# Projet Kubernetes

Le projet se déroule sur 5 parties couvrant l'installation de K8S, le déploiement , la surveillance et la sécurité d'applications 'cloud native'.
Nous utiliserons Amazon AWS tout au long du projet.
Les repos git que chaque stagiaire doit utiliser seront disponible dans cette organization Github: https://github.com/fc4it-k8s

**Partie 1 - Déployer un cluster K8S et installer les microservices de démos :**

- Déployez un cluster kubernetes 1.19 :

- Créez deux instances aws EC2 (1 master et 1 worker) en utilisant une image ubuntu 18 bionic.
- Utilisez kubeadm pour le provisionning de votre master et ajouter l'instance woker comme membre du cluster.

- Déployez l'application de démos:

- kubectl apply -f https://raw.githubusercontent.com/fc4it-k8s/ecsdemo-nodejs/master/kubernetes/deployment.yaml
- kubectl apply -f https://raw.githubusercontent.com/fc4it-k8s/ecsdemo-nodejs/master/kubernetes/service.yaml
- Exposez le service en tant que node port et visualiser l'output dans un navigateur
- Mettez à l'échelle le déploiement (3 réplicas).
  - Mettez à jour la version de Kubernetes vers la version 1.21
  - Configurez le dashboard de Kubernetes.

**Partie 2 - Migration vers EKS**

- **Déploiement du cluster EKS:**

Dans cette partie, nous utiliserons eksctl pour lancer et configurer notre cluster et nos nœuds EKS. La version eksctl doit être 0.38.0 ou supérieure pour déployer EKS 1.20.

- **Redéployer les microservices de la partie 1 dans le cluster EKS.**

- **Mise à jour de votre cluster eks:**

Dans cette étape, vous suivrez le processus suggéré par l'AWS pour mettre à niveau votre cluster de 1.20 à 1.21 :

- (Facultatif) Vérifiez si la nouvelle version que vous mettez à niveau comporte des dépréciations de l'API, ce qui signifie que vous devrez modifier vos fichiers YAML Spec pour qu'ils puissent continuer à travailler sur le nouveau cluster. Ce n'est le cas que pour certaines mises à jour de versions telles que 1.15 à 1.16. Il existe différents outils qui peuvent aider avec cela comme kube-no-trouble. Comme il n'y a pas de telles dépressions allant de 1.20 à 1.21, nous allons sauter cette étape ici.

**1-** Exécutez un "kubectl get node" et assurez-vous que tous vos nœuds exécutent la version actuelle. Kubernetes ne peut prendre en charge que les nœuds qui sont une version derrière - ce qui signifie qu'ils doivent tous correspondre à la version actuelle de Kubernetes afin que lorsque vous mettez à niveau le plan de contrôle EKS, ils ne soient qu'une version derrière.

**2-** Mise à jour du plan de contrôle EKS vers la nouvelle version majeure

**3-** Vérifier si les modules complémentaires de base (kubeproxy, CoreDNS et CNI) expédiés avec EKS nécessitent une mise à jour vers

la version principale. Dans notre cas (une mise à jour 1.20 à 1.21), nous devrons mettre à jour à la fois CoreDNS et kubeproxy.

**4-** Mettre à jour tous les nœuds de travail afin que le kubelet sur eux (qui sera désormais une version majeure de Kubernete derrière) corresponde à celui du plan de contrôle EKS. Bien que vous n'ayez pas à le faire immédiatement, c'est une bonne idée d'avoir les Nodes sur la même version que le plan de contrôle dès que possible - en plus, cela rendra l'étape 2 plus facile la prochaine fois que vous devez mettre à niveau.
Comme vous utilisez des groupes de nœuds gérés , EKS peut vous aider avec un processus automatisé et sécurisé qui orchestre à la fois le côté AWS et Kubernetes d'un remplacement/mise à jour de nœud.

**Partie 3 - Surveillance avec PROMETHEUS et GRAFANA:**

- Déployer Prometheus :

Nous utiliserons helm pour installer les outils de surveillance Prometheus & Grafana pour ce chapitre. Veuillez passer en revue les instructions d'installation de la barre (vues pendant les formations) si vous ne les avez pas installées.

- ajouter prométhée Helm repo

> helm repo add prometheus-community https://prometheus-community.github.io/helm-charts

- ajouter grafana Helm repo

> helm repo add grafana https://grafana.github.io/helm-charts

- Déployez Grafana :

Nous utiliserons les valeurs par défaut de Grafana, mais nous outrepassons plusieurs paramètres. Comme avec Prometheus, nous paramétrons la classe de stockage sur gp2, mot de passe admin, configurant la source de données pour pointer vers Prometheus et créant un équilibreur de charge externe pour le service.

