

# TP3 : Kubernetes dans un Contexte DevOps

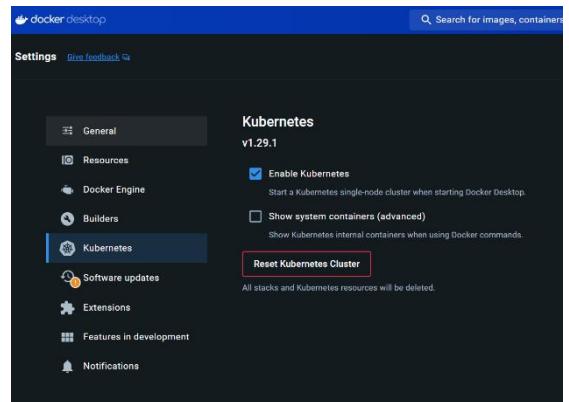
## 1. Installation et configuration de Kubernetes :

Kubernetes est une plateforme open-source d'orchestration de conteneurs, permettant la gestion automatisée et le déploiement d'applications conteneurisées.

**Docker Desktop** intègre **un serveur et un client Kubernetes autonomes**, Le serveur Kubernetes fonctionne localement au sein de l'instance Docker. Il s'agit d'un cluster mono-noeud qui s'exécute dans un conteneur Docker localement. Ce cluster est spécifiquement conçu pour les tests et le développement locaux, offrant une solution pratique sans nécessiter de configuration complexe.

### Activation de Kubernetes :

Une fois Docker Desktop installé, l'option permettant d'activer Kubernetes a été sélectionnée dans les paramètres de Docker Desktop. Cette fonctionnalité intégrée simplifie le déploiement et la gestion de clusters Kubernetes locaux.



### Vérification de l'installation :

Après l'activation de Kubernetes, la version de kubectl a été vérifiée en exécutant la commande **kubectl version** dans le terminal

```
PS C:\Users\MSI> kubectl version
Client Version: v1.29.1
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
Server Version: v1.29.1
PS C:\Users\MSI>
```

La version du client et du serveur Kubernetes a été affichée, confirmant que Kubernetes était installé et fonctionnel

### Vérification de l'état du cluster :

En utilisant la commande **kubectl cluster-info**, l'état du cluster Kubernetes a été vérifié. Les informations affichées ont confirmé que le cluster était prêt à être utilisé pour le déploiement d'applications.

```
PS C:\Users\MSI> kubectl cluster-info
Kubernetes control plane is running at https://kubernetes.docker.internal:6443
CoreDNS is running at https://kubernetes.docker.internal:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

Cette commande nous a fourni des informations sur l'état du cluster, y compris les URL de l'API Kubernetes et du tableau de bord. Si ces informations sont affichées, cela confirme que votre cluster est opérationnel.

la commande **kubectl get nodes** a été exécutée Cela permet de voir les nœuds du cluster.

```
PS C:\Users\MSI> kubectl get nodes
NAME     STATUS   ROLES      AGE     VERSION
docker-desktop   Ready    control-plane   4h54m   v1.29.1
PS C:\Users\MSI>
```

Un seul nœud nommé "docker-desktop" a été affiché, confirmant ainsi la présence d'un cluster Kubernetes fonctionnel sur Docker Desktop.

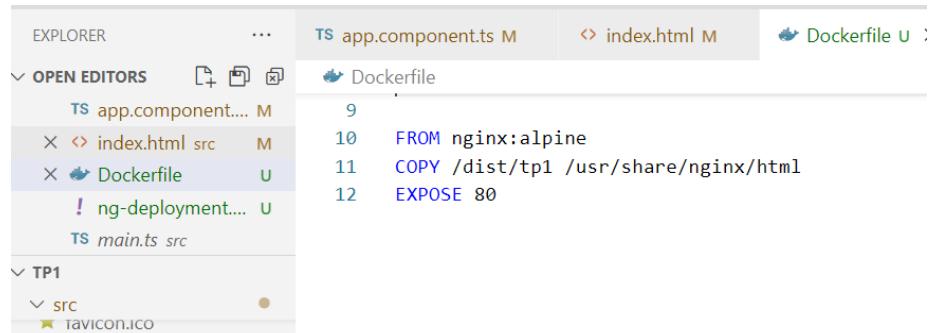
## 2. Construction de l'application Angular :

L'application Angular a été construite en utilisant la commande **ng build --prod**. Cela a généré les fichiers de production nécessaires pour l'application dans le répertoire **/dist**.

## 3. Construction de l'image Docker :

Un fichier Dockerfile a été créé dans le répertoire racine de l'application Angular pour spécifier comment l'application doit être empaquetée dans une image Docker.

Voici le contenu du fichier Dockerfile :



The screenshot shows a code editor interface with several files open:

- EXPLORER**: Shows files like `app.component.ts`, `index.html`, `Dockerfile`, `ng-deployment...`, and `main.ts`.
- OPEN EDITORS**: Shows the same files as the Explorer.
- TP1**: A folder containing `src` which has `TAVICON ICO`.

The **Dockerfile** content is as follows:

```
9
10 FROM nginx:alpine
11 COPY /dist/tp1 /usr/share/nginx/html
12 EXPOSE 80
```

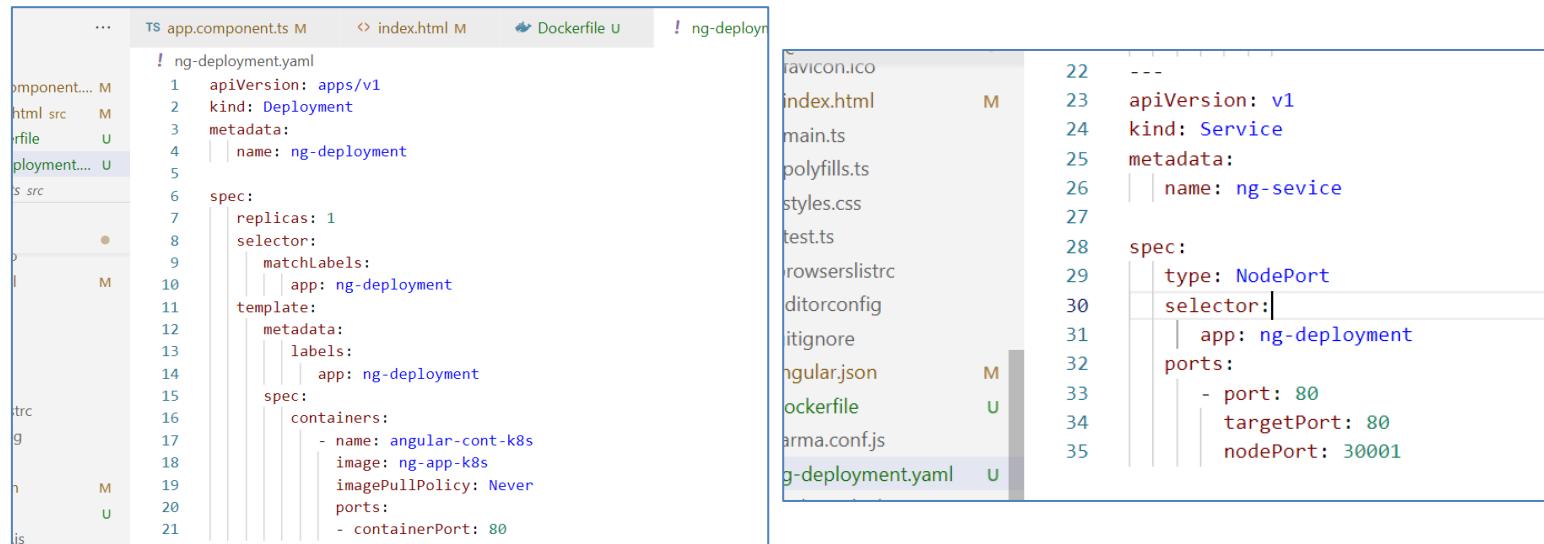
Ce Dockerfile utilise l'image NGINX Alpine comme base, copie les fichiers de l'application Angular du répertoire /dist/tp1 vers le répertoire NGINX, et expose le port 80 pour permettre l'accès à l'application.

La commande utilisée pour construire l'image Docker est : **docker build -t ng-app-k8s**.

#### 4. Déploiement sur Kubernetes et Définition d'un service :

Un fichier de configuration YAML nommé ng-deployment.yaml a été créé pour définir les spécifications du déploiement Kubernetes et du service pour l'application Angular.

Voici le contenu du fichier ng-deployment.yaml :



The screenshot shows a code editor with several files open. On the left, there's a sidebar with file icons. The main area has tabs for 'app.component.ts', 'index.html', 'Dockerfile', and 'ng-deployment.yaml'. The 'ng-deployment.yaml' tab is active, displaying the following YAML code:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ng-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ng-deployment
  template:
    metadata:
      labels:
        app: ng-deployment
    spec:
      containers:
        - name: angular-cont-k8s
          image: ng-app-k8s
          imagePullPolicy: Never
          ports:
            - containerPort: 80
```

On the right side of the editor, there's a vertical list of files: favicon.ico, index.html, main.ts, polyfills.ts, styles.css, test.ts, browserslistrc, ditorconfig, ignore, angular.json, Dockerfile, karma.conf.js, and ng-deployment.yaml. The 'ng-deployment.yaml' file is highlighted in green at the bottom of the list.

Ce fichier YAML définit un déploiement avec une réplique, spécifie l'image Docker à utiliser, et définit un service Kubernetes de type NodePort pour exposer l'application en interne.

La commande utilisée pour déployer sur Kubernetes est : **kubectl apply -f ng-deployment.yaml**

```
C:\Users\MSI\Desktop\insat\GL4\web angular\test\tp1>kubectl apply -f ng-deployment.yaml
deployment.apps/ng-deployment created
service/ng-sevice created
```

## 5. Vérification du déploiement et du service :

Les commandes **kubectl get deployments** et **kubectl get pods** ont été utilisées pour vérifier que le déploiement a été effectué correctement et que les pods sont en cours d'exécution.

```
PS C:\Users\MSI>
PS C:\Users\MSI> kubectl get deployments
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
ng-deployment   1/1     1           1          49m
PS C:\Users\MSI> kubectl get pods
NAME                  READY   STATUS    RESTARTS   AGE
ng-deployment-66d9f6d5fb-dccrh   1/1     Running   0          49m
PS C:\Users\MSI>
```

La commande **kubectl get services** a été utilisée pour vérifier que le service a été créé avec succès et obtenir des informations détaillées sur le service.

```
PS C:\Users\MSI> kubectl get services
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
kubernetes  ClusterIP  10.96.0.1      <none>        443/TCP      8h
ng-service  NodePort  10.111.244.96  <none>        80:30001/TCP  50m
PS C:\Users\MSI>
```

## 6. Test de l'application :

Pour confirmer que le déploiement était réussi, une vérification a été effectuée en accédant à l'application via le port NodePort spécifié.

En utilisant l'URL <http://localhost:30001> dans un navigateur web, l'application Angular a été testée avec succès et fonctionne correctement.

The screenshot shows a web browser window with the URL [localhost:30001](http://localhost:30001) in the address bar. The page displays a user interface for managing CVs and job offers. On the left, there's a sidebar with links like CvTech, Cv, Login, Mini word, RXJS, Products, and List MasterDetail. Below the sidebar, there's a section titled "Liste des CVs:" containing a search bar and two tabs: "Juniors" and "Seniors". Underneath are two card-like entries: one for "Aymen Sellaouti" and another for "Nidhal Jelassi". To the right of this is a large card for "Jelassi Nidhal, CEO" with a profile picture, a quote ("I'm the new Sinatra, and since I made it here I can make it anywhere, yeah, they love me everywhere."), and an "Auto Rotation" button. At the bottom, there's a section titled "Liste des Embauchés:" with a card for "Nidhal Jelassi" featuring a CV icon and his name.

