etna
L'ÉCOLE DES TECHNOLOGIES NUMÉRIQUES AVANCÉES

ISR-CLO4
Déploiement d'applications dans Kubernetes

etna
L'ÉCOLE DES TECHNOLOGIES NUMÉRIQUES AVANCÉES

CLO4

Introduction

Pour réaliser le projet une instance K3S est mise à votre disposition.

C’est la distribution allégée de Rancher. Tous les composants des fournisseurs de cloud (AWS, Azure et GCP) ont été retirés.

K3S est tout indiqué pour une installation sur une infrastructure bare-metal. Il fonctionne sur les architectures x86 et ARM.

K3S est contenu dans un seul binaire.

https://rancher.com/docs/k3s/latest/en/

etna
L'ÉCOLE DES TECHNOLOGIES NUMÉRIQUES AVANCÉES

Sur K3S, et plus généralement sur Kubernetes (ou K8S), un répartiteur et un ‘ingress controller’ sont nécessaires pour exposer les services à l’extérieur, et permettre ainsi leur accès depuis Internet ou tout autre réseau externe au cluster.

En fonction des distribution Kubernetes, il existent différents types de répartiteur de charge. Il est également possible d’ étendre les fonctionnalités de K8S ou de les remplacer par un composant tierce.

Pour ce projet, Traefik 2.1 est configuré en tant qu’ingress controller. C’est lui qui va gérer l’exposition de vos services HTTP et TCP, et la répartition de charge.

https://docs.traefik.io/v2.1/

Traefik intègre une fonctionnalité de découverte de services. Tous les labels définis dans les déploiements Kubernetes remontent dans Traefik, ainsi que toutes les méta-données générées par Kubernetes.

Cela permet à Traefik d'enregistrer automatiquement tout service en fonction de son port, son nom de domaine, etc.

Avec l'ingress controller Traefik il suffit de définir une ingress route afin de spécifier le domaine, l'URL et le port ou joindre un service.

Ainsi toutes les requêtes arrivant sur l'adresse IP externe de l'instance K3S (l'adresse IP de votre machine) sont filtrées et routées par Traefik.

Il est inutile d'exposer les ports d'écoute des pods sur l'interface de la machine hôte, le composant 'kube-proxy' se charge de lier dynamiquement, et automatiquement, un port de la machine hôte vers le pod.

etna
L'ÉCOLE DES TECHNOLOGIES NUMÉRIQUES AVANCÉES

CLO4

Labels & Sélecteurs

Les labels permettent d'ajouter des informations permettant de repérer des objets dans le cluster. Ces informations pourront ensuite être utilisées pour effectuer des sélections d'objets, elles sont collectées automatiquement par la découverte de service.

Les labels s'écrivent sous la forme d'une clé associée à une valeur :

- environment : production
- release : stable
- appName: myApp

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: label-demo
  labels:
    environment: production
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:

    - containerPort: 80
```

etna
L'ÉCOLE DES TECHNOLOGIES NUMÉRIQUES AVANCÉES

Les labels sont notamment utilisés par les sélecteurs, qui permettent de sélectionner des objets parmis un ensemble d'objets. Cela permet d'associer une ressource à une autre.
Par exemple un service pour une application.

Il est possible d'appliquer des opérateurs sur les sélecteurs :

- environment = production
- environment in (production, qa)
- tier != frontend

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80

      targetPort: 9376
```

etna
L'ÉCOLE DES TECHNOLOGIES NUMÉRIQUES AVANCÉES

CLO4

Les déploiements

etna
L'ÉCOLE DES TECHNOLOGIES NUMÉRIQUES AVANCÉES

Les déploiements sont des objets qui décrivent tous les paramètres de déploiement d'un pod. Ils sont écrits en YAML.

Les informations minimales requises contenues dans un déploiement :

- **apiVersion:** Version de l'API Kubernetes à utiliser
- **kind**: type d'objet
- **metadata**: définit les méta-données d'un objet (nom, labels et annotations)
- **spec**: caractéristiques de l'objet
  - **replicas**: nombre de pods à déployer
  - **template**: template du pod à créer
  - **containers**: nom, image et port du, ou des, conteneur(s)