- Tableau de bord de la surveillance du cluster

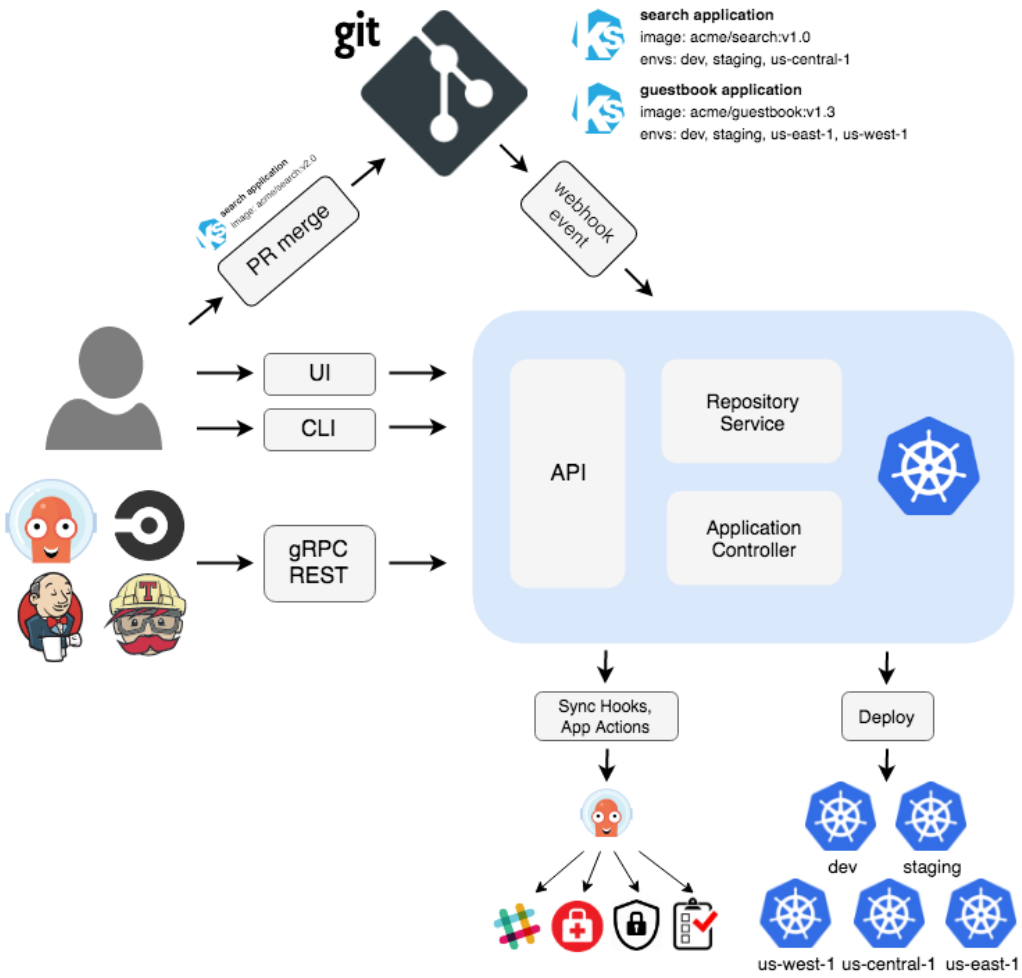
Utilisez l'identifiant 3119 du tableau de bord sous pour créer un tableau de bord de grappe.

- Tableau de bord de surveillance des unités

Vous devez créer un tableau de bord pour surveiller toutes les unités. Utilisez l'ID du tableau de bord 6417 sous Grafana.com Dashboard pour importer le tableau de bord.

## Partie 4 - Déploiement continu avec ARGOCD:

### 4.1. Installer argocd



ArgoCD est composé de trois composants principaux :

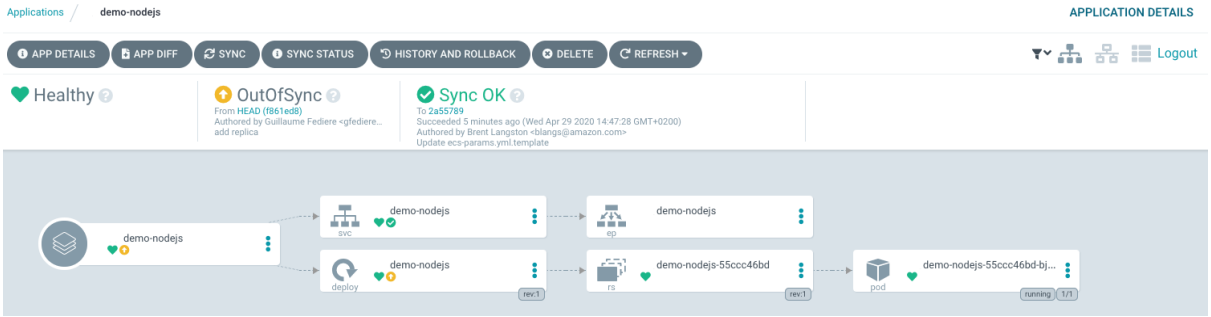
- Serveur API : expose l'API pour les systèmes WebUI / CLI / CICD
- Serveur de dépôt : Service interne qui maintient un cache local du dépôt git contenant les manifestes de l'application.
- Contrôleur d'application : Contrôleur Kubernetes qui contrôle et surveille les applications en continu et compare cet état actuel avec l'état cible souhaité (spécifié dans le référentiel).

Tous ces composants pourraient être installés à l'aide d'un manifeste fourni par le projet Argo.
https://raw.githubusercontent.com/argoproj/argo-cd/v2.0.4/manifests/install.yaml

### 4.2. Mettre à jour l'application:

Notre application est maintenant déployée dans notre ArgoCD. Nous allons maintenant mettre à jour notre dépôt github synchronisé avec notre application, vous devez changer la **spec.replicas: 2** dans le fichier Mettre à jour la spec.replicas: 2 dans le fichier **deployment.yaml**

Accédez à l'interface web d'ArgoCD pour synchroniser et déployer la nouvelle version.




## Chapitre 5- CIS EKS BENCHMARK ASSESSMENT USING KUBE-BENCH:

Dans cette partie, nous examinons comment évaluer les nœuds de cluster EKS d'Amazon que vous avez créés par rapport au benchmark CIS EKS Kubernetes.

- Installer kube-bench dans le nœud
- Exécuter le job k8s de kube-bench
- Lancer kube-bench en mode debug

**Livrables :**

- Un cluster installé avec kubeadm avec l'implémentation des étapes de la partie 1.
- Un cluster EKS avec les différentes étapes des parties 2,3,4 et 5 implémentées.
- Les slides à présenter durant la soutenance.