La commande **kubectl apply -f ng-deployment.yaml** a été utilisée pour appliquer les modifications sur le cluster Kubernetes.

```
C:\Users\MSI\Desktop\insat\GL4\web angular\test\tp1>kubectl apply -f ng-deployment.yaml
deployment.apps/ng-deployment configured
service/ng-service unchanged
```

Vérification du déploiement et des nouvelles répliques :

Les commandes **kubectl get deployments** et **kubectl get pods** ont été utilisées pour vérifier que le déploiement a été mis à jour avec le nouveau nombre de répliques et que les nouvelles répliques étaient en cours de création et de déploiement sur les nœuds du cluster Kubernetes.

```
PS C:\Users\MSI> kubectl get deployments
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
ng-deployment   3/3     3           3          83m
PS C:\Users\MSI> kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
ng-deployment-66d9f6d5fb-d9h5v   1/1     Running   0          88s
ng-deployment-66d9f6d5fb-dccrh   1/1     Running   0          84m
ng-deployment-66d9f6d5fb-kk65f   1/1     Running   0          88s
PS C:\Users\MSI>
```

**NB** : la commande **kubectl delete -f ng-deployment.yaml** supprimerait le déploiement spécifié dans le fichier YAML ng-deployment.yaml, ainsi que tout service ou autre ressource liée qui a été créée en même temps que ce déploiement.

## 8. Exposition externe de l'application via un LoadBalancer :

Un fichier de configuration YAML nommé **ng-service-loadbalancer.yaml** a été créé pour définir un service de type LoadBalancer pour exposer l'application de manière externe.

Voici le contenu du fichier ng-service-loadbalancer.yaml :

```
36
37   apiVersion: v1
38   kind: Service
39   metadata:
40     name: ng-service-loadbalancer
41   spec:
42     selector:
43       app: ng-deployment
44     ports:
45       - port: 80
46         targetPort: 80
47       type: LoadBalancer
```

Cette configuration définit un service de type **LoadBalancer** qui sélectionne les pods du déploiement ng-deployment et expose le port 80.

La commande utilisée pour appliquer les modifications sur Kubernetes est :

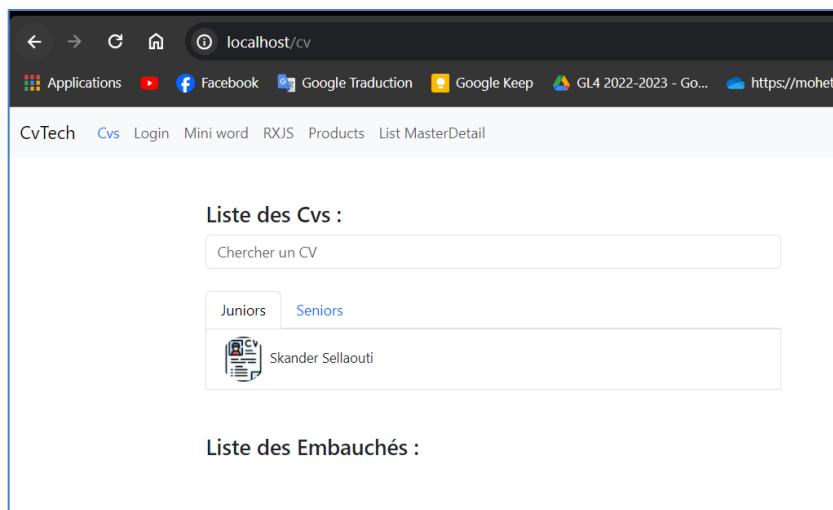
```
kubectl apply -f ng-service-loadbalancer.yaml
```

Vérification de l'accessibilité de l'application depuis l'extérieur :

Après avoir appliqué les modifications, nous avons utilisé la commande **kubectl get services** pour obtenir l'adresse IP externe attribuée au service de type LoadBalancer.

```
PS C:\Users\MSI> kubectl get services
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP  PORT(S)        AGE
kubernetes     ClusterIP  10.96.0.1    <none>       443/TCP       9h
ng-service-loadbalancer  LoadBalancer  10.98.164.69  localhost   80:31727/TCP  12s
ng-service     NodePort    10.111.244.96 <none>       80:30001/TCP  104m
PS C:\Users\MSI> kubectl describe services ng-service-loadbalancer
Name:           ng-service-loadbalancer
Namespace:      default
Labels:         <none>
Annotations:   <none>
Selector:       app=ng-deployment
Type:          LoadBalancer
IP Family Policy: SingleStack
IP Families:   IPv4
IP:             10.98.164.69
IPs:            10.98.164.69
LoadBalancer Ingress:  localhost
Port:          <unset>  80/TCP
TargetPort:    80/TCP
NodePort:      <unset>  31727/TCP
Endpoints:    10.1.0.17:80,10.1.0.18:80,10.1.0.19:80
Session Affinity: None
External Traffic Policy: Cluster
Events:        <none>
PS C:\Users\MSI>
```

Dans le navigateur web nous avons accédé à cette adresse IP externe « localhost » pour vérifier que l'application est accessible depuis l'extérieur du cluster.



## 9. Mise à jour de l'image du conteneur :

Une nouvelle version de l'image du conteneur a été construite en intégrant les mises à jour nécessaires à l'application. Cette nouvelle version a été taguée avec un identifiant unique : **v2.0**

Name	Tag
ng-app-k8s d40004354bcd	v2.0
ng-app-k8s bfce95ab3bf	latest

## Mise à jour du déploiement Kubernetes :

Le fichier `ng-deployment.yaml`, qui définit le déploiement de l'application sur Kubernetes, a été modifié pour référencer la nouvelle version de l'image du conteneur.

```
! ng-deployment.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: ng-deployment
5
6  spec:
7    replicas: 3
8    selector:
9      matchLabels:
10     app: ng-deployment
11    template:
12      metadata:
13        labels:
14          app: ng-deployment
15      spec:
16        containers:
17          - name: angular-cont-k8s
18            image: ng-app-k8s:v2.0
19            imagePullPolicy: Never
20            ports:
21              - containerPort: 80
```

## Application des modifications sur Kubernetes :

Les modifications ont été appliquées sur le cluster Kubernetes en utilisant la commande **kubectl apply -f** avec le fichier de configuration YAML modifié.

Cela a déclenché le processus de mise à jour du déploiement sur le cluster.

## Vérification de la mise à jour :

L'état du déploiement a été surveillé en utilisant la commande `kubectl get deployments` pour vérifier que la nouvelle version de l'application a été déployée avec succès.

La commande **kubectl get pods** a été utilisée pour vérifier que les nouveaux pods étaient en cours de création et que les anciens étaient terminés sans erreur.

NAME	READY	STATUS	RESTARTS	AGE
project-5fd897c54b-s8bsm	1/1	Running	0	5m13s
project-5fd897c54b-v5jrc	1/1	Terminating	0	5m13s
project-5fd897c54b-v7mv1	0/1	Terminating	0	5m13s
project-76f7c854cd-7k2hn	0/1	ContainerCreating	0	5s
project-76f7c854cd-mf664	1/1	Running	0	13s
project-76f7c854cd-wj9gs	1/1	Running	0	10s
project-5fd897c54b-v7mv1	0/1	Terminating	0	5m13s
project-5fd897c54b-v7mv1	0/1	Terminating	0	5m13s
project-5fd897c54b-v5jrc	0/1	Terminating	0	5m13s

L'application elle-même a été testée pour s'assurer que les fonctionnalités étaient toujours opérationnelles et qu'il n'y avait pas d'interruption de service pour les utilisateurs finaux.

The screenshot shows a web application interface. On the left, there's a sidebar with a search bar and two tabs: "Juniors" and "Seniors". Below the tabs are two resume cards. The first card is for "Aymen Sellaouti" and the second is for "Nidhal Jelassi". On the right, there's a large image of a person holding a resume with the word "CV" on it. Below the image, the text reads "Jelassi Nidhal" and "CEO". There's also a quote: "I'm the new Sinatra, and since I made it here I can make it anywhere, yeah love me everywhere".

## 10. Surveillance et journalisation :

### 1. Surveillance des ressources Kubernetes à l'aide de Prometheus :

#### a. Installation de Prometheus :

On commence par installer Prometheus dans notre cluster Kubernetes. On peut le faire à l'aide d'Helm, un gestionnaire de packages pour Kubernetes comme suit :

```
C:\Users\user>helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
"prometheus-community" already exists with the same configuration, skipping
```

```
C:\Users\user>helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "prometheus-community" chart repository
Update Complete. ⚡Happy Helming!⚡
```

```
C:\Users\user>helm install prometheus prometheus-community/prometheus
NAME: prometheus
LAST DEPLOYED: Mon Apr  1 22:48:51 2024
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The Prometheus server can be accessed via port 80 on the following DNS name from within your cluster:
prometheus-server.default.svc.cluster.local

Get the Prometheus server URL by running these commands in the same shell:
  export POD_NAME=$(kubectl get pods --namespace default -l "app.kubernetes.io/name=prometheus,app.kubernetes.io/instance=prometheus" -o jsonpath=".items[0].metadata.name")
  kubectl --namespace default port-forward $POD_NAME 9090

The Prometheus alertmanager can be accessed via port 9093 on the following DNS name from within your cluster:
prometheus-alertmanager.default.svc.cluster.local

Get the Alertmanager URL by running these commands in the same shell:
  export POD_NAME=$(kubectl get pods --namespace default -l "app.kubernetes.io/name=alertmanager,app.kubernetes.io/instance=prometheus" -o jsonpath=".items[0].metadata.name")
  kubectl --namespace default port-forward $POD_NAME 9093
#####
##### WARNING: Pod Security Policy has been disabled by default since #####
##### it deprecated after k8s 1.25+. use #####
##### (index .Values "prometheus-node-exporter" "rbac" #####
##### "pspEnabled") with (index .Values #####
##### "prometheus-node-exporter" "rbac" "pspAnnotations") #####
##### in case you still need it. #####
#####

The Prometheus PushGateway can be accessed via port 9091 on the following DNS name from within your cluster:
prometheus-prometheus-pushgateway.default.svc.cluster.local

Get the PushGateway URL by running these commands in the same shell:
  export POD_NAME=$(kubectl get pods --namespace default -l "app=prometheus-pushgateway,component=pushgateway" -o jsonpath=".items[0].metadata.name")
  kubectl --namespace default port-forward $POD_NAME 9091

For more information on running Prometheus, visit:
https://prometheus.io/
C:\Users\user>
```

Pour vérifier que l'installation s'est effectuée avec succès et tous les pods associé à prometheus marchent, on exécute **kubectl get pods** :

```
C:\Users\user>kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
prometheus-alertmanager-0          1/1     Running   0          54m
prometheus-kube-state-metrics-65468947fb-gwbnp  1/1     Running   0          54m
prometheus-prometheus-node-exporter-dkv9k        1/1     Running   0          54m
prometheus-prometheus-pushgateway-76976dc66-wxj72  1/1     Running   0          54m
prometheus-server-5bb9fdbccb-hwtcd      2/2     Running   0          54m
todo-app-5c7bf5c889-5p4hv          1/1     Running   2 (5h38m ago) 30h
todo-app-5c7bf5c889-c4hsg          1/1     Running   2 (5h38m ago) 30h
todo-app-5c7bf5c889-fhbft         1/1     Running   2 (5h38m ago) 30h
todo-database-6cd4b4c7b7-4c8ph       1/1     Running   2 (5h38m ago) 32h
```

On peut accéder aux configuration des métriques à scraper par Prometheus dans le config-map prometheus-server qui contient le fichier de configuration Prometheus en utilisant la commande kubectl describe configmap prometheus-server :

```
C:\Users\user>kubectl describe configmap prometheus-server
Name:          prometheus-server
Namespace:     default
Labels:        app.kubernetes.io/component=server
               app.kubernetes.io/instance=prometheus
               app.kubernetes.io/managed-by=Helm
               app.kubernetes.io/name=prometheus
               app.kubernetes.io/part-of=prometheus
               app.kubernetes.io/version=v2.51.1
               helm.sh/chart=prometheus-25.19.0
Annotations:   meta.helm.sh/release-name: prometheus
               meta.helm.sh/release-namespace: default

Data
====

prometheus.yml:
----  
global:  
  evaluation_interval: 1m  
  scrape_interval: 1m  
  scrape_timeout: 10s  
rule_files:  
- /etc/config/recording_rules.yml  
- /etc/config/alerting_rules.yml  
- /etc/config/rules  
- /etc/config/alerts  
scrape_configs:  
- job_name: prometheus  
  static_configs:  
  - targets:  
    - localhost:9090  
  bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token  
  job_name: kubernetes-apiservers  
  kubernetes_sd_configs:  
  - role: endpoints  
  relabel_configs:  
    action: keep
```

#### b. Visualisation des métriques avec Grafana :

Pour mieux comprendre les métriques scrapés par Prometheus, on peut utiliser **Grafana** qui est une plateforme de visualisation de données interactive Open Source qui s'intègre très bien avec Prometheus.

