Documentation Technique: Migration Dynamique des Services entre le Cloud et l'Edge

Racem MOALLA

August 9, 2024

Contents

1 Préparation de l'Infrastructure

Pour déployer notre solution de migration dynamique des services entre le cloud et le edge, nous allons créer une infrastructure distribuée sur Google Cloud Platform (GCP). Cette infrastructure inclura deux clusters Kubernetes, une machine virtuelle pour simuler la voiture autonome "Picar-X" et une machine virtuelle pour exécuter Jenkins.

1.1 Étapes de Préparation de l'Infrastructure

1. **Créer deux clusters Kubernetes** :
   - **Cluster Europe** : Pour gérer les services à faible latence, proche de la VM simulant la voiture autonome.
   - **Cluster North America** : Pour gérer les services nécessitant des capacités de calcul élevées.
2. **Créer une VM "Platoon" en Europe** :
   - Cette VM simulera la voiture autonome en collectant des données et en les transmettant au cluster Europe pour prétraitement.
3. **Créer une VM pour Jenkins** :
   - Cette VM sera utilisée pour orchestrer les migrations dynamiques des services entre les clusters Europe et North America.

1.2 Prérequis

- Compte Google Cloud avec les permissions nécessaires pour créer des clusters Kubernetes et des machines virtuelles.
- Google Cloud SDK installé et configuré sur votre machine locale.

2 Configurer la VM Jenkins

1. **Connectez-vous à la VM** :

```
gcloud compute ssh jenkins-vm --zone europe-west1-b
```

2. **Installez Docker** :

```
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/debian
    bullseye stable"
sudo apt update
sudo apt install docker-ce
```

3. **Installez et lancez Jenkins via Docker** :

```
sudo docker pull jenkins/jenkins:lts
sudo docker run -d -p 8080:8080 -p 50000:50000 -v jenkins_home:/var/jenkins_home --name
    jenkins jenkins/jenkins:lts
```

4. **Récupération du mot de passe initial** :

```
sudo docker exec -it jenkins cat /var/jenkins_home/secrets/initialAdminPassword
```

Accédez à Jenkins en ouvrant un navigateur et en naviguant vers `http://[EXTERNAL_IP]:8080`.

2.1 Configuration de Jenkins pour la Migration Dynamique

1. **Installez les plugins nécessaires sur Jenkins** :
   - Kubernetes
   - Kubernetes Credentials
   - Kubernetes CLI
2. **Configurez les credentials pour accéder aux clusters Kubernetes depuis Jenkins** :
   (a) **Créer un compte de service Jenkins dans votre cluster** :

```
kubectl create serviceaccount jenkins

cat <<EOF | kubectl create -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
 name: jenkins-integration
 labels:
   k8s-app: jenkins-image-builder
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: jenkins
  namespace: default
EOF
```

   (b) **Récupérez le token du service account Jenkins** :

```
  kubectl get secrets $(kubectl get serviceaccounts jenkins -o jsonpath='{.secrets[].
      name}') -o jsonpath='{.data.token}' | base64 -d
```

   (c) **Ajoutez le token dans les credentials de Jenkins** :
      i. Connectez-vous à Jenkins, cliquez sur "Manage Jenkins" > "Credentials" > puis cliquez sur "global" et cliquez sur "Add Credentials".
      ii. Ajoutez le token obtenu comme un nouveau secret textuel.

   (d) **Récupérez le fichier de configuration kubeconfig** :

```
cd ~/.kube
cat config
echo -n <contents_of_the_certificate-authority-data_entry_of_my_kubeconfig_file> |
    base64 --decode
```

   (e) **: Intégrer le cluster Kubernetes distant avec Jenkins** : Pour intégrer le cluster Kubernetes distant avec Jenkins, suivez les étapes ci-dessous :

   Remplissez la configuration du plugin Kubernetes. Pour ce faire, ouvrez l'interface utilisateur de Jenkins et naviguez vers Gérer Jenkins > Gérer les Nœuds et les Clouds > Configurer les Clouds > Ajouter un nouveau cloud > Kubernetes et entrez l'URL de Kubernetes et le certificat

3 Configuration des Pipelines Jenkins

3.1 Création des Pipelines Jenkins

1. **Configurez les pipelines Jenkins suivants** : Les fichiers Jenkinsfile pour chaque pipeline sont stockés dans un dépôt GitHub. Suivez les étapes ci-dessous pour configurer chaque pipeline en pointant vers le fichier Jenkinsfile approprié dans GitHub.