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
          - containerPort: 80
```

etna
L'ÉCOLE DES TECHNOLOGIES NUMÉRIQUES AVANCÉES

Stratégies de mise à jour

Une fois un déploiement réalisé, une application subira maintes mises à jour. Pour gérer ces mises à jour sans coupure de service, il existe plusieurs méthodes parmis les suivantes :

- Rolling update
- Blue green
- Canary deployments

Les méthodes **blue green** et **canary deployments** font partie du concept de “Zero Downtime Deployment”.
La méthode de **Rolling update** est un concept de mise à jour en continu.

La méthode **Canary deployment** nécessite l’installation d’un **Ingress controller** ayant cette fonctionnalité.

etna
L'ÉCOLE DES TECHNOLOGIES NUMÉRIQUES AVANCÉES

CLO4

Service

etna
L'ÉCOLE DES TECHNOLOGIES NUMÉRIQUES AVANCÉES

Un service Kubernetes est un objet qui permet d'abstraire l'exposition réseau d'une application sur un ensemble de Pods. On parle d'un service réseau.
Un service dispose de plusieurs fonctionnalités telle que la découverte de services, un résolveur DNS et un répartiteur de charge.

Il existe plusieurs types de services.

etna
L'ÉCOLE DES TECHNOLOGIES NUMÉRIQUES AVANCÉES

ClusterIP

C'est le type de service par défaut. Une adresse IP virtuelle est générée par Kubernetes afin d'accéder au service depuis les autres pods.

L'adresse IP provisionnée par Kubernetes n'est seulement disponible qu'à l'intérieur du cluster.

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80

       targetPort: 9376
```

etna
L'ÉCOLE DES TECHNOLOGIES NUMÉRIQUES AVANCÉES

LoadBalancer

Ce type de service utilise le load balancer de l'infrastructure sous-jacente pour exposer les services (GCP, AWS, Azure, OpenStack, VmWare, etc.).

L'adresse IP provisionnée est externe au cluster et joignable depuis l'extérieur du cluster.

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
  clusterIP: 10.0.171.239
  type: LoadBalancer
status:
  loadBalancer:
    ingress:

      - ip: 192.0.2.127
```

NodePort

Ce type de service expose un port spécifique sur chaque noeud du cluster, joignable depuis l'extérieur du cluster via l'adresse IP du noeud et le port spécifié.

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  type: NodePort
selector:
  app: nginx
ports:
 -  nodePort: 42021
    port: 80
    targetPort: 80
```

HeadLess

Ce type de service n'utilise pas de cluster IP, ni IP externe ni de répartiteur de charge.
La résolution DNS interne à Kubernetes renverra directement l'adresse IP de chaque pod.

Ce type de service est utile si un répartiteur de charge n'est pas nécessaire, et pour des Ingress Controller tel que Traefik.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: whoami

spec:
  ports:
    - protocol: TCP
      name: web
      port: 80
  selector:
    app: whoami
```

etna
L'ÉCOLE DES TECHNOLOGIES NUMÉRIQUES AVANCÉES

CLO4

Ingress Traefik

etna
L'ÉCOLE DES TECHNOLOGIES NUMÉRIQUES AVANCÉES

Ingress natif

La ressource **Ingress** permet de configurer un répartiteur de charge ou un proxy, en entrée du cluster.
De base elle se définit par son type de ressource (kind) et par des paramètres de règles, de protocoles,, chemins et backend.

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: test-ingress
  annotations:

nginx.ingress.kubernetes.io/rewrite-t
arget: /
spec:
  rules:
  - http:
      paths:
      - path: /testpath
        backend:
          serviceName: test

            servicePort: 80
```

https://kubernetes.io/fr/docs/concepts/services-networking/ingress/

Ingress Traefik

Après avoir configuré l'**ingress controller Traefik**, la définition d'un ingress adopte les paramètres propres à Traefik.

Traefik gère le routage des requêtes HTTP en fonction du domaine et éventuellement du préfix du chemin d'URL.

Note: l'objet IngressRoute, tout comme tout type d'Ingress requiert la définition d'un service.

```yaml
apiVersion: traefik.containo.us/v1alpha1
kind: IngressRoute
metadata:
  name: simpleingressroute
  namespace: default
spec:
  entryPoints:
    - web
  routes:
  - match: Host(`your.domain.com`) &&
PathPrefix(`/notls`)
    kind: Rule
    services:
    - name: whoami
      port: 80
```

https://docs.traefik.io/v2.1/user-guides/crd-acme/

etna
L'ÉCOLE DES TECHNOLOGIES NUMÉRIQUES AVANCÉES

CLO4

Persistent Volumes & Storage Classes

Un **PersistentVolume** est une couche d'abstraction qui expose une API permettant de gérer le stockage alloué au cluster. Cette ressource implémente l'utilisation du stockage par les pods, configuré au préalable. Ce stockage peut être sur le système de fichier local, en réseau via NFS, Ceph, GlusterFS, S3, et bien d'autres.

