

DevOps

Cours 5: Kubernetes

Antoine Balliet

antoine.balliet@dauphine.psl.eu

Précédemment ...

Orchestration de conteneurs

- Concepts

- Logique de cycle de vie & environnement du conteneur
- Déploiement de plusieurs conteneurs / services
- Déclaratif -> format YAML
 - docker-compose.yml
 - (k8s) deployment.yaml
 - (k8s) service.yaml

```
services:
  web:
    build: .
    ports:
      - "8000:5000"
  redis:
    image: "redis:alpine"
```

- Technologies

- Docker Compose
 - 1 seul noeud
- Kubernetes
 - système distribué multi-noeud
 - Master Node
 - Worker Node(s)

Précédemment ...

TP

- Docker Compose
 - serveur web python
 - redis: système de stockage clé / valeur

```
services:
  web:
    build: .
    ports:
      - "8000:5000"
  redis:
    image: "redis:alpine"
```

2 services

- 1 local que l'on build
- 1 direct du hub

- Exemple d'amélioration de la résilience d'un système distribué

```
def get_hit_count():
    retries = 5
    while True:
        try:
            return cache.incr('hits')
        except redis.exceptions.ConnectionError as exc:
            if retries == 0:
                raise exc
            retries -= 1
            time.sleep(0.5)
```

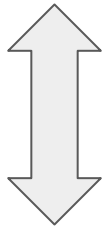
Incrément de la clé
"hits", on réessaie 5
fois si échoué

Précédemment ...

TP

- Communication entre les 2 services

```
(base) antoineballiet@macbook-pro-1 ~  
$ docker run --rm redis:alpine  
1:C 27 Jan 2025 00:11:43.158 * oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo  
1:C 27 Jan 2025 00:11:43.158 * Redis version=7.4.2, bits=64, commit=00000000, modified=0, p  
1:C 27 Jan 2025 00:11:43.158 # Warning: no config file specified, using the default config.  
1:M 27 Jan 2025 00:11:43.158 * monotonic clock: POSIX clock_gettime  
1:M 27 Jan 2025 00:11:43.159 * Running mode=standalone, port=6379.  
1:M 27 Jan 2025 00:11:43.159 * Server initialized  
1:M 27 Jan 2025 00:11:43.159 * Ready to accept connections tcp
```



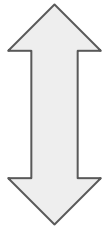
```
import redis  
from flask import Flask  
  
app = Flask(__name__)  
cache = redis.Redis(host='redis', port=6379)
```

Précédemment ...

TP

- Communication entre les 2 services

```
(base) antoineballiet@macbook-pro-1 ~  
$ docker run --rm redis:alpine  
1:C 27 Jan 2025 00:11:43.158 * oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo  
1:C 27 Jan 2025 00:11:43.158 * Redis version=7.4.2, bits=64, commit=00000000, modified=0, p  
1:C 27 Jan 2025 00:11:43.158 # Warning: no config file specified, using the default config.  
1:M 27 Jan 2025 00:11:43.158 * monotonic clock: POSIX clock_gettime  
1:M 27 Jan 2025 00:11:43.159 * Running mode=standalone, port=6379.  
1:M 27 Jan 2025 00:11:43.159 * Server initialized  
1:M 27 Jan 2025 00:11:43.159 * Ready to accept connections tcp
```



