec:
  numberOfLocations: 2
  placementPolicy: traffic-aware
  selector:
    matchLabels:
      app: hello
      tier: backend
      track: stable
  replicas: 2
  template:
    metadata:
      labels:
        app: hello
        tier: backend
        track: stable
    spec:
      containers:
        - name: hello
          image: "gcr.io/google-samples/hello-go-gke:1.0"
          resources:
            requests:
              memory: 1024Mi
              cpu: 1000m
            limits:
              memory: 1024Mi
              cpu: 1000m
          ports:
            - name: http
              containerPort: 80
```

Multi-cluster service and inter-cluster network routing

- Multi-cluster service creates services on the clusters containing the corresponding deployments
- Relies on Cilium cluster-mesh for inter-cluster network routing and load balancing




```yaml
apiVersion: fogguru.eu/v1
kind: MultiClusterService
metadata:
  name: hello
  annotations:
    io.cilium/global-service: "true"
spec:
  selector:
    app: hello
    tier: backend
  ports:
  - protocol: TCP
    port: 80
    targetPort: http
```

Multi-cluster horizontal pod autoscaler

- Control the scaling of MCDs from the management cluster
- Compute the number of desired replicas based on resource utilization
- Pass to the multi-cluster scheduler

```yaml
apiVersion: fogguru.eu/v1
kind: MultiClusterHorizontalPodAutoscaler
metadata:
  name: hello
spec:
  scaleTargetRef:
    apiVersion: fogguru.eu/v1
    kind: MultiClusterDeployment
    name: hello
  minReplicas: 2
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
```

Cloud cluster provisioner and autoscaler

- Periodically checks of status of MCDs
- Provision a K8S cluster in the cloud provider of choice via Cluster API
- Join to the federation
- Scale-out or -in the cluster nodes as necessary
- Remove the cloud cluster is not needed anymore

```
apiVersion: fogguru.eu/v1
kind: CloudProvisioner
metadata:
  name: cp1
spec:
  cloudClusterName: cloud1
  gatewayIP: 10.16.91.27
  floatingIP: 10.16.92.1
  extNetworkID: c95efff4-fd35-46f1-af1d-d65459fcebef
  securityGroupID: 9cc0b5b7-222f-41bb-8960-98f5cce14
  cloudsYaml:
Y2xvdWRzOgogIG9wZW5zdGFjazoKICAgIGF1dGg6CiAgICAgIGF1
  certText:
W0dsb2JhbF0KYXV0aC11cmw9aHR0cDovLzEwLjE2LjYxLjI1NTo1
```

Implementation

- Implemented in Python
- Using Kopf (Kubernetes Operators Framework)

multiclusterscheduler.py

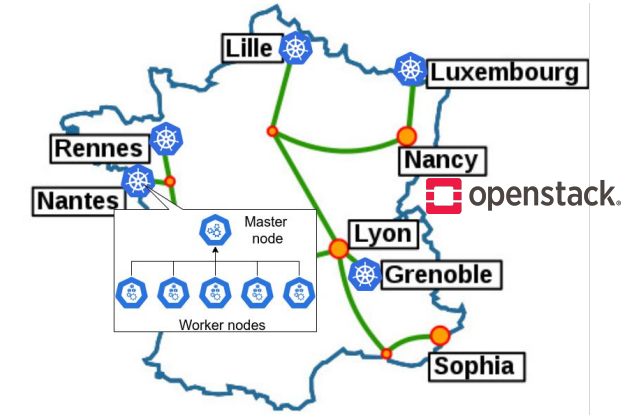
multi-cluster-scheduler > multiclusterscheduler.py > ...

```python
import kopf
import yaml, pandas as pd
from utils import findPossibleClusters, getFogAppLocations,  getCloudCluster, \
    createDeployment, createService, deleteDeployment, deleteService, patchDeployment, patchService, createJob, \
    deleteJob, patchJob, getMaximumReplicas, findNearestClusters, getAllocatableCapacity, getFogAppClusters, getServiceClusters
import json
import time

# Create multi-cluster deployment
@kopf.on.create('fogguru.eu', 'v1', 'multiclusterdeployments')
def create_fn(body, spec, patch, **kwargs):
    # Get info from multiclusterdeployments object
    fogapp_name = body['metadata']['name']
    fogapp_image = spec['template']['spec']['containers'][0]['image']
    fogapp_replicas = spec['replicas']
    fogapp_cpu_request = int(spec['template']['spec']['containers'][0]['resources']['requests']['cpu'][:-1])
    #fogapp_cpu_limit = spec['template']['spec']['containers']['resources']['limits']['cpu']
    fogapp_memory_request = int(spec['template']['spec']['containers'][0]['resources']['requests']['memory'][:-2])
    #fogapp_memory_limit = spec['template']['spec']['containers']['resources']['limits']['memory']
    #fogapp_type = spec['appType']
    #fogapp_type = body['kind']
    spec_text = str(spec)

    # Make sure image is provided
    if not fogapp_image:
        raise kopf.HandlerFatalError(f"Image must be set. Got {fogapp_image}.")

    if not fogapp_replicas:
        raise kopf.HandlerFatalError(f"Number of replicas must be set. Got {fogapp_replicas}.")