Après avoir installé Grafana, on exécute le binaire grafana-server.exe puis on accède à Grafana avec l'url <https://localhost:3000>. Par défaut, le port de Grafana est 3000 et les paramètres d'authentification sont : admin /admin :

The left side shows the command-line log output from the Grafana server, detailing its startup process. The right side shows the 'Welcome to Grafana' login screen with 'admin' as both the email and password.

```

C:\Users\user\Downloads\grafana-enterprise-10.4.1.windows-amd64\grafana-v10.4.1\bin\grafana-server.exe
[04-02-15:15:22] Completed alerting migration
[04-02-15:15:22] Running in alternative execution of Error/NoData mode logger=ngalert.state.manager
[04-02-15:15:22] registering usage stat providers
[04-02-15:15:22] starting to provision alerting
[04-02-15:15:22] finished to provision alerting
[04-02-15:15:22] Starting Multiorg Alertmanager
[04-02-15:15:22] Failed to get render plugin sources
[04-02-15:15:22] Warning state cache for startup
[04-02-15:15:22] Storage starting
[04-02-15:15:22] Scheduling and sending of reports disabled, SMTP is not configured and enabled. Configure SMTP in config file
[04-02-15:15:22] logger=report
[04-02-15:15:22] HTTP Server Listen
[04-02-15:15:22] State cache has been initialized
[04-02-15:15:22] Starting scheduler
[04-02-15:15:22] starting
[04-02-15:15:22] Database locked, sleeping then retrying
[try=0 code="database is locked"]
[04-02-15:15:22] starting to provision dashboards
[04-02-15:15:22] finished to provision dashboards
[04-02-15:15:23] Update check succeeded
[04-02-15:15:23] Update check succeeded
[04-02-15:15:23] Adding GroupVersion playlist.grafana.app v6alpha1 to ResourceManager logger=grafana.apiserver
[04-02-15:15:23] Adding GroupVersion featureToggle.grafana.app v6alpha1 to ResourceManager logger=grafana.apiserver
[04-02-15:15:23] Request Completed
[status=302 remote_addr=[::1] time_ms=1 duration=1.908ms size=29 referer= handlers/ status_source=server
[04-02-15:16:06] Usage stats are ready to report
[logger=infra.usagestats]

```

On clique sur « Data Source » sur la page d'accueil de grafana pour ouvrir le menu Configuration puis on sélectionne Prometheus comme type :

The left screenshot shows the Grafana home page with various links and sections like 'Basic', 'TUTORIAL', 'DATA SOURCE AND DASHBOARDS', 'GRAFANA FUNDAMENTALS', 'DATA SOURCES', and 'DASHBOARDS'. The right screenshot shows the 'Add data source' configuration dialog, specifically the 'Time series databases' section where 'Prometheus' is selected.

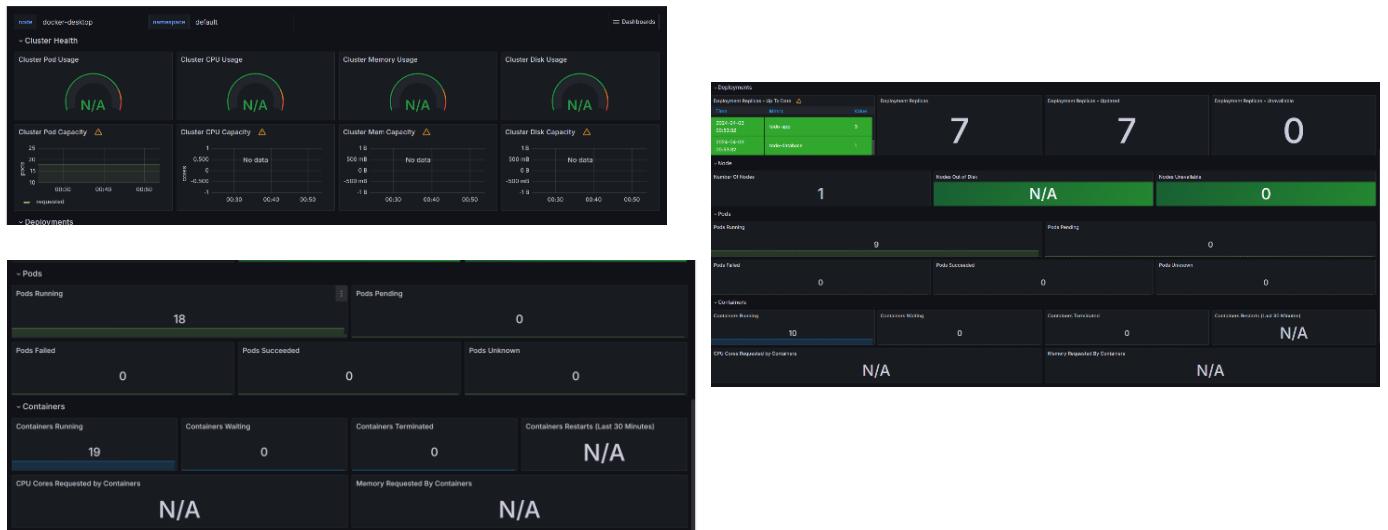
On définit l'URL du serveur Prometheus appropriée qui est <http://localhost:9090/> et on ajuste les autres paramètres de la source de données. Enfin on clique sur « Enregistrer et tester » pour enregistrer la nouvelle source de données :

The screenshot shows the 'Save & test' step of the configuration process. It displays a success message: 'Successfully queried the Prometheus API.' and instructions to start visualizing data. At the bottom are 'Delete' and 'Save & test' buttons.

Pour surveiller les données scapées par Prometheus, On importe le dashboard « Kubernetes Cluster(Prometheus) » qui donne un aperçu global de l'état des ressources du cluster Kubernetes, y compris le nombre de nœuds, de pods, l'utilisation de la CPU et de la mémoire, etc

Le dashboard montre que le nombre de noeuds dans le cluster « Docker Desktop » dans le namespace « default » est 1, le nombre de déploiements est 7 et le nombre de pods en cours d'exécution est 9 (la totalité des pods) de même pour les conteneurs ou tous les conteneurs en l'occurrence 10 sont en cours d'exécution sur le cluster ce qui explique que pour chaque pods on a au moins un conteneur.

De plus, d'après la courbe « Cluster Pod Capacity » on peut voir que de la capacité totale demandée pour les pods dans votre cluster Kubernetes est 18. Cette valeur représente la capacité demandée, qui peut être configurée pour répondre aux besoins spécifiques de l'application.



## 1. Journalisation des pods de l'application à l'aide de fluentd :

### a. Installation de fluentd :

```
C:\Users\user>helm repo add fluent https://Fluent.github.io/helm-charts
"fluent" has been added to your repositories

C:\Users\user>
C:\Users\user>helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "fluent" chart repository
...Successfully got an update from the "prometheus-community" chart repository
Update Complete. Happy Helm-ing!

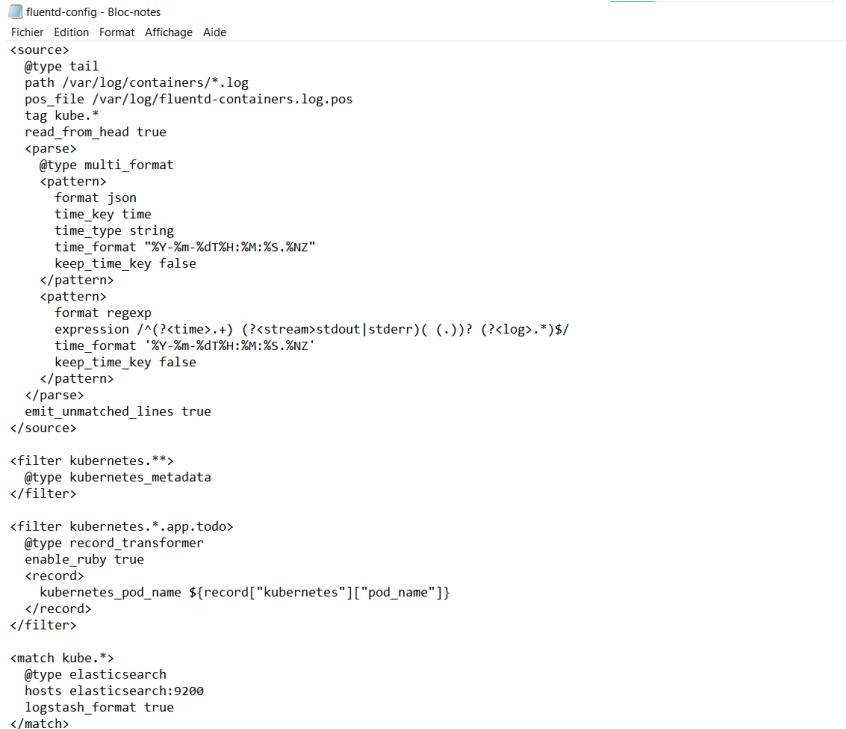
C:\Users\user>helm install fluent fluent/fluentd
NAME: fluent
LAST DEPLOYED: Wed Apr 3 12:14:52 2024
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
Get Fluentd build information by running these commands:

export POD_NAME=$(kubectl get pods --namespace default -l "app.kubernetes.io/name=fluentd,app.kubernetes.io/instance=fluentd" -o jsonpath=".items[0].metadata.name")
kubectl --namespace default port-forward $POD_NAME 24231:24231
curl http://127.0.0.1:24231/metrics
```

### b. Configuration de fluentd :

On crée un fichier de configuration Fluentd avec un nom approprié, par exemple **fluentd-config.yaml**. Ce fichier contient :

- Une **source** nommée `<tail>` pour lire les fichiers de logs dans `/var/log/containers/*.log`.
- Un **filtre** qui spécifie comment les logs doivent être analysés.
- Un **filtre kubernetes\_metadata** pour enrichir les logs avec des métadonnées Kubernetes telles que le namespace, le nom du pod, etc.
- Un **filtre record\_transformer** pour filtrer les logs des pods spécifiques de l'application.
- Une **destination** qui spécifie où envoyer les logs traités. Dans cet exemple, nous utilisons Elasticsearch comme destination.



```
<fluentd-config - Bloc-notes>
Fichier Edition Format Affichage Aide
<source>
  @type tail
  path /var/log/containers/*.log
  pos_file /var/log/fluentd-containers.log.pos
  tag kube./*
  read_from_head true
  <parse>
    @type multi_format
    <pattern>
      format json
      time_key time
      time_type string
      time_format "%Y-%m-%dT%H:%M:%S.%NZ"
      keep_time_key false
    </pattern>
    <pattern>
      format regexp
      expression /^(?<time>.+) (?<stream>stdout|stderr)( (.))?( ?<log>.*$)/
      time_format "%Y-%m-%dT%H:%M:%S.%NZ"
      keep_time_key false
    </pattern>
  </parse>
  emit_unmatched_lines true
</source>

<filter kube.**>
  @type kubernetes_metadata
</filter>

<filter kubernetes.*.app.todo>
  @type record_transformer
  enable_ruby true
  <record>
    kubernetes_pod_name ${record["kubernetes"]["pod_name"]}
  </record>
</filter>

<match kube.*>
  @type elasticsearch
  hosts elasticsearch:9200
  logstash_format true
</match>
```

Ensuite, on crée une ConfigMap Kubernetes pour le fichier de configuration Fluentd en utilisant la commande **kubectl create configmap** :

```
C:\Users\user\Desktop>kubectl create configmap fluentd-config --from-file=fluentd-config.yaml
configmap/fluentd-config created

C:\Users\user\Desktop>
```

