TP FINAL

DOCKER, KUBERNETES & ANSIBLE

- Membre 1
- Membre 2
- Membre 3

INFORMATIONS

- Dans chaque groupe, déterminez qui est le membre 1, qui est le membre 2 et qui est le membre 3.

- Chaque groupe doit travailler dans son propre espace de noms isolé.

PARTIE 1

Membre 2

- Récupérer le code source de django-welcome depuis https://github.com/formation-kubernetes/django-welcome
- Générer l'image docker avec pour nom : [NOM-GROUPE]-app et avec pour version v1.
- Publier l'image générée sur un registre public (docker hub ou gitlab)

Membre 3

- Récupérer l'image publiée par le **membre 2**.
- Lancer le conteneur avec la commande python manage.py runserver 0.0.0.0:7000.
- Vérifier que le site est fonctionnel en l'ouvrant dans votre navigateur local.

Membre 1

- Créer un espace de noms nommé [NOM_EQUIPE]-alone
- Créer une resource deployment nommé [NOM_EQUIPE]-app dans le cluster K8S, charger de lancer l'image publiée par le **membre 2** en créant/gérant 1 pod.
- Créer un service externe nommé [NOM_EQUIPE]-svc qui va servir le deployment [NOM_EQUIPE]-app.

Membre 2

- Vérifier que le deployment [NOM_EQUIPE]-app est actif et fonctionnel
- Vérifier que le service externe [NOM_EQUIPE]-svc est bien actif et l'adresse IP externe disponible.
- Copier l'adresse IP externe du service et le mettre dans le tchat en taggant le formateur.
- Ouvrir l'IP externe du service créé sur un navigateur et vérifier que le site django est accessible.

PARTIE 2

Membre 3

- Récupérer le code source de django-withconfig depuis https://github.com/formation-kubernetes/django-withconfig
- Générer l'image docker avec pour nom : [NOM-GROUPE]-app et avec pour version v2.
- Publier l'image générée sur un registre public (docker hub ou gitlab)

Membre 2

- Créer un configmap nommé [NOM_EQUIPE]-config avec pour variables MEMBER_1, MEMBER_2 et MEMBER_3 ; qui vont contenir respectivement les noms du membre 1, du membre 2 et du membre 3.

- Créer un Secret nommé [NOM_EQUIPE]-secret avec pour variable TEAM ; qui va contenir le nom de l'équipe.

Membre 1

- Vérifier que la configmap [NOM_EQUIPE]-config est disponible avec les bonnes variables.

- Vérifier que le secret [NOM_EQUIPE]-secret est bien disponible avec la bonne variable et dont les valeurs sont hash.

Membre 2

- Modifier le deployment [NOM_EQUIPE]-app (sans supprimer la ressource déjà en cours) afin qu'il lance la dernière version de l'image publiée par le **membre 3** et en lui fournissant les variables d'environnements [NOM_EQUIPE]-config et [NOM_EQUIPE]-secret .

- Ouvrez l'IP externe du service [NOM_EQUIPE]-svc sur un navigateur et vérifiez que le nouveau site est accessible et que les informations de l'équipe sont affichées (nom de l'équipe et noms des membres).

PARTIE 3

Membre 1

- Dans l’espace de nom de l’équipe, Installer prometheus sur un port compris entre 30000 et 33000.
- Dans l’espace de nom de l’équipe, Installer grafana sur le port compris entre 30000 et 33000.
- Obtenir l’adresse IP du node.
- Ouvrir le navigateur, se connecter à grafana et ajouter prometheus comme source de données.

Membre 3

- Construire votre dashboard grafana en ajoutant des graphiques relatives aux métriques des pods, des deployments, des services, etc.

- Copier le lien d’accès au tableau de bord grafana et le mettre dans le tchat en taggant le formateur.

PARTIE 4

Membre 1

- Récupérer le code source de django-withcrash depuis https://github.com/formation-kubernetes/django-withcrash
- Générer l'image docker avec pour nom : [NOM-GROUPE]-app et avec pour version v3.
- Publier l'image générée sur un registre public (docker hub ou gitlab)

Membre 3

- Modifier le deployment [NOM_EQUIPE]-app (sans supprimer la ressource déjà en cours) afin qu'il lance la dernière version de l'image publiée par le **membre 1;** avec 3 replicas.

- Vérifier que le lancement du deployment à réussi, ensuite, Ouvrez l'IP externe du service [NOM_EQUIPE]-svc dans un navigateur et vérifiez que le site crash.

- Consulter la description du deployment sur le cluster K8S et assurez-vous qu'il lance la version v3 de l'image [NOM-GROUPE]-app ; Vérifier également que 3 pods sont lancés.