Pas de port
binding 🤔

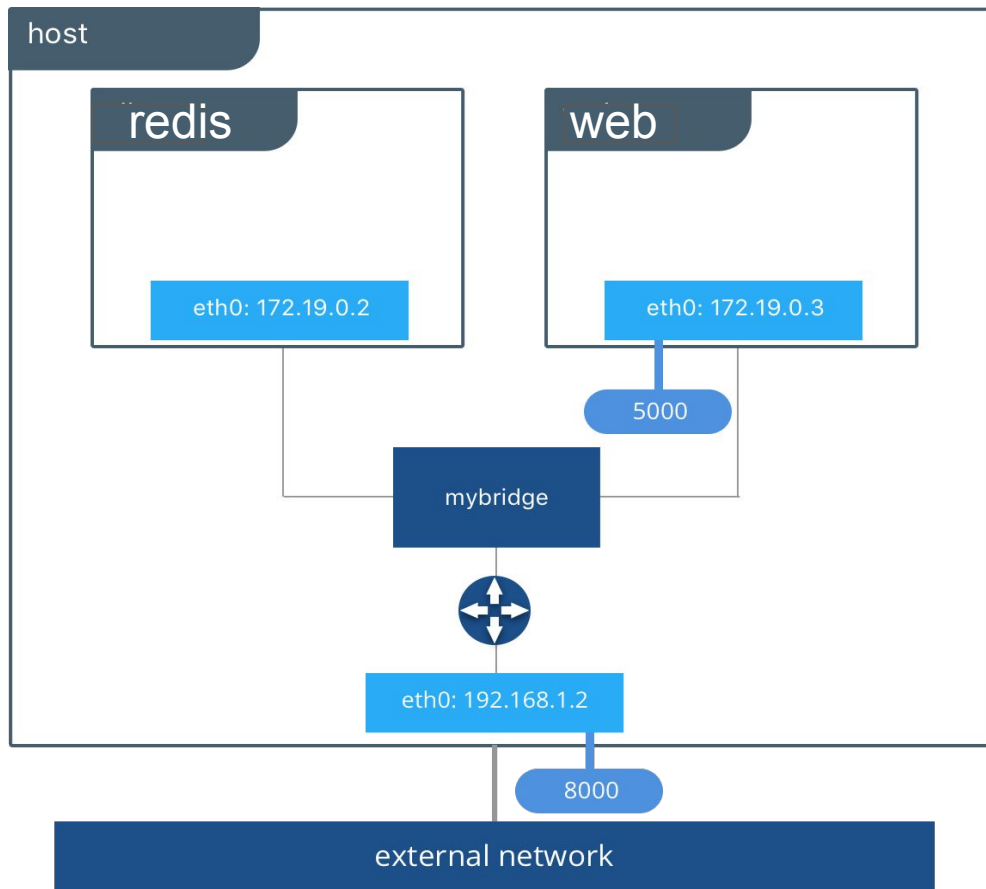
```
import redis  
from flask import Flask  
  
app = Flask(__name__)  
cache = redis.Redis(host='redis', port=6379)
```

```
services:  
  web:  
    build: .  
    ports:  
      - "8000:5000"  
  redis:  
    image: "redis:alpine"
```

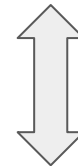
Précédemment ...

TP

- Communication avec interface bridge



```
services:  
  web:  
    build: .  
    ports:  
      - "8000:5000"  
  redis:  
    image: "redis:alpine"
```