Ensuite, on déploie Fluentd sur votre cluster Kubernetes en créant un fichier YAML pour le déploiement Fluentd par exemple nommé fluentd-deployment.yaml avec le champ **configMap.name** correspond au nom de la ConfigMap créée précédemment.

```
fluentd-deployment - Bloc-notes
Fichier Edition Format Affichage Aide
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluentd
  namespace: default  # Modifier le namespace à "default"
  labels:
    k8s-app: fluentd-logging
spec:
  selector:
    matchLabels:
      name: fluentd
  template:
    metadata:
      labels:
        name: fluentd
    spec:
      containers:
        - name: fluentd
          image: fluent/fluentd-kubernetes-daemonset:v1.16.5-debian-kafka-1.0
          ports:
            - containerPort: 24224
              name: fluentd
              protocol: TCP
          volumeMounts:
            - name: config-volume
              mountPath: /fluentd/etc
      volumes:
        - name: config-volume
          configMap:
            name: fluentd-config
```

On utilise la commande **kubectl apply** pour déployer le DaemonSet Fluentd :

```
C:\Users\user\Desktop>kubectl apply -f fluentd-deployment.yaml
daemonset.apps/fluentd created
```

Pour vérifier le déploiement de **Fluentd** que le DaemonSet Fluentd a été déployé avec succès en utilisant la commande suivante :

```
C:\Users\user\Desktop>kubectl get daemonset fluentd
NAME      DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
fluentd   1          1          0        1           0           <none>       67s

C:\Users\user\Desktop>
```

### c. Affichage des logs des pods :

On vérifie les logs de Fluentd en utilisant la commande suivante : **kubectl logs -l name=fluentd** :

```
2024-03-27 15:53:15 +0000 [info]: init supervisor logger path=/ml rotate_age=ml rotate_size=nl
2024-03-27 15:53:15 +0000 [info]: parsing config file is succeeded path=/fluentd/etc/../../../../etc/fluent/fluent.conf
2024-03-27 15:53:16 +0000 [info]: gem 'fluentd' version '1.16.2'
2024-03-27 15:53:16 +0000 [info]: gem 'fluent-plugin-concat' version '2.5.0'
2024-03-27 15:53:16 +0000 [info]: gem 'fluent-plugin-dedot_filter' version '1.0.0'
2024-03-27 15:53:16 +0000 [info]: gem 'fluent-plugin-detect-exceptions' version '0.0.15'
2024-03-27 15:53:16 +0000 [info]: gem 'fluent-plugin-elasticsearch' version '5.2.5'
2024-03-27 15:53:16 +0000 [info]: gem 'fluent-plugin-grok-parser' version '2.6.2'
2024-03-27 15:53:16 +0000 [info]: gem 'fluent-plugin-json-in-json-2' version '1.0.2'
2024-03-27 15:53:16 +0000 [info]: gem 'fluent-plugin-kubernetes_metadata_filter' version '3.2.0'
2024-03-27 15:53:16 +0000 [info]: gem 'fluent-plugin-multi-format-parser' version '1.0.0'
2024-03-27 15:53:16 +0000 [info]: gem 'fluent-plugin-passur-cri' version '0.1.1'
2024-03-27 15:53:16 +0000 [info]: gem 'fluent-plugin-prometheus' version '2.1.0'
2024-03-27 15:53:16 +0000 [info]: gem 'fluent-plugin-record-modifier' version '2.1.1'
2024-03-27 15:53:16 +0000 [info]: gem 'fluent-plugin-rewrite-tag-filter' version '2.4.0'
2024-03-27 15:53:16 +0000 [info]: gem 'fluent-plugin-systemd' version '1.0.5'
2024-03-27 15:53:19 +0000 [warn]: [filter_kube_metadata] !! The environment variable 'K8S_NODE_NAME' is not set to the node name which c
t the API server and watch efficiency !!
2024-03-27 15:53:20 +0000 [info]: using configuration file: <ROOT>
<label @FLUENT_LOG>
<match **>
  @type null
  @id ignore_fluent_logs
</match>
```

## 11. Réalisation d'un pipeline CI CD intégrant à la fois Jenkins, Docker, Git, Github, DockerHub, Apache Maven, Ansible et K8s :

Le pipeline qu'on propose dans cette partie commence par surveiller le référentiel Git où le code source de l'application est hébergé. Lorsqu'un développeur effectue des modifications et les pousse sur le référentiel, le pipeline est déclenché.

Ensuite, Maven entre en jeu. Il récupère le code source de l'application depuis Git et exécute des étapes telles que la compilation du code, l'exécution des tests unitaires et la gestion des dépendances à l'aide du fichier de configuration pom.xml.

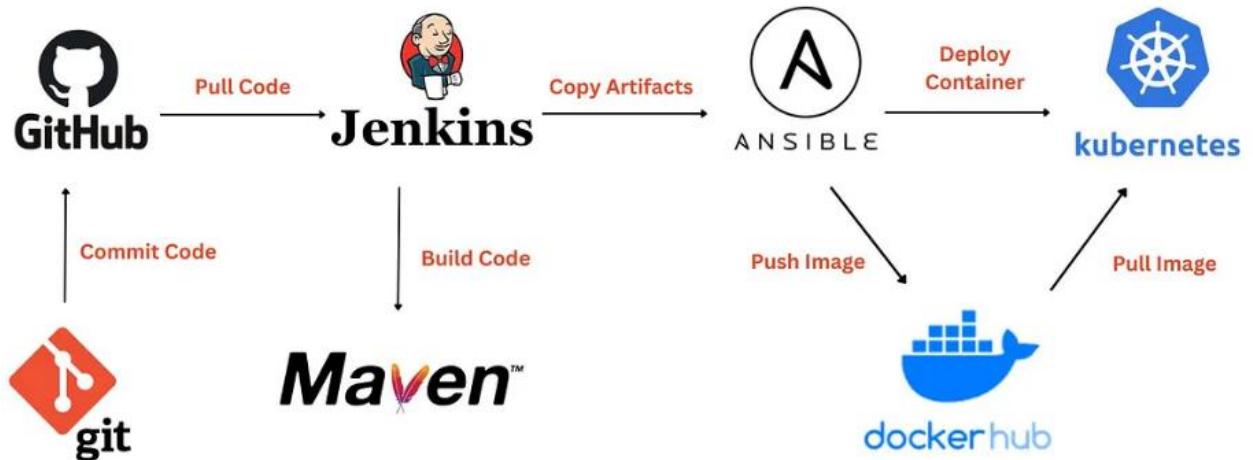
Ensuite, Ansible entre en jeu pour orchestrer les différentes étapes du processus. Tout d'abord, Ansible utilise des playbooks pour construire l'image Docker de l'application en fonction des spécifications définies. Cela peut inclure le téléchargement des dépendances nécessaires, la compilation du code et la configuration de l'environnement.

Une fois que l'image Docker est construite, Ansible continue en orchestrant le processus de push de cette image vers Docker Hub. Il utilise les modules Docker et les API Docker pour gérer cette opération, garantissant ainsi que l'image est correctement transférée vers le registre Docker Hub.

Après avoir poussé l'image avec succès, Ansible reprend le contrôle pour déployer l'application sur un cluster Kubernetes. Il utilise à nouveau des playbooks Ansible pour automatiser le déploiement de l'application sur les nœuds du cluster. Cela comprend la création des déploiements, des services et d'autres ressources Kubernetes nécessaires pour exécuter l'application de manière évolutive et résiliente.

Une fois toutes ces étapes terminées avec succès, l'application est déployée et disponible pour être utilisée par les utilisateurs finaux. Ce processus de pipeline CI/CD, orchestré par Ansible, garantit un déploiement automatisé et

cohérent de l'application à chaque mise à jour du code source, de la construction de l'image jusqu'au déploiement sur Kubernetes.



NB : Le travail qui suit a été réalisé dans une machine virtuelle Ubuntu

### 1. Jenkins :

#### a. Téléchargement de Jenkins sur l'instance VM

```

vboxuser@jenkins:~$ sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
--2024-04-03 22:28:23-- https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
Resolving pkg.jenkins.io (pkg.jenkins.io)... 151.101.2.133, 151.101.66.133, 151.101.130.133, ...
Connecting to pkg.jenkins.io (pkg.jenkins.io)|151.101.2.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3175 (3.1K) [application/pgp-keys]
Saving to: '/usr/share/keyrings/jenkins-keyring.asc'

/usr/share/keyrings/jenkins-keyr 100%[=====] 3.10K ---KB/s   in 0.003s
2024-04-03 22:28:28 (925 KB/s) - '/usr/share/keyrings/jenkins-keyring.asc' saved [3175/3175]

vboxuser@jenkins:~$ echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
  
```

#### b. Installation de Java dans l'instance VM :

```

PROCESSING triggers for netcat-openv (3.7.0-1ubuntu0.1) ...
vboxuser@jenkins:~$ java -version
openjdk version "17.0.10" 2024-01-16
OpenJDK Runtime Environment (build 17.0.10+7-Ubuntu-122.04.1)
OpenJDK 64-Bit Server VM (build 17.0.10+7-Ubuntu-122.04.1, mixed mode, sharing)
  
```

### c. Installation de fontconfig :

C'est une bibliothèque conçue pour fournir une configuration, une personnalisation et un accès aux applications de polices à l'échelle du système. Il est requis par Jenkins, car Jenkins dispose de fonctionnalités qui rendent les polices

```
vboxuser@jenkins: ~$ sudo apt-get install fontconfig openjdk-17-jre
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
fontconfig is already the newest version (2.13.1-4.2ubuntu5).
The following packages were automatically installed and are no longer required:
  libltdl-dev libltdl0 liblxss1-dev libxau-dev libxcb1-dev libxdmcp-dev libxt-dev openjdk-11-jdk-headless
  systemd-hwe-hwdb x11proto-dev xorg-sgml-doctools xtrans
Use 'sudo apt autoremove' to remove them.
Suggested packages:
  fonts-ipafont-mincho fonts-wqy-microhei | fonts-wqy-zhengl
The following NEW packages will be installed:
  openjdk-17-jre openjdk-17-jre-headless
0 upgraded, 1 newly installed, 0 to remove and 458 not upgraded.
Need to get 48.4 MB of archives.
After this operation, 0 B of additional disk space will be used.
Get: http://fr.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 openjdk-17-jre-headless amd64 17.0.10+7-1-22.04.1 [48.2 MB]
In�: http://fr.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 openjdk-17-jre-headless amd64 17.0.10+7-1-22.04.1
Get: http://fr.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 openjdk-17-jre amd64 17.0.10+7-1-22.04.1 [203 kB]
Get: http://fr.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 openjdk-17-jre-headless amd64 17.0.10+7-1-22.04.1 [48.2 MB]
Fetched 48.4 MB in 0s (0 B/s)
Selecting previously unselected package openjdk-17-jre-headless=amd64.
(Reading database ... 204676 files and directories currently installed.)
Preparing to unpack .../openjdk-17-jre-headless_17.0.10+7-1-22.04.1_amd64.deb ...
Unpacking openjdk-17-jre-headless (17.0.10+7-1-22.04.1) ...
Selecting previously unselected package openjdk-17-jre=amd64.
Preparing to unpack .../openjdk-17-jre_17.0.10+7-1-22.04.1_amd64.deb ...
Unpacking openjdk-17-jre (17.0.10+7-1-22.04.1) ...
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/java to provide /usr/bin/java (java) in auto mode
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/jpackage to provide /usr/bin/jpackage (jpackage) in auto mode
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/keytool to provide /usr/bin/keytool (keytool) in auto mode
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/rmiregistry to provide /usr/bin/rmiregistry (rmiregistry) in auto mode
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/lib/jexec to provide /usr/bin/jexec (jexec) in auto mode
setting up openjdk-17-jre=amd64 (17.0.10+7-1-22.04.1) ...
```