    # Get namespace
    if 'namespace' in body['metadata']:
        fogpapp_namespace = body['metadata']['namespace']
    else:
        fogpapp_namespace = "default"
```

Demo

Pre-requisites

- A management cluster and few workload clusters (K8S)
- KubeFed, Prometheus, Cluster API on management cluster
- Workload clusters with Cilium and Cilium cluster mesh, Serf, Prometheus
- Credentials for a cloud provider (OpenStack, AWS, GCP, etc.)
- Physical / virtual network between clusters (eg. VPN)



Testbed containing a management cluster (Rennes) and 5 workload clusters (Rennes, Nantes, Lille, Grenoble, Luxembourg). OpenStack cloud cluster in Nancy.

Each of the five clusters has a master node and 5 worker nodes

Clusters 1 & 5
- 4 CPU cores
- 16 GB RAM

Clusters 2, 3, 4
- 2 CPU cores
- 4 GB RAM

	Rennes	Nantes	Lille	Luxembourg	Nancy	Grenoble
Rennes	-	2.16	23.26	27.41	25.18	17.45
Nantes	2.16	-	22.21	26.29	24.16	16.38
Lille	23.26	22.21	-	11.88	9.70	12.06
Luxembourg	27.41	26.29	11.88	-	2.90	15.33
Nancy	25.18	24.16	9.70	2.90	-	13.14
Grenoble	17.45	16.38	12.06	15.33	13.14	-

Inter-site network latency (RTT) in milliseconds

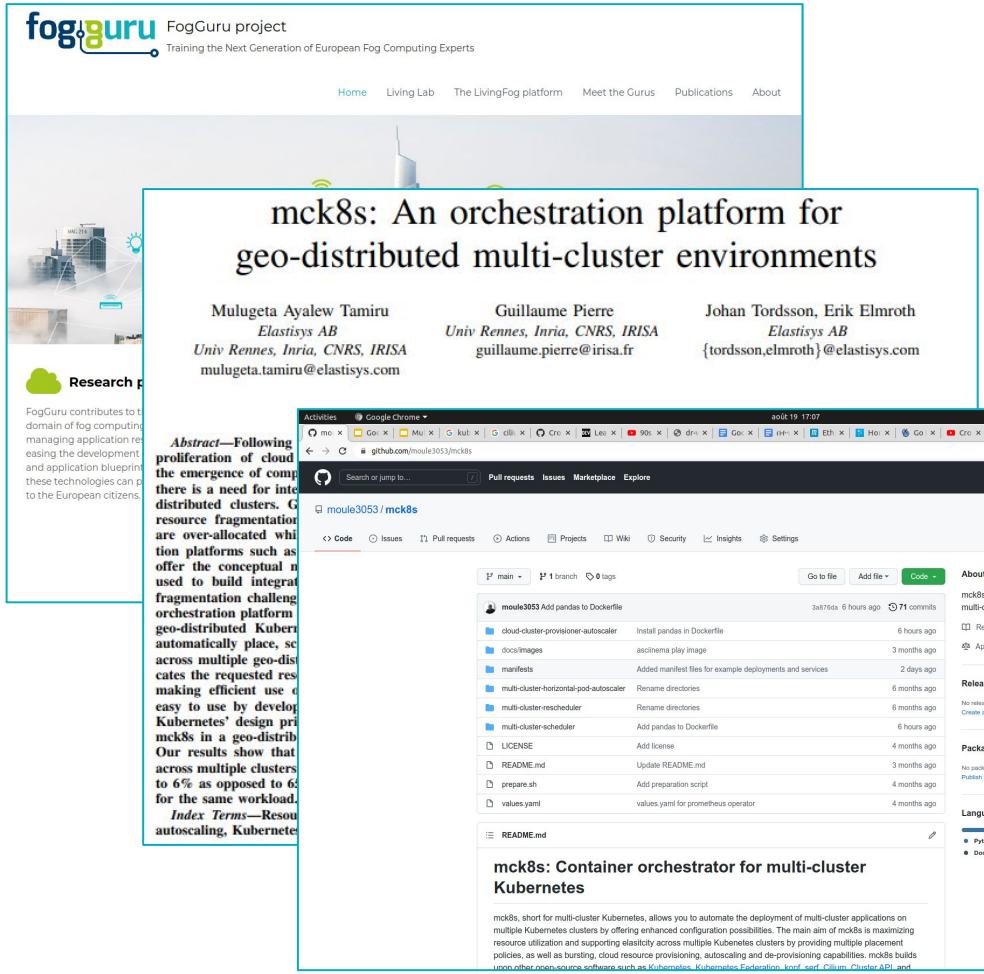
Demo

Learn more & contribute

FogGuru project website:
http://www.fogguru.eu

Related paper (accepted at 30th International Conference on Computer Communications and Networks (ICCCN 2021)):
https://hal.inria.fr/hal-03205743

Github:
https://github.com/moule3053/mck8s



fog guru

TRAINING THE NEXT GENERATION OF EUROPEAN FOG COMPUTING EXPERTS

The FogGuru project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant 765452.

European Commission