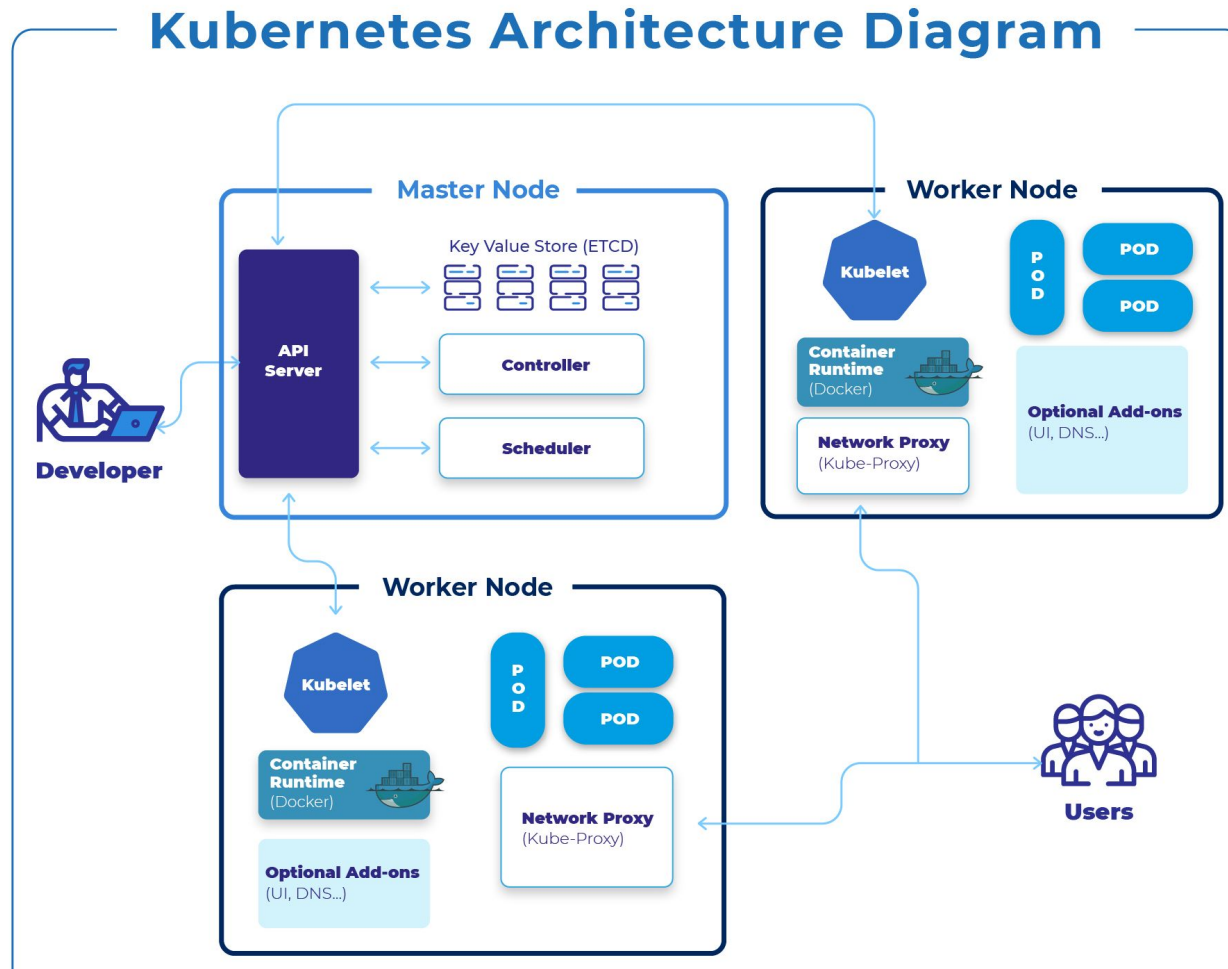
```
networks:  
  custom_network:  
    driver: bridge  
services:  
  web:  
    build: .  
    networks:  
      - custom_network  
    ports:  
      - "8000:5000"  
  redis:  
    networks:  
      - custom_network  
    image: "redis:alpine"
```

Kubernetes

- Les limites de la gestion isolée
 - Déploiements automatisés
 - équilibrage des ressources sur des noeuds mutualisés
 - dépendances entre containers
 - Gestion des applications
 - reprise après incident
 - garantir la disponibilité
 - Scalabilité
 - Charge d'utilisation variable
- Kubernetes - créé par Google, maintenu par la Cloud Native Computing Foundation

Précédemment ...

- Kubernetes - Système orchestration multi-noeud



Kubernetes

- Kubernetes - Système orchestration multi-noeud
 - Déploiements automatisés
 - Déploiement déclaratif
 - Automatisation des tâches répétitives
 - Rolling updates
 - Rollbacks
 - Gestion des applications
 - Auto-réparation
 - Planification intelligente
 - Gestion des ressources
 - Scalabilité
 - Scaling horizontal automatique
 - Scaling des nœuds (avec cloud provider)

Kubernetes

- Création de cluster
 - Déploiement manuel
 - Provisionnement de machines
 - Déploiement master node / worker nodes
 - Service managé : Cloud Provider
 - GKE sur GCP
 - EKS sur AWS
 - AKS sur Azure

Quelques avantages

- Maintenance automatisée
- Provisionnement rapide
- Pas de gestion du plan de contrôle (API server, scheduler, etcd)
- SLA
- Autoscaling natif

Précédemment ...

TP

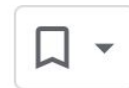
Creating a GKE cluster

A GKE cluster is a managed set of Compute Engine virtual machines that operate as a single GKE cluster.