### d. Installation et démarrage de Jenkins:

```
vboxuser@jenkins: ~$ sudo apt install jenkins -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
jenkins is already the newest version (2.446.2).
The following packages were automatically installed and are no longer required:
  libgcj-dev libpthread-stubs0-dev libbz2-dev libx11-dev libxau-dev libxcb1-dev libxdmcp-dev libxt-dev openjdk-11-jdk-headless
  systemd-hwe-hwdb x11proto-dev xorg-sgml-doctools xtrans-dev
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 458 not upgraded.
After this operation, 0 B of additional disk space will be used.
Setting up jenkins (2.446.2) ...
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2024-04-03 22:41:20 WAT; 51s ago
       Main PID: 13087 (Java)
      Tasks: 53 (limit: 13108)
     Memory: 4.4M
        CPU: 983ms
       CGroup: /system.slice/jenkins.service
           └─13087 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --http

Apr 03 22:40:40 jenkins Jenkins[13087]: Please use the following password to proceed to installation:
Apr 03 22:40:40 jenkins Jenkins[13087]: e0c1199f90f5469287d292c4e94698a1
Apr 03 22:40:40 jenkins Jenkins[13087]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Apr 03 22:40:40 jenkins Jenkins[13087]: ****
Apr 03 22:40:40 jenkins Jenkins[13087]: 2024-04-03 21:40:43.249+0000 [id=54] INFO h.m.DownloadService$Downloadable#lo
Apr 03 22:41:20 jenkins Jenkins[13087]: 2024-04-03 21:41:20.705+0000 [id=57] INFO jenkins.InitReactorRunner$#onAttal
Apr 03 22:41:20 jenkins Jenkins[13087]: 2024-04-03 21:41:20.771+0000 [id=25] INFO hudson.lifecycle.Lifecycle#onReady:[
Apr 03 22:41:20 jenkins systemd[1]: Started Jenkins Continuous Integration Server.
lines 1-20/20 (END)

[1]+  Stopped                  sudo systemctl status jenkins
vboxuser@jenkins: ~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
e0c1199f90f5469287d292c4e94698a1
```

### e. Accès à Jenkins :

Pour accéder à l'application Jenkins, on ouvre un navigateur Web et on entre l'url **http://localhost :8080**. La première fois que vous accédez à Jenkins, il sera verrouillé avec un mot de passe généré automatiquement. Vous devez afficher ce mot de passe à l'aide de la commande suivante :

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

On Copie le mot de passe, puis on revient dans le navigateur, on le colle dans le champ Password administrateur et on clique sur « Continue ».

Ensuite, On peut voir cette page Web :

The screenshot shows the Jenkins 'Customize Jenkins' page. At the top, it says 'Getting Started' and 'Customize Jenkins'. Below that, it says 'Plugins extend Jenkins with additional features to support many different needs.' There are two main options: 'Install suggested plugins' (which is highlighted in blue) and 'Select plugins to install'. Both options have a brief description below them.

Install suggested plugins  
Select plugins to install

Install plugins the Jenkins community finds most useful.  
Select and install plugins most suitable for your needs.

Capture d'écran de Jenkins installé sur le serveur virtuel AWS EC2 avec le pointeur vers la page Web « Plugins ».

#### f. Création d'un pipeline Jenkins:

Maintenant que Jenkins fonctionne correctement, On peut commencer à créer le pipeline Jenkins. Pour créer un pipeline Jenkins, On peut créer un nouveau « projet Freestyle ». Pour créer un nouveau « projet Freestyle », On doit nous rendre sur le tableau de bord Jenkins et cliquer sur le bouton « New Item ».

The screenshot shows the Jenkins 'Enter an item name' dialog. A user has typed 'CI\_CD' into the input field. Below the input field, there are four project creation options: 'Freestyle project', 'Maven project', 'Pipeline', and 'Multi-configuration project'. Each option has a small icon and a brief description. The 'OK' button is visible at the bottom left of the dialog.

Enter an item name

CI\_CD

CI\_CD

**Freestyle project**  
Classic, general-purpose job type that checks out from up to one SCM, executes build steps, archiving artifacts and sending email notifications.

**Maven project**  
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration required.

**Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building workflows and/or organizing complex activities that do not easily fit in free-style job type.

**Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such as testing multiple environments or specific builds, etc.

OK

On fournit ensuite la **description** du pipeline.

## CI\_CD

CI CD pipeline

Plain text [Preview](#)

[Save](#)

[Disable Project](#)

[Permalinks](#)

### 2. Git, Github :

Maintenant que Jenkins s'exécute sur l'instance Ubuntu, vous pouvez configurer Git avec le pipeline, ce qui permet d'extraire le code le plus récent du référentiel Github de notre projet vers l'instance VM où Jenkins est installé.

#### a. Installation de Git :

```
vboxuser@jenkins:~$ sudo apt install git
[sudo] password for vboxuser:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libbice-dev libpthread-stubs0-dev libsm-dev libxi1-dev libxau-dev libxcb1-dev libxdmcp-dev libxt-dev openjdk-11-jdk-headless
  systemd-hwe-hwdb xiipproto-dev xorg-sgml-doctools xtrans-dev
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 458 not upgraded.
Need to get 4,147 kB of archives.
After this operation, 21.0 MB of additional disk space will be used.
Do you want to continue? [Y/n]
Get:1 http://fr.archive.ubuntu.com/ubuntu jammy/main amd64 liberror-perl all 0.17029-1 [26.5 kB]
Get:2 http://fr.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git-man all 1:2.34.1-1ubuntu1.10 [954 kB]
Get:3 http://fr.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git amd64 1:2.34.1-1ubuntu1.10 [3,166 kB]
Fetched 4,147 kB in 9s (475 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 205064 files and directories currently installed.)
Preparing to unpack .../liberror-perl_0.17029-1_all.deb ...

```

#### b. Installation du plugin Github :

Comme Git fonctionne correctement sur l'instance VM, nous pouvons maintenant intégrer Jenkins à Git. Pour démarrer cette intégration, installons le plugin Github « Pipeline For Blue Ocean » dans la section Jenkins Dashboard.

The screenshot shows the Jenkins 'Plugins' page. On the left, there's a sidebar with links for 'Updates', 'Available plugins' (which is selected and highlighted in grey), 'Installed plugins', 'Advanced settings', and 'Download progress'. The main area has a search bar at the top with 'Github' typed in. Below it is a table with columns 'Install', 'Name ↓', and 'Released'. The first row shows the 'GitHub Pipeline for Blue Ocean' plugin, version 1.27.11, released 1 mo 17 days ago. The second row shows the 'Jersey 2 API' plugin, version 2.42-147.va\_28a\_44603b\_d5, released 9 days 5 hr ago. The third row shows the 'Docker API' plugin, version 3.3.4-B6.v39b\_a\_5ede342c, released 4 mo 7 days ago. A yellow callout box highlights the 'Docker API' entry.

### c. Configuration du plug-in Github :

Maintenant que le plug-in GitHub Jenkins est installé, On doit le configurer pour qu'il intègre enfin Jenkins à Git.

#### Git installations

This screenshot shows the 'Git installations' configuration page. It features a 'Name' field containing 'Git' with a red arrow pointing to it. Below it is a 'Path to Git executable' field containing 'git' with another red arrow pointing to it. There's also a checkbox for 'Install automatically'. At the bottom is a 'Save' button.

### d. Integration de Git dans le pipeline :

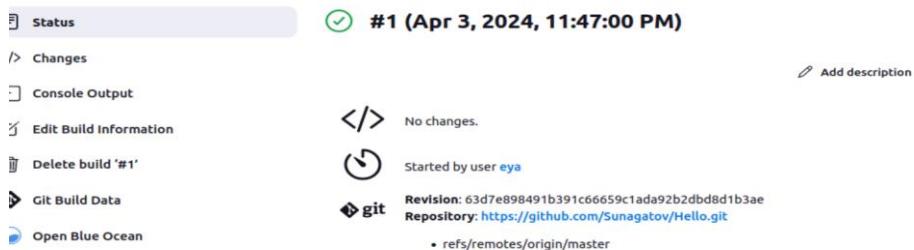
Maintenant que le plug-in Github Jenkins est installé et configuré, vous pouvez désormais l'utiliser dans votre pipeline. Cela permettra à notre pipeline d'extraire le code du projet à partir du référentiel GitHub spécifié:

The screenshot shows the configuration page for a Jenkins job named 'Pull\_Code\_from\_Github\_Job'. In the 'Configure' section, under 'Source Code Management', the 'Git' option is selected. The 'Repositories' section contains a single repository entry with a 'Repository URL' of 'https://github.com/Sunagatov>Hello.git'. There are also 'Credentials' and 'Advanced' sections. At the bottom are 'Save' and 'Apply' buttons.

### e. Test de l'intégration de Git dans le pipeline :

On peut désormais utiliser le pipeline mis à jour pour extraire un projet de Github. Pour ce faire, vous devez cliquer sur le bouton « Build Now ».

Ouvrez la première build à partir de l'historique des builds comme suit :



The screenshot shows the Jenkins Pipeline interface for build #1, which was triggered on April 3, 2024, at 11:47:00 PM. The build status is green with a checkmark. On the left, there's a sidebar with links for 'Status', 'Changes', 'Console Output', 'Edit Build Information', 'Delete build '#1', 'Git Build Data', and 'Open Blue Ocean'. The main content area displays the build configuration: a 'git' icon, the revision '63d7e898491b391c66659c1ada92b2dbd8d1b3ae', the repository URL 'https://github.com/Sunagatov/Hello.git', and a note that there are 'No changes.' Started by user 'eya'. A link to 'Add description' is also present.

On peut vérifier que le pipeline fonctionne bien.

Il suffit d'utiliser cette commande : `cd /var/lib/jenkins/workspace/CI_CD`

De cette façon, on peut voir que notre projet de Github a été extrait vers l'instance Ubuntu :

```
total 16
-rw-r--r-- 1 jenkins jenkins 354 Apr  3 23:47 Dockerfile
-rw-r--r-- 1 jenkins jenkins 1918 Apr  3 23:47 pom.xml
-rw-r--r-- 1 jenkins jenkins 183 Apr  3 23:47 README.md
drwxr-xr-x 3 jenkins jenkins 4096 Apr  3 23:47 src
```

### 3. Apache Maven :

Maintenant qu'on a intégré Git dans le pipeline, on peut l'améliorer davantage en incorporant Apache Maven, ce qui permet de générer et d'empaqueter notre projet.

#### a. Téléchargement de Apache Maven :

```
vboxuser@jenkins:/var/lib/jenkins/workspace/Pull_Code_From_Github_Job$ cd /opt
vboxuser@jenkins:/opt$ sudo wget https://dlcdn.apache.org/maven/maven-3/3.9.4/binaries/apache-maven-3.9.4-bin.tar.gz
[sudo] password for vboxuser:
--2024-04-04 00:01:13-- https://dlcdn.apache.org/maven/maven-3/3.9.4/binaries/apache-maven-3.9.4-bin.tar.gz
Resolving dlcdn.apache.org (dlcdn.apache.org)... 151.101.2.132, 2a04:4e42::644
Connecting to dlcdn.apache.org (dlcdn.apache.org)|151.101.2.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 9336368 (8.9M) [application/x-gzip]
Saving to: 'apache-maven-3.9.4-bin.tar.gz'

apache-maven-3.9.4-bin.tar.gz    100%[=====] 8.90M   722KB/s   in 19s
2024-04-04 00:01:33 (484 KB/s) - 'apache-maven-3.9.4-bin.tar.gz' saved [9336368/9336368]
```

#### b. Extraction de Apache Maven de l'archive :

```
vboxuser@jenkins:/opt$ sudo tar -xvzf apache-maven-*.tar.gz
apache-maven-3.9.4/README.txt
apache-maven-3.9.4/LICENSE
apache-maven-3.9.4/NOTICE
apache-maven-3.9.4/lib/
apache-maven-3.9.4/lib/aopalliance.license
apache-maven-3.9.4/lib/commons-cli.license
apache-maven-3.9.4/lib/commons-codec.license
apache-maven-3.9.4/lib/commons-lang3.license
apache-maven-3.9.4/lib/failureaccess.license
apache-maven-3.9.4/lib/guava.license
apache-maven-3.9.4/lib/guice.license
apache-maven-3.9.4/lib/httpclient.license
apache-maven-3.9.4/lib/httpcore.license
apache-maven-3.9.4/lib/jansi.license
apache-maven-3.9.4/lib/javax.annotation-api.license
apache-maven-3.9.4/lib/javax.inject.license
apache-maven-3.9.4/lib/jcl-over-slf4j.license
apache-maven-3.9.4/lib/org.eclipse.sisu.inject.license
apache-maven-3.9.4/lib/org.eclipse.sisu.plexus.license
apache-maven-3.9.4/lib/plexus-cipher.license
```