Un **PersistentVolumeClaim** est une demande de stockage qui va consommer les ressources **PersistentVolume.** Cette demande est définie par une taille et un type d'accès (en écriture, lecture ou lecture seule).

https://kubernetes.io/fr/docs/concepts/storage/persistent-volumes/

Un objet **PersistentVolume** permet d'allouer un espace de stockage afin de conserver les données, même après redémarrage d'un pod l'utilisant, ou même après suppression si la configuration le permet. On parle alors d'un service **Stateful.** A la différence d'un service **Stateless** qui ne conserve pas les données qu'il produit.

Cela est tout indiqué pour les services de base de données, ou pour le partage de données au sein du cluster.

Une **StorageClass** est une ressource qui définit le type de stockage d'un **PersistentVolume**, et la façon dont il doit être consommé.

Les paramètres essentiels :

- **provisioner** : Le plugin utilisé pour fournir le stockage : AWS EBS, Ceph, GlusterFS, etc.
- **parameters** : les classes de stockage du fournisseur (GP2 par exemple pour AWS).
- **reclaimPolicy** : Définit le cycle de vie du volume de stockage. Peut avoir les valeurs **retain** ou **delete**
  - **retain** : le volume est conservé même après la suppression du pod attaché.
  - **delete** : le volume est supprimé avec la suppression du pod attaché.

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard
provisioner: kubernetes.io/aws-ebs
parameters:
  type: gp2
reclaimPolicy: Retain
allowVolumeExpansion: true
mountOptions:
  - debug

volumeBindingMode: Immediate
```

https://kubernetes.io/docs/concepts/storage/storage-classes/

etna
L'ÉCOLE DES TECHNOLOGIES NUMÉRIQUES AVANCÉES

CLO4

Les secrets dans Kubernetes

etna
L'ÉCOLE DES TECHNOLOGIES NUMÉRIQUES AVANCÉES

Un objet **secret** permet de stocker et gérer les accès aux informations sensibles, telles que les mots de passe, jetons d'authentification et clés SSH.

Les secrets sont stockés de manière distribuée et chiffrée. Par défaut le chiffrement est faible (base64).

L'accès aux secrets est possible directement ou via des variables d'environnement de pod, ou en tant que volume. A l'instar des **ConfigMaps.**

Un objet secret est accessible par tous les noeuds d'un cluster d'un même namespace, il n'y pas de restrictions d'accès du moment que l'utilisateur connaît le nom et le chemin d'un secret.

C'est pourquoi en environnement de production, il est préférable de confier la gestion des secrets à une application dédiée, telle que Vault. Cette dernière ajoute des listes de contrôles d'accès, un chiffrement fort, la rotation automatique des secrets, et bien d'autres fonctionnalités.

https://kubernetes.io/fr/docs/concepts/configuration/secret/

etna
L'ÉCOLE DES TECHNOLOGIES NUMÉRIQUES AVANCÉES

CLO4

Commandes utiles

etna
L'ÉCOLE DES TECHNOLOGIES NUMÉRIQUES AVANCÉES

Pour interagir avec les APIs Kubernetes, on utilise pour commencer l'outil en ligne de commande **kubectl**.
Après avoir installé **kubectl** il est vivement recommandé d'activer l'auto-complétion.

Exemples de base :

Application d'une configuration :
`kubectl apply -f fichier.yaml`

Suppression d'une configuration :
`kubectl delete -f fichier.yaml`

Les commandes peuvent s'appliquer à plusieurs fichiers au sein d'un répertoire :
`kubectl apply -f myApp/`
Y compris le répertoire courant :
`kubectl apply -f .`

https://kubernetes.io/fr/docs/reference/kubectl/overview/

etna
L'ÉCOLE DES TECHNOLOGIES NUMÉRIQUES AVANCÉES

Pour aller plus loin il est possible de construire des paquetages d'application Kubernetes et de les déployer comme un gestionnaire de paquets.
L'outil de paquetages Kubernetes le plus utilisé est Helm.
Il permet de déployer également un ensemble d'applications, avec leur dépendances respectives.

https://helm.sh

Gautier Loterman

N’hésitez pas à poser vos questions via le post FAQ