- **Pipeline de déploiement (jenkinsfile_déploiement)** :
  (a) Ce pipeline déploie les services sur les nœuds edge.
  (b) Accédez à Jenkins, cliquez sur "New Item", nommez le pipeline "Déploiement", et sélectionnez "Pipeline".
  (c) Sous "Pipeline", sélectionnez "Pipeline script from SCM".
  (d) Choisissez "Git" comme SCM et entrez l'URL de votre dépôt GitHub contenant le fichier `jenkinsfile_déploiement`.
  (e) Spécifiez le chemin d'accès au fichier Jenkinsfile : `jenkinsfile_déploiement`.
- **Pipeline de surcharge des nœuds (jenkinsfile_chargeNode)** :
  (a) Ce pipeline simule la surcharge des nœuds edge.
  (b) Accédez à Jenkins, cliquez sur "New Item", nommez le pipeline "Charge Node", et sélectionnez "Pipeline".
  (c) Sous "Pipeline", sélectionnez "Pipeline script from SCM".
  (d) Choisissez "Git" comme SCM et entrez l'URL de votre dépôt GitHub contenant le fichier `jenkinsfile_chargeNode`.
  (e) Spécifiez le chemin d'accès au fichier Jenkinsfile : `jenkinsfile_chargeNode`.
- **Pipeline de vérification des ressources (jenkinsfile_checkEdgeRessource)** :
  (a) Ce pipeline vérifie les ressources des nœuds edge et déclenche la migration vers le cloud si nécessaire.
  (b) Accédez à Jenkins, cliquez sur "New Item", nommez le pipeline "Check Edge Resource", et sélectionnez "Pipeline".
  (c) Sous "Pipeline", sélectionnez "Pipeline script from SCM".
  (d) Choisissez "Git" comme SCM et entrez l'URL de votre dépôt GitHub contenant le fichier `jenkinsfile_checkEdgeRessource`.
  (e) Spécifiez le chemin d'accès au fichier Jenkinsfile : `jenkinsfile_checkEdgeRessource`.
- **Pipeline de migration Cloud-edge (jenkinsfile_migration_Cloud-edge)** :
  (a) Ce pipeline migre les services du cloud vers les nœuds edge.
  (b) Accédez à Jenkins, cliquez sur "New Item", nommez le pipeline "Migration Cloud-Edge", et sélectionnez "Pipeline".
  (c) Sous "Pipeline", sélectionnez "Pipeline script from SCM".
  (d) Choisissez "Git" comme SCM et entrez l'URL de votre dépôt GitHub contenant le fichier `jenkinsfile_migration_Cloud-edge`.
  (e) Spécifiez le chemin d'accès au fichier Jenkinsfile : `jenkinsfile_migration_Cloud-edge`.

2. **Remarque** : N'oubliez pas de modifier les informations de cluster dans chaque pipeline Jenkins. Cela inclut la mise à jour du nom des credentials et les adresses des clusters. Assurez-vous que ces informations sont correctes et spécifiques à votre environnement de déploiement.

4 Configurer la VM "Picar-X"

1. **Connectez-vous à la VM** :

```
gcloud compute ssh platoon-vm --zone europe-west1-b
```

2. **Clonez le dépôt Git contenant les fichiers nécessaires** :

```
git clone https://github.com/RacemMoalla/Meet-in-the-Middle.git
cd Meet-in-the-Middle/Picar-X
```

3. **Installez les dépendances nécessaires pour simuler la voiture autonome** :
   (a) Le fichier `requirement.txt` est déjà présent dans le dépôt cloné.
   (b) Exécutez les commandes suivantes :

```
sudo apt install -y python3 python3-pip
pip3 install --no-cache-dir --break-system-packages -r requirement.txt
```

4. **Note** : Le fichier `data.json` contient l'adresse IP du service de décision. Cette adresse sera mise à jour dynamiquement après chaque migration.

5. **Installez les dépendances pour le serveur Node.js** :

```
sudo apt install -y nodejs npm
cd /path/to/MAJ-de-l'adresse-ip-du-service-decision.js
npm install express body-parser fs
```

6. **Lancez le serveur MAJ-de-l'adresse-ip-du-service-decision.js pour écouter et mettre à jour l'adresse dans `data.json` après chaque migration** :

```
node MAJ-de-l'adresse-ip-du-service-decision.js
```

7. **Lancez le simulateur de données** :

```
python3 simulateur_de_donnees.py
```

   - Ce simulateur est un service qui simule les données envoyées par les capteurs de la voiture et les transmet au service de passerelle (gateway).

8. **Lancez le service de passerelle (gateway)** :

   (a) Le service de passerelle récupère les données simulées et les envoie au service de décision pour traitement.

   (b) En cas de dégradation de la latence dans le cloud, ce service déclenche le job de migration vers le edge.

   (c) Assurez-vous de mettre à jour les informations de Jenkins et le token dans le fichier de configuration.

```
python3 gateway.py
```

5 Conclusion

Ce document a détaillé la préparation et la configuration d'une infrastructure distribuée sur Google Cloud Platform (GCP) pour la migration dynamique des services entre le cloud et l'edge. Nous avons couvert la création de clusters Kubernetes et de machines virtuelles, ainsi que l'installation et la configuration de Jenkins pour orchestrer ces migrations.

Nous avons décrit la configuration des pipelines Jenkins pour divers scénarios de déploiement et de migration, ainsi que la simulation d'une voiture autonome avec la VM "Picar-X". Des instructions spécifiques pour vérifier les ressources des nœuds edge et déclencher les migrations appropriées ont également été fournies.

Cette documentation vise à être un guide complet pour la mise en œuvre de ce système complexe, en facilitant la compréhension et la réplication du processus.

L'intégration de notre cas d'utilisation, la voiture autonome, dans cette architecture distribuée, démontre la flexibilité et la scalabilité de notre solution.