#### c. Ajout de **JAVA\_HOME** et **M2\_HOME** :

On Attribue le chemin d'accès à JDK17 pour **JAVA\_HOME** et le chemin d'accès au répertoire maven pour **M2\_HOME** variable.

```
GNU nano 6.2                                .bash_profile
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

M2_HOME=/opt/apache-maven-3.9.4
M2=/opt/apache-maven-3.9.4/bin
JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64/bin/java

PATH=$PATH:$HOME/bin:$M2:$M2_HOME:$JAVA_HOME

export PATH
```

```
vboxuser@jenkins:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/snap/bin:/home/vboxuser/bin:/opt
/apache-maven-3.9.4/bin:/opt/apache-maven-3.9.4:/usr/lib/jvm/java-17-openjdk-amd64/bin/java
```

Pour vérifier que Maven a été installé avec succès, on peut exécuter cette commande :

```
vboxuser@jenkins:~$ mvn -v
Apache Maven 3.9.4 (dfbb324ad4a7c8fb0bf182e6d91b0ae20e3d2dd9)
Maven home: /opt/apache-maven-3.9.4
Java version: 17.0.10, vendor: Private Build, runtime: /usr/lib/jvm/java-17-openjdk-amd64
Default locale: en_NG, platform encoding: UTF-8
OS name: "linux", version: "6.5.0-26-generic", arch: "amd64", family: "unix"
```

#### d. Installation du plug-in Maven Integration :

Dashboard > Manage Jenkins > Plugins

**Plugins**

- Updates
- Available plugins**
- Installed plugins
- Advanced settings
- Download progress

Search: maven

Install	Name	Released
<input checked="" type="checkbox"/>	Maven Integration 3.23	Build Tools 8 mo 2 days ago
<input type="checkbox"/>	Config File Provider 968.ve1ca_eb_913fb8c	Groovy-related External Site/Tool Integrations Maven 2 mo 10 days ago
<input type="checkbox"/>	Jira 3.13	External Site/Tool Integrations Maven jira 20 days ago

#### e. Configuration du plug-in Maven Integration :

JDK Installations

Add JDK

Name: java-17

JAVA\_HOME: /usr/lib/jvm/java-17-openjdk-amd64

/usr/lib/jvm/java-17-openjdk-amd64 doesn't look like a JDK directory

Install automatically

Add Maven

Name: apache-maven-3.9.4

MAVEN\_HOME: /opt/apache-maven-3.9.4

Install automatically

#### f. Intégrer Apache Maven dans le pipeline :

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps**
- Post-build Actions

Build Steps

Invoke top-level Maven targets

Maven Version: apache-maven-3.9.4

Goals: clean package

POM: pom.xml

Save Apply

#### g. Test de l'intégration d'Apache Maven dans le pipeline :

On peut maintenant utiliser notre pipeline mis à jour pour générer notre projet Github. Pour ce faire, vous devez cliquer sur le bouton « Build Now »

Si vous ouvrez votre terminal Ubuntu, on peut vérifier que le pipeline fonctionne bien.

Il suffit d'utiliser cette commande : `cd/var/lib/jenkins/workspace/CI_CD/target`

De cette façon, On peut voir l'artefact JAR, indiquant la génération réussie de notre projet à partir de GitHub

```
vboxuser@jenkins:/var/lib/jenkins/workspace/CI_CD/target$ ls -la
total 19056
drwxr-xr-x 6 jenkins jenkins 4096 Apr  4 01:24 .
drwxr-xr-x 5 jenkins jenkins 4096 Apr  4 01:23 ..
drwxr-xr-x 3 jenkins jenkins 4096 Apr  4 01:23 classes
drwxr-xr-x 3 jenkins jenkins 4096 Apr  4 01:23 generated-sources
-rw-r--r-- 1 jenkins jenkins 19481477 Apr  4 01:24 hello-0.0.1-SNAPSHOT.jar
-rw-r--r-- 1 jenkins jenkins 3699 Apr  4 01:24 hello-0.0.1-SNAPSHOT.jar.original
drwxr-xr-x 2 jenkins jenkins 4096 Apr  4 01:24 maven-archiver
drwxr-xr-x 3 jenkins jenkins 4096 Apr  4 01:23 maven-status
```

## 4. Docker :

### a. Installation de docker :

```
vboxuser@jenkins:~$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libice-dev libpthread-stubs0-dev libsm-dev libxi-dev libxau-dev libxcb1-dev libxdmcp-dev libxt-dev openjdk-11-jdk-headless
  systemd-hwe-hwdb x11proto-dev xorg-sgml-doctools xtrans-dev
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  bridge-utils containerd pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools btrfs-progs cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd docker.io pigz runc ubuntu-fan
0 upgraded, 6 newly installed, 0 to remove and 458 not upgraded.
Need to get 69.4 MB of archives.
After this operation, 266 MB of additional disk space will be used.
Do you want to continue? [Y/n]
Get:1 http://fr.archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1 [63.6 kB]
Get:2 http://fr.archive.ubuntu.com/ubuntu jammy/main amd64 bridge-utils amd64 1.7-1ubuntu3 [34.4 kB]
Get:3 http://fr.archive.ubuntu.com/ubuntu jammy-updates/main amd64 runc amd64 1.1.7-0ubuntu1-22.04.2 [4,267 kB]
Get:4 http://fr.archive.ubuntu.com/ubuntu jammy-updates/main amd64 containerd amd64 1.7.2-0ubuntu1-22.04.1 [36.0 MB]
Get:5 http://fr.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 docker.io amd64 24.0.5-0ubuntu1-22.04.1 [28.9 MB]
Get:6 http://fr.archive.ubuntu.com/ubuntu jammy/universe amd64 ubuntu-fan all 0.12.16 [35.2 kB]
Fetched 69.4 MB in 1min 30s (774 kB/s)
Preconfiguring packages...
Selecting previously unselected package pigz.
(Reading database ... 266049 files and directories currently installed.)
Preparing to unpack .../0-pigz_2.6-1_amd64.deb ...
Unpacking pigz (2.6-1) ...
Selecting previously unselected package bridge-utils.
Preparing to unpack .../1-bridge-utils_1.7-1ubuntu3_amd64.deb ...
Unpacking bridge-utils (1.7-1ubuntu3) ...
Selecting previously unselected package runc.
```

### b. Démarrage de Docker :

```
Processing triggers for man-db (2.10.2-1) ...
vboxuser@jenkins:~$ sudo systemctl enable docker
vboxuser@jenkins:~$ sudo systemctl start docker
vboxuser@jenkins:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2024-04-04 01:53:14 WAT; 2min 52s ago
     Docs: https://docs.docker.com
 Main PID: 25247 (dockerd)
   Tasks: 10
   Memory: 27.5M
      CPU: 1.041s
      CGroup: /system.slice/docker.service
              └─25247 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Apr  04 01:53:12 jenkins systemd[1]: Starting Docker Application Container Engine...
Apr  04 01:53:12 jenkins dockerd[25247]: time="2024-04-04T01:53:12.864439284+01:00" level=info msg="Starting up"
Apr  04 01:53:12 jenkins dockerd[25247]: time="2024-04-04T01:53:12.868001233+01:00" level=info msg="detected 127.0.0.53 nameserver, >
Apr  04 01:53:13 jenkins dockerd[25247]: time="2024-04-04T01:53:13.300432403+01:00" level=info msg="Loading containers: start."
Apr  04 01:53:14 jenkins dockerd[25247]: time="2024-04-04T01:53:14.092301240+01:00" level=info msg="Loading containers: done."
Apr  04 01:53:14 jenkins dockerd[25247]: time="2024-04-04T01:53:14.238283973+01:00" level=info msg="Docker daemon" commit="24.0.5-0u>
Apr  04 01:53:14 jenkins dockerd[25247]: time="2024-04-04T01:53:14.238488170+01:00" level=info msg="Daemon has completed initializ>
Apr  04 01:53:14 jenkins dockerd[25247]: time="2024-04-04T01:53:14.361641473+01:00" level=info msg="API listen on /run/docker.sock"
Apr  04 01:53:14 jenkins systemd[1]: Started Docker Application Container Engine.
lines 1-21/21 (END)
```

c. Création d'un Dockerfile :

```
GNU nano 6.2                               Dockerfile
ARG HOME_APP=/opt/app

FROM maven:3.8.3-openjdk-17 as maven_build
ARG HOME_APP
WORKDIR $HOME_APP
ADD pom.xml .
RUN mvn verify --fail-never
ADD .
RUN mvn package

FROM eclipse-temurin:17-jre-jammy
ARG HOME_APP
WORKDIR $HOME_APP
COPY --from=maven_build $HOME_APP/target/hello-0.0.1-SNAPSHOT.jar .
ENTRYPOINT ["java", "-jar", "hello-0.0.1-SNAPSHOT.jar"]
```

d. Connection à Docker :

Avant de commencer le test de l'environemnt, on s'assure de nous authentifier sur Dockerhub.

```
vboxuser@jenkins:/opt$ sudo docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.co
m to create one.
Username: ayouta
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store
Login Succeeded
```

e. Test de l'environnement Docker et de Dockerfile :

Une fois notre connexion réussie à Dockerhub terminée, on est maintenant prêt à commencer à tester le fichier Dockerfile qu'on a préparé.

On commence par exécuter cette commande pour créer une image docker :

```
vboxuser@jenkins:/var/lib/jenkins/workspace/CI_CD$ sudo docker build -t hello:latest .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
Install the buildx component to build images with BuildKit:
https://docs.docker.com/go/buildx/

Sending build context to Docker daemon 13.82kB
Step 1/13 : ARG HOME_APP=/opt/app
Step 2/13 : FROM maven:3.8.3-openjdk-17 as maven_build
3.8.3-openjdk-17: Pulling from library/maven
28587b6e6475: Pull complete
b1655352c888: Pull complete
1f9646f00e96: Pull complete
6cd514e60c46: Pull complete
0ab2a7ea62ca: Pull complete
d72072d9f9d3: Pull complete
9eca236510fe: Pull complete
Digest: sha256:8a66581a077762c8752a9f64f73cd8c59e9c4446eb810417119e0436b075931
Status: Downloaded newer image for maven:3.8.3-openjdk-17
--> 0b9ddcb8259e
Step 3/13 : ARG HOME_APP
--> Running in eb09c40c77ae
Removing intermediate container eb09c40c77ae
--> 1ff249daa552
Step 4/13 : WORKDIR $HOME_APP
--> Running in 6d9fc500bb1c
Removing intermediate container 6d9fc500bb1c
--> 27a5268b2473
Step 5/13 : ADD pom.xml .
--> 25a221867
```

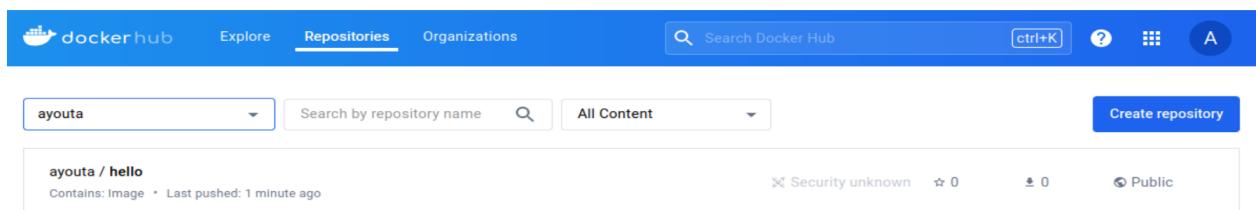
Ensuite, on exécute la commande suivante pour établir une balise qui facilitera le téléchargement de l'image vers Dockerhub :

```
vboxuser@jenkins:/var/lib/jenkins/workspace/CI_CD$ sudo docker tag hello:latest ayouta/hello:latest
[sudo] password for vboxuser:
```

Enfin, procédez à l'envoi de l'image Docker vers Dockerhub via l'exécution de cette commande :

```
vboxuser@jenkins:/var/lib/jenkins/workspace/CI_CD$ sudo docker push ayouta/hello:latest
The push refers to repository [docker.io/ayouta/hello]
a4d301b80589: Pushed
b3ad2aab4039: Pushed
5fc689bf1885: Mounted from library/eclipse-temurin
64a82ca504d4: Mounted from library/eclipse-temurin
e030219377cf: Mounted from library/eclipse-temurin
90c4c6bb3fef: Mounted from library/eclipse-temurin
5498e8c22f69: Mounted from library/eclipse-temurin
latest: digest: sha256:08bb190dfd140224b797dcce9bda6b8071218c71409958ed216de9a502681d2b size: 1786
vboxuser@jenkins:/var/lib/jenkins/workspace/CI_CD$
```

En suivant ces étapes, on accède à votre compte Dockerhub pour vérifier si on peut voir une nouvelle image ou non. On doit maintenant observer que l'image a bien été ajoutée. Ce résultat confirme que l'installation de l'environnement Docker a réussi et que le fichier Dockerfile est correct.