1. Create the cluster.

```
gcloud container clusters create-auto helloworld-gke \  
  --location us-central1
```

gcloud container clusters create-auto

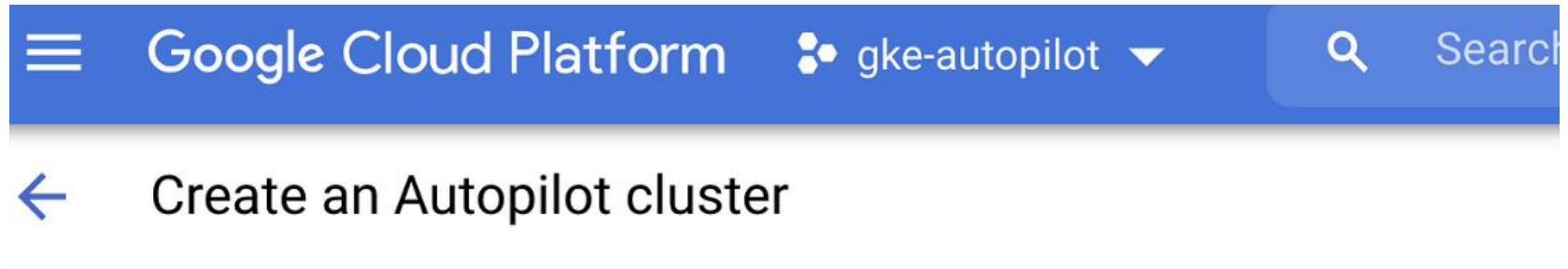


NAME

gcloud container clusters create-auto - create an Autopilot cluster for running containers

Précédemment ...

TP



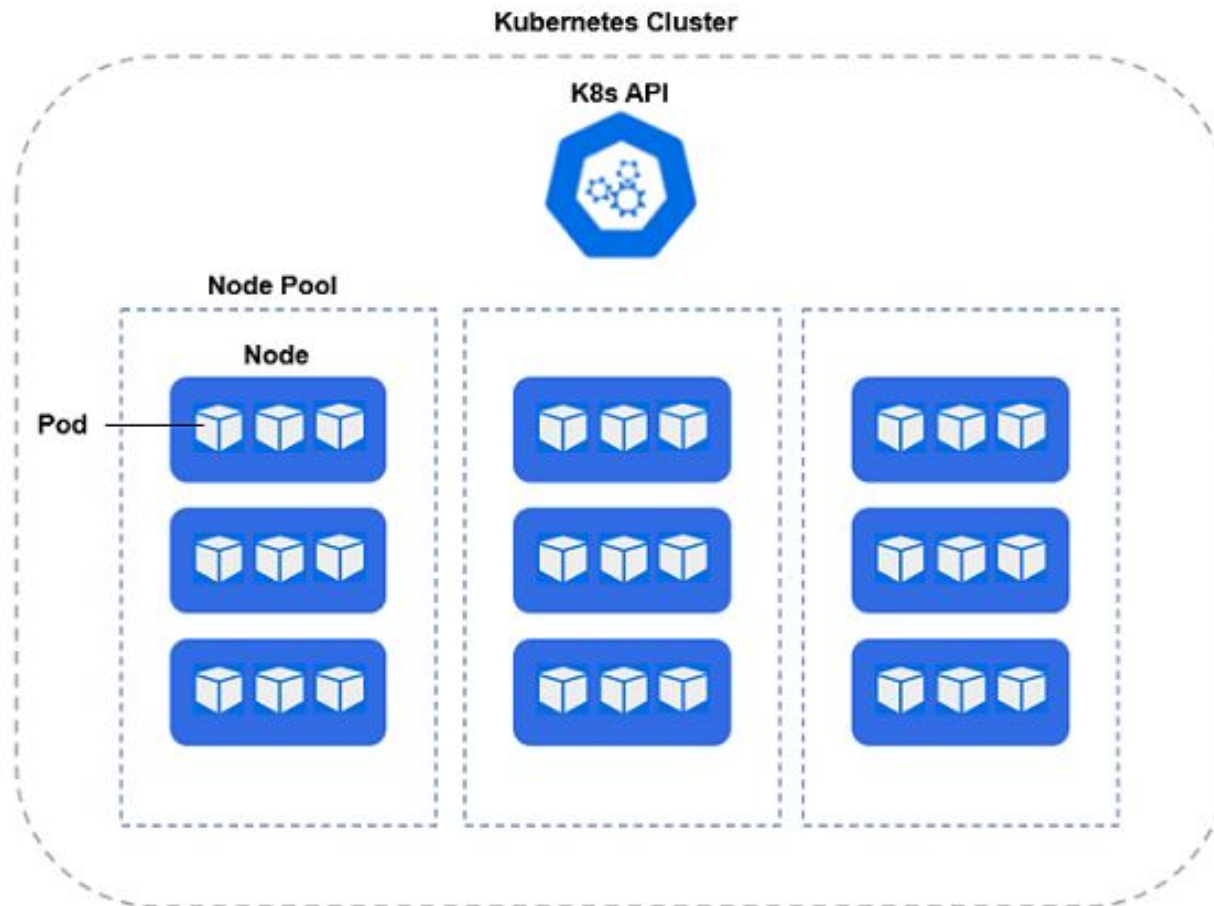
Create an Autopilot cluster by specifying a name and region. After the cluster is created, you can deploy your workload through Kubernetes and we'll take care of the rest, including:

- ✓ **Nodes:** Automated node provisioning, scaling, and maintenance
- ✓ **Networking:** VPC-native traffic routing for public or private clusters
- ✓ **Security:** Shielded GKE Nodes and Workload Identity
- ✓ **Telemetry:** Operations logging and monitoring

--

Précédemment ...

TP



Node = machine

Node Pool = ensemble de machines ayant le même type

Pod = plus petite unité de kubernetes - ensemble de container - partage même réseau, machine, contexte d'exécution

Kubernetes & Terraform

- Kubernetes expose une API => Un [terraform provider existe](#) !



kubernetes



Official

by:



[HashiCorp](#)

Container Orchestration

- Créer et versionner les ressources kubernetes

Ressources Kubernetes (principales)

- Ressources de calcul
 - Deployment
 - ReplicaSet
 - Job / CronJob
- Ressources de configuration
 - ConfigMap
 - Secret
 - Service Account
- Ressources réseau
 - Service
 - Ingress
- Ressources de stockage
 - PersistentVolume
 - PersistentVolumeClaim

Précédemment ...

TP

Définitions des ressources en YAML

Déploiement:

https://cloud.google.com/kubernetes-engine/docs/archive/deploy-app-container-image#deploy_an_app

Service:

https://cloud.google.com/kubernetes-engine/docs/archive/deploy-app-container-image#deploy_a_service

Déploiement -> spec pour créer des pods

Service -> Expose un endpoint fixe réseau pour accéder aux pods. Ici avec un LoadBalancer (équilibreur de charge)

Précédemment ...

TP

Inspection des ressources avec le CLI **kubectl**

2. Deploy the resource to the cluster:

```
kubectl apply -f deployment.yaml
```

3. Track the status of the Deployment:

```
kubectl get deployments
```

The Deployment is complete when all of the **AVAILABLE** deployments are **READY**.

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
helloworld-gke	1/1	1	1	20s

Précédemment ...

TP

Inspection des ressources avec le CLI **kubectl**

2. Create the Hello World Service:

```
kubectl apply -f service.yaml
```

3. Get the external IP address of the service:

```
kubectl get services
```

It can take up to 60 seconds to allocate the IP address. The external IP address is listed under the column **EXTERNAL-IP** for the **hello** Service.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
hello	LoadBalancer	10.22.222.222	35.111.111.11	80:32341/TCP	1m
kubernetes	ClusterIP	10.22.222.1	<none>	443/TCP	20m

Critique du TP

Des fichiers YAML sans lien entre eux 🤪

Quels problèmes avons-nous constaté
pendant le TP Kubernetes ?

Simplification ? 😞 -
Beaucoup de code à
copier ...