Membre 2

- Faites un rollback sur la version précédente fonctionnelle de l’application.
- Ouvrez l'IP externe du service [NOM_EQUIPE]-svc sur un navigateur et vérifiez que le site est de nouveau opérationnel.
- Consulter la description du deployment sur le cluster K8S et assurez-vous qu’il lance la version v2 de l’image [NOM-GROUPE]-app; Vérifiez également qu'un seul pod a été lancé.

PARTIE 5

Membre 3

- Créer un persitentvolumeclaim nommé [NOM_EQUIPE]-mongo-pvc ; disposant de 2Gi en capacité de stockage.

- Créer une resource deployment nommé [NOM_EQUIPE]-mongo-deployment, charger de lancer l'image mongo sur 2 replicas ; en utilisant le persitentvolumeclaim [NOM_EQUIPE]-mongo-pvc comme volume de stockage.

- Créer un service interne nommé [NOM_EQUIPE]-mongo-svc qui va servir le deployment [NOM_EQUIPE]-mongo-deployment.

Membre 2

- Vérifier le persistantvolumeclaim [NOM_EQUIPE]-mongo-pvc est monté.
- Vérifier que les pods et le deployment [NOM_EQUIPE]-mongo-deployment sont opérationnel.
- Vérifier que le service [NOM_EQUIPE]-mongo-svc est disponible.

Membre 1

- Créer une resource deployment nommé [NOM_EQUIPE]-mongo-express-deployment, charger de lancer l'image mongo-express pour administrer le serveur mongo [NOM_EQUIPE]-mongo-svc ; avec pour adminusername le prénom d'un membre d'équipe et pour adminpassword une combinaison de 8 chiffres.

- Créer un service externe nommé [NOM_EQUIPE]-mongo-express-svc qui va servir le deployment [NOM_EQUIPE]-mongo-express-deployment.

Membre 2

- Vérifier que les pods et le deployment [NOM_EQUIPE]-mongo-express-deployment sont opérationnel.
- Vérifier que le service [NOM_EQUIPE]-mongo-express-svc est disponible.
- Obtenir l'IP externe du service [NOM_EQUIPE]-mongo-express-svc ; et accéder à mongodb-express via son navigateur.

Membre 2

- Créer une collection via MongoDB Express et y ajouter 3 data.

- Supprimer le deployment [NOM_EQUIPE]-mongo-deployment ; Recréer ensuite un deployment avec le même nom mais disposant de 3 replicas.

- Vérifiez que les données créées précédemment à l'aide de MongoDB Express sont toujours présentes et n'ont pas été supprimées..

PARTIE 6

Membre 1

- Dans un fichier playbook ansible, rédiger les tâches pour :

- Supprimer et recréer le service [NOM_EQUIPE]-svc
- Déployer les ressources deployment pour mongo ([NOM_EQUIPE]-mongo-deployment) et mongo-express ([NOM_EQUIPE]-mongo-express-deployment).
- Déployer un ReplicaSet pour l'image httpd.
- Supprimer un ingress nommé [NOM_EQUIPE]-ingress

- Exécutez le fichier et assurez-vous que toutes les tâches ont été accomplies avec succès ou non !

PARTIE 7

Membre 3

- Récupérer le code source de django-private depuis https://github.com/formation-kubernetes/django-private
- Générer l'image docker avec pour nom : [NOM-GROUPE]-app-private et avec pour version v1.
- Publier l'image générée sur un registre privée (gitlab)

Membre 2

- Modifier le fichier (sans exécuter) de la ressource deployment [NOM_EQUIPE]-app pour qu'il lance la dernière version de l'image publiée par le **membre 3;** avec 1 replicas.

- Ajouter une tâche au fichier playbook ansible pour exécuter le fichier de la ressource deployment [NOM_EQUIPE]-app

- Exécuter ansible et Vérifier que le deployment [NOM_EQUIPE]-app a échoué au démarrage ; vérifier que l'échec est dû à l'impossibilité pour les pods de récupérer l'image.

Membre 3

- Générer une clé sur gitlab permettant la récupération de l'image sur les registres privés.
- Transmettre ces clés au membre 1.

Membre 1

- Créer une ressource secret [NOM_EQUIPE]-registry pour stocker les clés fournies par le membre 1.

- Modifier le fichier (sans exécuter) de la ressource deployment [NOM_EQUIPE]-app en lui fournissant la ressource secret [NOM_EQUIPE]-registry lui permettant de récupérer l'image privée.

- Exécuter ansible et Vérifier que le deployment [NOM_EQUIPE]-app a réussi au démarrage ; vérifier que le site est fonctionnel en utilisant le navigateur.