## 5. Ansible :

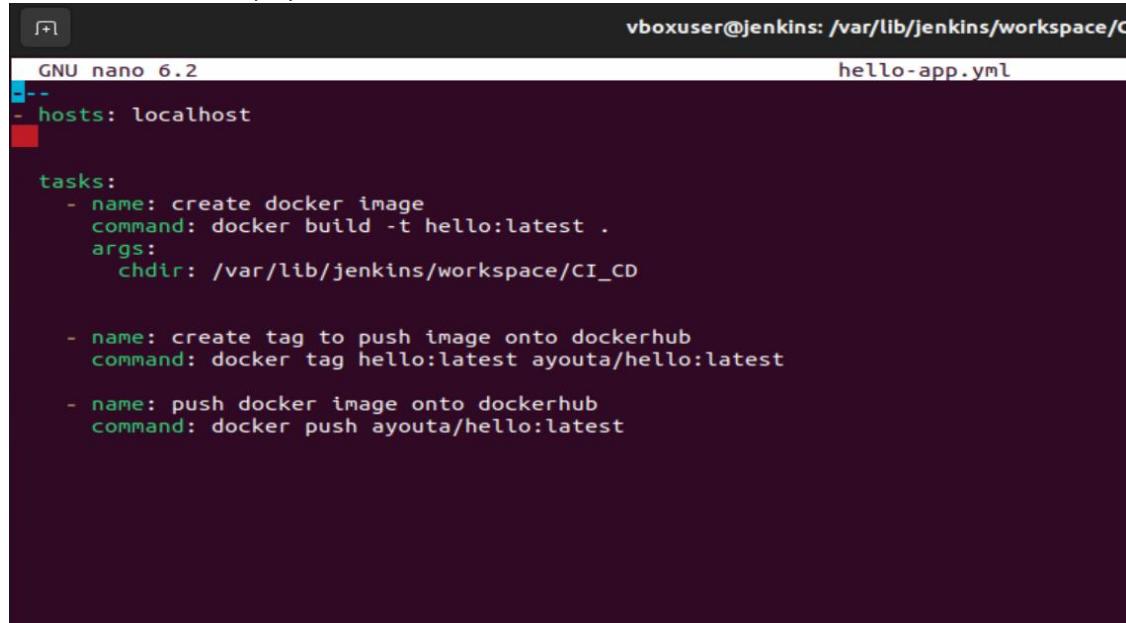
### a. Installation de Ansible :

```
vboxuser@jenkins:/opt/docker$ ansible --version
ansible 2.10.8
  config file = None
  configured module search path = ['/home/vboxuser/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0]
```

### b. Création d'un playbook Ansible pour les tâches Docker :

Maintenant qu'Ansible est installé et prêt à l'emploi, on peut créer un playbook Ansible pour notre pipeline. Ce playbook permettra à Ansible de créer et d'envoyer une nouvelle image Docker à Dockerhub.

Voici le contenu de ce playbook :



```
vboxuser@jenkins: /var/lib/jenkins/workspace/C
GNU nano 6.2                                     hello-app.yml
--+
- hosts: localhost

tasks:
  - name: create docker image
    command: docker build -t hello:latest .
    args:
      chdir: /var/lib/jenkins/workspace/CI_CD

  - name: create tag to push image onto dockerhub
    command: docker tag hello:latest ayouta/hello:latest

  - name: push docker image onto dockerhub
    command: docker push ayouta/hello:latest
```

Ensuite on exécute ce playbook comme suit.  
Une fois l'opération terminée, On voit le résultat de l'exécution réussie du playbook Ansible

```
vboxuser@jenkins:/var/lib/jenkins/workspace/CI_CD$ sudo nano hello-app.yml
vboxuser@jenkins:/var/lib/jenkins/workspace/CI_CD$ sudo -u vboxuser ansible-playbook hello-app.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'
PLAY [localhost] ****
TASK [Gathering Facts] ****
ok: [localhost]
TASK [create docker image] ****
changed: [localhost]
TASK [create tag to push image onto dockerhub] ****
changed: [localhost]
TASK [push docker image onto dockerhub] ****
changed: [localhost]
PLAY RECAP ****
localhost          : ok=4    changed=3   unreachable=0   failed=0    skipped=0   rescued=0   ignored=0
```

De plus, en visitant le compte Dockerhub on peut vérifier qu'une nouvelle image est maintenant visible.

The screenshot shows the Dockerhub interface. At the top, there are dropdown menus for 'ayouta' (repository name), a search bar ('Search by repository name') with a magnifying glass icon, and a dropdown for 'All Content'. Below this, a card displays the repository 'ayouta / hello'. It shows 'Contains: Image' and 'Last pushed: 2 minutes ago'. To the right, there is a 'Security unknown' badge and a star rating of '0'. The card has a light gray background with dark gray text.

#### c. Intégration des tâches Ansible Docker dans le pipeline :

On ajoute à notre pipeline CI\_CD une autre tâche via Jenkins pour exécutant la commande shell ansible-playbook appropriée ainsi que les paramètres requis, en choisissant l'option « **Execute Shell** » comme « **Build Step** ».

The screenshot shows the Jenkins Pipeline configuration screen. At the top, there is a 'Goals' section with a dropdown containing 'clean package'. Below it is an 'Advanced' dropdown set to 'Edited'. The main area shows a step titled 'Execute shell'. Under 'Command', it says 'See the list of available environment variables' and contains the command 'ansible-playbook /var/lib/jenkins/workspace/CI\_CD/hello-app.yml'. At the bottom, there are 'Save' and 'Apply' buttons.

#### d. Test de Ansible Docker dans le pipeline :

Après avoir enregistré la tâche dans le pipeline, on clique sur « Build Now » et on vérifie le journal de la console. On voit bien le succès du build de tout le pipeline

```
http://172.17.0.1:8080/jenkins/job/CI_CD/26/console
```

```
Login Succeeded
+ ansible-playbook /var/lib/jenkins/workspace/CI_CD/hello-app.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that
the implicit localhost does not match 'all'

PLAY [localhost] ****
TASK [Gathering Facts] ****
ok: [localhost]

TASK [create docker image] ****
changed: [localhost]

TASK [create tag to push image onto dockerhub] ****
changed: [localhost]

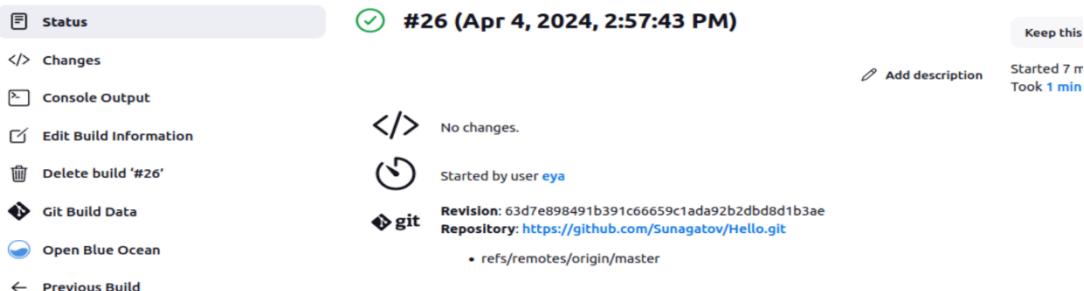
TASK [push docker image onto dockerhub] ****
changed: [localhost]

PLAY RECAP ****
localhost : ok=4    changed=3    unreachable=0    failed=0    skipped=0
rescued=0  ignored=0

Finished: SUCCESS
```



The Jenkins UI shows build #26 details. The build status is green with a checkmark, indicating success. The build was started by user 'eya' 7 minutes ago and took 1 minute. The build summary shows 4 successful tasks, 3 changed files, and 0 errors. The 'Console Output' section is collapsed.



The Jenkins UI shows build #26 details expanded. The build status is green with a checkmark, indicating success. The build was started by user 'eya' 7 minutes ago and took 1 minute. The build summary shows 4 successful tasks, 3 changed files, and 0 errors. The 'Console Output' section is expanded, showing the log output:

```
</> No changes.  
Started by user eya  
Revision: 63d7e898491b391c66659c1ada92b2dbd8d1b3ae  
Repository: https://github.com/Sunagatov/Hello.git  
* refs/remotes/origin/master
```

Other build-related links are visible on the left: Status, Changes, Console Output, Edit Build Information, Delete build '#26', Git Build Data, Open Blue Ocean, and Previous Build.

## 6. Kubernetes :

### a. Installation de kubectl :

```
vboxuser@jenkins:/opt/docker$ sudo curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl
  % Total    % Received % Xferd  Average Speed   Time   Time     Current
          Dload  Upload Total Spent   Left Speed
100 47.4M  100 47.4M    0      0  371k   0  0:02:11  0:02:11 --:--:--  687k
vboxuser@jenkins:/opt/docker$ sudo chmod +x kubectl
vboxuser@jenkins:/opt/docker$ sudo mv kubectl /usr/local/bin/
vboxuser@jenkins:/opt/docker$ kubectl version --output=yaml
clientVersion:
  buildDate: "2024-03-15T00:08:19Z"
  compiler: gc
  gitCommit: 6813625b7cd706db5bc7388921be03071e1a492d
  gitTreeState: clean
  gitVersion: v1.29.3
  goVersion: go1.21.8
  major: "1"
  minor: "29"
  platform: linux/amd64
kustomizeVersion: v5.0.4-0.20230601165947-6ce0bf390ce3

Error from server (Forbidden): <html><head><meta http-equiv='refresh' content='1;url=/login?from=%2Fversion%3Ftimeout%3D32s' /><script id='redirect' data-redirect-url='/Login?from=%2Fversion%3Ftimeout%3D32s' src='/static/6efa291e/scripts/redirect.js'></script></head><body style='background-color:white; color:white;'>
Authentication required
<!--
-->

</body></html>
```

### b. Installation de minikube et son démarrage :

```
vboxuser@jenkins:/opt/docker$ sudo install minikube-linux-amd64 /usr/local/bin/minikube
vboxuser@jenkins:/opt/docker$ minikube start --driver=<driver>
bash: syntax error near unexpected token `newline'
vboxuser@jenkins:/opt/docker$ minikube start
  minikube v1.32.0 on Ubuntu 22.04 (vbox/amd64)
  Automatically selected the docker driver. Other choices: none, ssh
  Using Docker driver with root privileges
  Starting control plane node minikube in cluster minikube
  Pulling base image ...
  Downloading Kubernetes v1.28.3 preload ...
  > preloaded-images-k8s-v18-v1...: 403.35 MiB / 403.35 MiB 100.00% 246.36
  > gcr.io/k8s-minikube/kicbase...: 453.90 MiB / 453.90 MiB 100.00% 254.48
  Creating docker container (CPUs=2, Memory=2700MB) ...
  Preparing Kubernetes v1.28.3 on Docker 24.0.7 ...
  ■ Generating certificates and keys ...
  ■ Booting up control plane ...
  ■ Configuring RBAC rules ...
  Configuring bridge CNI (Container Networking Interface) ...
  Verifying Kubernetes components...
  ■ Using image gcr.io/k8s-minikube/storage-provisioner:v5
  ■ Enabled addons: storage-provisioner, default-storageclass
  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
vboxuser@jenkins:/opt/docker$
```

```
Done! kubectl is now configured to use minikube cluster and default namespace
vboxuser@jenkins:/opt/docker$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

### c. Création un fichier yaml de déploiement Kubernetes :