Passage de variables ? 🔍

Versioning et
collaboration ? 🤪

Helm



Helm: Le gestionnaire de paquets pour Kubernetes

- définir les ressources nécessaire à une application
- partager l'ensemble appelé "helm chart"
 - Artifact Hub <https://artifacthub.io/>
 - Private registry (i.e. Artifact Registry)
 - Réutiliser un "chart" avec des paramètres uniques

Helm

- Chart

Ensemble de ressources pour créer une instance d'une application Kubernetes

- Config


Configuration qui peut être fusionnées dans un “chart” pour créer une “release”, un déploiement de l'application complète

- Release

Une release est une instance en cours d'exécution d'un chart combiné à une config

Helm

examples / charts / hello-world / templates / service.yaml


 **adamreese** Lets do this


Code Blame 15 lines (15 loc) · 373 Bytes


```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: {{ include "hello-world.fullname" . }}
5    labels:
6      {{- include "hello-world.labels" . | nindent 4 }}
7  spec:
8    type: {{ .Values.service.type }}
9    ports:
10     - port: {{ .Values.service.port }}
11       targetPort: http
12       protocol: TCP
13       name: http
14    selector:
15      {{- include "hello-world.selectorLabels" . | nindent 4 }}
```



<https://github.com/helm/examples/tree/main/charts/hello-world>

Helm





**nginx**




 **Helm chart**


 **Networking**

 Bitnami  Bitnami

NGINX Open Source is a web server that can be also used as a reverse proxy, load balancer, and HTTP cache. Recommended for high-demand content.



 **SUBSCRIPTIONS: 23**  **WEBHOOKS: 3**  **PRODUCTION USERS: 2**

 **nginx**

Bitnami package for NGINX Open Source

NGINX Open Source is a web server that can be also used as a reverse proxy, load balancer, and HTTP cache. Recommended for high-demanding sites due to its ability to provide faster content.

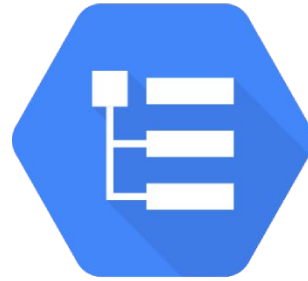
Helm

- Documentation

NGINX deployment parameters

Name	Description	Value
replicaCount	Number of NGINX replicas to deploy	1
revisionHistoryLimit	The number of old history to retain to allow rollback	10
updateStrategy.type	NGINX deployment strategy type	RollingUpdate
updateStrategy.rollingUpdate	NGINX deployment rolling update configuration parameters	{}
podLabels	Additional labels for NGINX pods	{}
podAnnotations	Annotations for NGINX pods	{}
podAffinityPreset	Pod affinity preset. Ignored if affinity is set. Allowed values: soft or hard	""
podAntiAffinityPreset	Pod anti-affinity preset. Ignored if affinity is set. Allowed values: soft or hard	soft
nodeAffinityPreset.type	Node affinity preset type. Ignored if affinity is set. Allowed values: soft or hard	""
nodeAffinityPreset.key	Node label key to match Ignored if affinity is set.	""
nodeAffinityPreset.values	Node label values to match. Ignored if affinity is set.	[]
affinity	Affinity for pod assignment	{}
hostNetwork	Specify if host network should be enabled for NGINX pod	false

APM: Métriques / Logging / Alerting



APM: Application Performance Monitoring

- ingérer les métriques et logs des systèmes
- créer des indicateurs / tableaux de bord
- visibilité / trace entre les systèmes
- alerting
 - routage des erreurs par équipe
 - contexte de l'erreur

TP

- Finir le TP 4 Partie 2 Kubernetes
 - Premières ressources kubernetes
 - N'hésitez pas à me poser des questions
- TP 5
 - Helm chart
 - CLI helm
 - Terraform provider helm
 - Déploiement d'un orchestrateur de pipeline déclaratif (Kestra)
 - Docker / Docker compose
 - Kubernetes

Pour aller plus loin

- Open Telemetry
 - <https://opentelemetry.io/>
- Signoz (Datadog mais Open Source)
 - <https://signoz.io/>
- Aller plus loin sur Kubernetes
 - <https://karpenter.sh/>