```
GNU nano 6.2                                     hello-app-deployment.yaml *
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ayouta-hello-app
  labels:
    app: hello-app

spec:
  replicas: 2
  selector:
    matchLabels:
      app: hello-app

  template:
    metadata:
      labels:
        app: hello-app
    spec:
      containers:
        - name: hello-app
          image: ayouta/hello
          imagePullPolicy: Always
          ports:
            - containerPort: 8080
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
```

d. Création d'un fichier yaml de service Kubernetes :

```
GNU nano 6.2                                     hello-app-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: ayouta-hello-app-service
  labels:
    app: hello-app
spec:
  selector:
    app: hello-app

  ports:
    - port: 8081
      targetPort: 8080
      nodePort: 30000
  type: NodePort
```

e. Tester le cluster Kubernetes à l'aide de kubectl

Une fois que votre cluster Kubernetes minikube est installé et configuré avec succès, et que nos fichiers de service et de déploiement Kubernetes sont prêts,

on utilisez la commande suivante pour appliquer la configuration du déploiement : **kubectl apply -f hello-app-deployment.yaml**

Cela créera un déploiement avec le nombre spécifié de réplicas et une stratégie de mise à jour propagée, garantissant la disponibilité et la facilité de gestion de votre application.

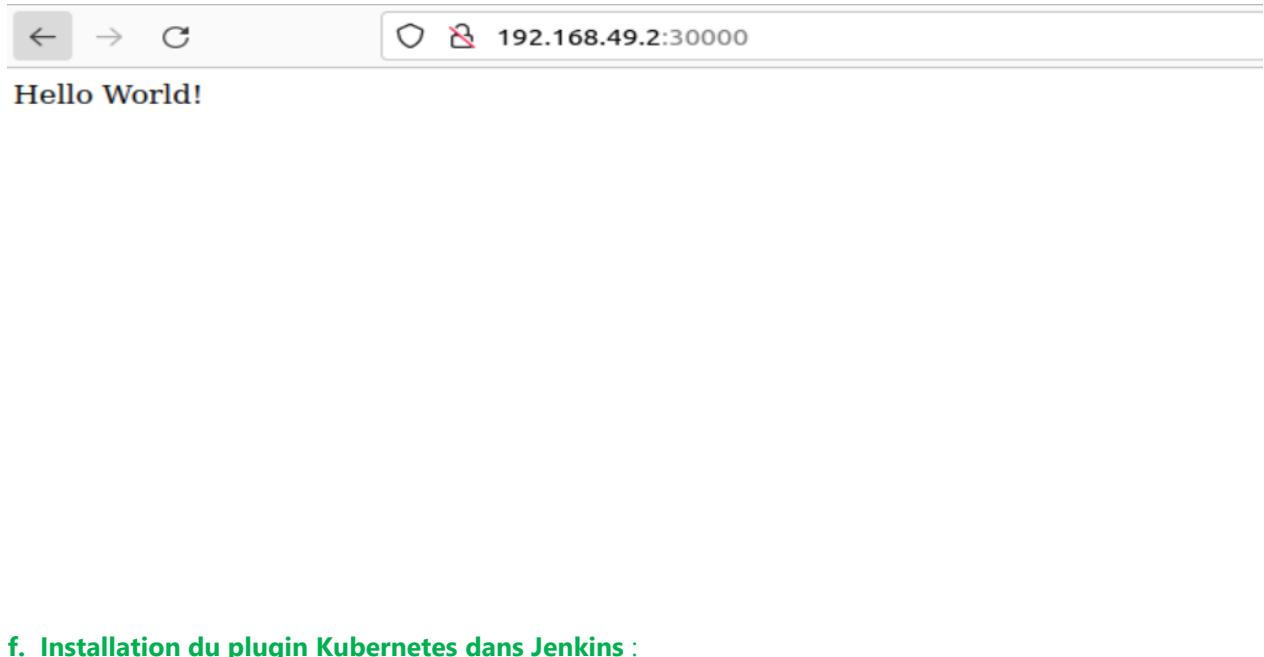
```
vboxuser@jenkins:/$ kubectl apply -f hello-app-deployment.yaml
deployment.apps/ayouta-hello-app created
vboxuser@jenkins:/$
```

Ensuite, appliquez la configuration du service à l'aide de cette commande : `kubectl apply -f hello-app-service.yaml`  
Cela mettra en place un service de type NodePort, exposant votre application à Internet.  
On surveille ensuite l'état de notre service :

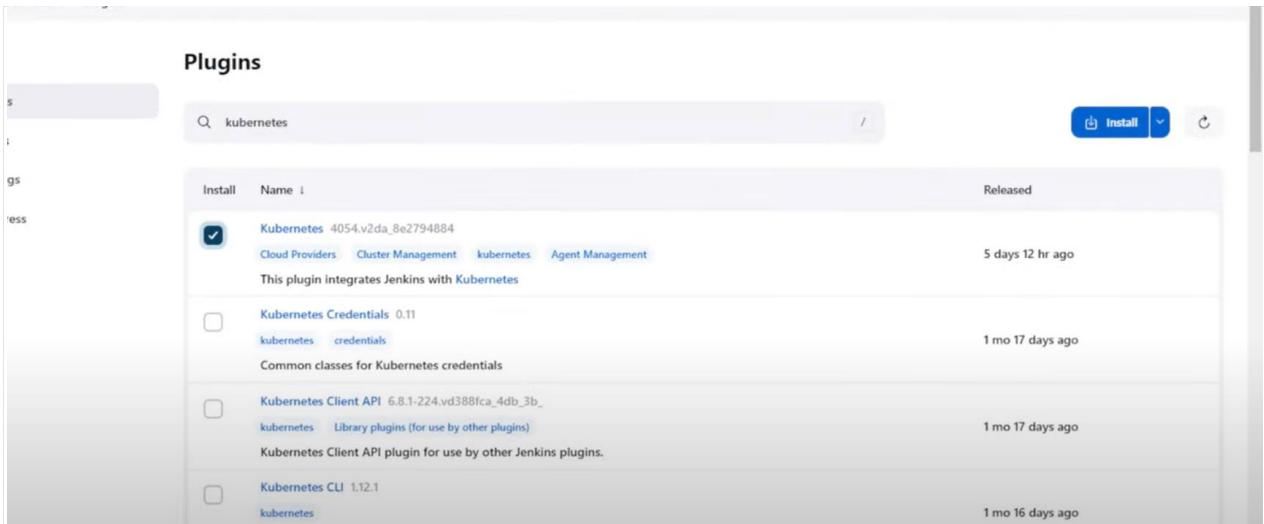
```
vboxuser@jenkins:/$ kubectl get service ayouta-hello-app-service
NAME                  TYPE        CLUSTER-IP      EXTERNAL-IP    PORT(S)         AGE
ayouta-hello-app-service   NodePort    10.105.18.118  <none>        8081:30000/TCP  16s
```

À l'aide d'un navigateur Web, on entre l'adresse IP du serveur minikube suivie de :30000. Après un bref instant, la page se chargera et affichera le message « HelloWorld ».

```
vboxuser@jenkins:/$ minikube ip
192.168.49.2
vboxuser@jenkins:/$
```



#### f. Installation du plugin Kubernetes dans Jenkins :



### g. Configuration du plugin Kubernetes :

Pour configurer l'intégration de Jenkins avec Kubernetes,

On retourne à la page principale de configuration de Jenkins, en sélectionnant "Manage Jenkins" (Gérer Jenkins), puis "Configure System" (Configurer le système). Faites défiler jusqu'à la section "Cloud" et cliquez sur "Add a new cloud" (Ajouter un nouveau cloud). Choisissez "Kubernetes" dans le menu déroulant.

Dans la configuration de Kubernetes, vous doit renseigner plusieurs détails notamment :

- Name : un nom unique à cette configuration Kubernetes pour la distinguer d'autres configurations.
- Kubernetes URL : l'URL de notre cluster Kubernetes.
- Kubernetes Namespace : Indiquez le namespace où Jenkins doit déployer les conteneurs.
- Credentials : les informations d'identification nécessaires pour accéder à votre cluster Kubernetes.
- Jenkins URL : Entrez l'URL publique de votre instance Jenkins.

#### Cloud kubernetes Configuration

Name ?  
kubernetes

Kubernetes Cloud details ^      Edited

Kubernetes URL ?  
https://192.168.49.2:8443

Use Jenkins Proxy ?

Kubernetes server certificate key ?

Disable https certificate check ?



#### h. Création d'un nouveau playbook Ansible pour les tâches Kubernetes :

Maintenant que Kubernetes est configuré et prêt à l'emploi, on peut créer un nouveau playbook Ansible avec des tâches Kubernetes pour votre pipeline. Ce playbook permettra à Ansible d'exécuter votre application sur le cluster Kubernetes avec des commandes kubectl :

```
GNU nano 6.2                                     kubernetes-hello-app.yml *
```

```
---  
- hosts: localhost  
  
  tasks:  
    - name: deploy regapp on kubernetes  
      command: kubectl apply -f hello-app-deployment.yaml  
  
    - name: create service for regapp  
      command: kubectl apply -f hello-app-service.yaml  
  
    - name: update deployment with new pods if image updated in docker hub  
      command: kubectl rollout restart deployment.apps/ayouta-hello-app
```

e. Création d'un Job pour le déploiement avec Kubernetes en utilisant Ansible :

On choisit comme type de job « Freestyle Project »



The screenshot shows the Jenkins interface for creating a new item. A search bar at the top contains the text "CD\_Job". Below it, a list of project types is shown:

- Freestyle project**: Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, follows branching, archiving artifacts and sending email notifications.
- Maven project**: Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- Pipeline**: Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (for workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**: Suitable for projects that need a large number of different configurations, such as testing on multiple environments or specific builds, etc.

An "OK" button is visible at the bottom of the list.

f. Ajout d'une tâche Ansible Pour le déploiement sur Kubernetes :

On ajoute une tâche de type "Invoke Ansible Playbook" dans Jenkins qui permet d'automatiser le déploiement sur Kubernetes en utilisant Ansible, en exécutant un playbook Ansible qui contient les instructions nécessaires pour déployer le projet sur le cluster Kubernetes.

On ne doit également pas oublier de faire d'un « Build Now » de ce job de déploiement après avoir enregistrer cette tâche.

## Build Steps

**Invoke Ansible Playbook**

**Ansible installation**

ansible

**Playbook path** ?

/var/lib/jenkins/workspace/CD\_Job/kubernetes-hello-app.yml

**Inventory**

Do not specify Inventory

File or host list

Inline content

**Host subset** ?

Save      Apply

g. Ajout du Job CD\_Job comme Post Build Action pour le pipeline CI\_CD :

Nous ajoutons l'entrée "CD-Job" comme projet à construire, assurant ainsi que notre pipeline CD ne sera déclenché que lorsque la version du build sera considérée comme stable, assurant ainsi une gestion efficace du déploiement

## Post-build Actions

**Build other projects** ?

**Projects to build**

CD\_Job

Trigger only if build is stable

Trigger even if the build is unstable

Trigger even if the build fails

Add post-build action ▾

Save      Apply

h. Test de la version finale du pipeline :

Lorqu'on clique sur « Build Now » dans le « CI\_CD » pipeline, on remarque que le output log a été déclenché dans les journaux de la console de compilation et que l'état final est marqué comme SUCCESS :

```
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time:  4.194 s
[INFO] Finished at: 2024-04-04T22:34:17Z
[INFO] -----
          Connecting from host      [localhost]
          Connecting with configuration  [localhost]  ...
EXEC: completed after 14,009 ms
Disconnecting configuration  [localhost]  ...
Transferred 1 file(s)
Triggering a new build of CI_CD
Finished: SUCCESS
```