Mục lục

[Setup k8 cluster 3](#_Toc12017334)

[Chuẩn bị sever 3](#_Toc12017335)

[Remove cluster cũ (optional) 4](#_Toc12017336)

[Install cluster mới 4](#_Toc12017337)

[Test cluster mới 5](#_Toc12017338)

[Setup local registry 5](#_Toc12017339)

[Install service 5](#_Toc12017340)

[Test service 6](#_Toc12017341)

[Thêm forward domain cho DNS service của k8s 7](#_Toc12017342)

[Sửa Corefile của coredns 7](#_Toc12017343)

[Test dns 8](#_Toc12017344)

[Tìm hiểu thêm về hostname, DNS trên k8s 8](#_Toc12017345)

[Setup nginx ingress cho k8s services 8](#_Toc12017346)

[Install nginx ingress 8](#_Toc12017347)

[Test ingress 8](#_Toc12017348)

[Một số lệnh hay dùng 9](#_Toc12017349)

[Viết yaml 9](#_Toc12017350)

[Tạo pod, deployment, service 9](#_Toc12017351)

[Gán label cho pod 9](#_Toc12017352)

[Setup NFS persistent volume 9](#_Toc12017353)

[Run NFS pod 9](#_Toc12017354)

[Tạo PV 11](#_Toc12017355)

[Test mount PV 11](#_Toc12017356)

[Setup alluxio cluster trên k8s 13](#_Toc12017357)

[Build alluxio docker 13](#_Toc12017358)

[Copy alluxio-conf đến persistent volume 17](#_Toc12017359)

[Copy keytab đến secret 17](#_Toc12017360)

[Run alluxio cluster 17](#_Toc12017361)

[Mount HDFS 21](#_Toc12017362)

[Stop alluxio cluster (optional) 21](#_Toc12017363)

[Setup Spark cluster trên k8s 21](#_Toc12017364)

[Build spark docker 21](#_Toc12017365)

[Run spark cluster 23](#_Toc12017366)

[Test spark 27](#_Toc12017367)

[Stop spark cluster (optional) 27](#_Toc12017368)

[Setup Jupyter 27](#_Toc12017369)

[Build jupyter docker 27](#_Toc12017370)

[Copy hadoop-conf đến persistent volume 31](#_Toc12017371)

[Copy keytab đến secret 31](#_Toc12017372)

[Start jupyter 31](#_Toc12017373)

[Test jupyter 33](#_Toc12017374)

[Stop jupyter (optional) 35](#_Toc12017375)

[Setup Zeppelin 35](#_Toc12017376)

[Build zeppelin docker 35](#_Toc12017377)

[Copy hadoop-conf đến persistent volume 45](#_Toc12017378)

[Copy keytab đến secret 45](#_Toc12017379)

[Start zeppelin 45](#_Toc12017380)

[Test zepplin 47](#_Toc12017381)

[Stop zeppelin (optional) 48](#_Toc12017382)

[Setup airflow 48](#_Toc12017383)

[Build airflow docker 48](#_Toc12017384)

[Copy hadoop-conf đến persistent volume 76](#_Toc12017385)

[Copy keytab đến secret 76](#_Toc12017386)

[Start airflow 76](#_Toc12017387)

[Test airflow 77](#_Toc12017388)

[Stop airflow (optional) 78](#_Toc12017389)

[Quản lý cấp phát resource trên k8s 78](#_Toc12017390)

[Quản lý memory & cpu của pod 78](#_Toc12017391)

[Autoscale 79](#_Toc12017392)

[Cronjob 79](#_Toc12017393)

[Monitor 79](#_Toc12017394)

[kubectl top 79](#_Toc12017395)

[Check readiness & liveness probe 79](#_Toc12017396)

[Dùng Kubernetes dashboard 79](#_Toc12017397)

[Monitor performance của k8s và alert bằng Prometheus & Grafana 81](#_Toc12017398)

[Mô hình: 81](#_Toc12017399)

[Deploy: 81](#_Toc12017400)

[PoC prometheus, xem dashboard trên grafana, alert bằng alertmanager, gmail 83](#_Toc12017401)

[Start prometheus (port 9090) 83](#_Toc12017402)

[Start các node exporter (port 9100) 84](#_Toc12017403)

[Start grafana (port 3000) 84](#_Toc12017404)

[Start alertmanager (port 9093) 84](#_Toc12017405)

[Centralize POD logs, k8s events bằng EFK stack 86](#_Toc12017406)

[Mô hình 86](#_Toc12017407)

[Deploy: 86](#_Toc12017408)

[Security 99](#_Toc12017409)

[Benchmark security 99](#_Toc12017410)

[Update authentication 100](#_Toc12017411)

[Disable anonymous login 100](#_Toc12017412)

[Authen user bằng X509 cert 100](#_Toc12017413)

[Disable auto-mout “default” service account khi tạo pod 101](#_Toc12017414)

[Setup Pod Policy 101](#_Toc12017415)

[Enable PodSecurityPolicy 101](#_Toc12017416)

[Nguyên tắc set policy 101](#_Toc12017417)

[Setup Network Policy 103](#_Toc12017418)

[Nguyên tắc set policy 103](#_Toc12017419)

[Demo 103](#_Toc12017420)

[Enable audit logging cho kube-apiserver 108](#_Toc12017421)

[Phát hiện các security event bất thường bằng Falco 108](#_Toc12017422)

[Giải thích: 108](#_Toc12017423)

[Setup: 108](#_Toc12017424)

[Test: 109](#_Toc12017425)

# Setup k8 cluster

master: h1.spark.org (192.168.2.101)

worker 1: h2.spark.org (.102)

worker 2: h3.spark.org (.1030

## Chuẩn bị sever

Cài CentOS 7

Disable swap

sudo swapoff -a

sudo sed -i '/ swap / s/^\(.\*\)$/#\1/g' /etc/fstab

Cài đặt kubelet và docker-ce packages

sudo vi /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86\_64

enabled=1

gpgcheck=1

repo\_gpgcheck=1

gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg

sudo yum -y install kubeadm docker

#sudo systemctl restart docker && systemctl enable docker

#sudo systemctl restart kubelet && systemctl enable kubelet

Force docker dùng system

sudo vi /etc/docker/daemon.json

{

"exec-opts": ["native.cgroupdriver=systemd"]

}

#sudo systemctl restart docker

sudo systemctl enable docker.service

sudo systemctl enable kubelet.service

sudo vi /etc/sysconfig/kubelet

KUBELET\_EXTRA\_ARGS=--runtime-cgroups=/systemd/system.slice --kubelet-cgroups=/systemd/system.slice

sudo systemctl start docker.service

sudo systemctl start kubelet.service

sudo vi /etc/sysctl.conf

net.bridge.bridge-nf-call-iptables = 1

sudo sysctl -p

Config kubelet

sudo vi /etc/sysconfig/kubelet

KUBELET\_EXTRA\_ARGS=--node-ip=192.168.2.101

#192.168.2.102, …

## Remove cluster cũ (optional)

master>

kubectl delete node h2.spark.org

kubectl delete node h3.spark.org

workers 1,2>

sudo kubeadm reset

master>

sudo kubeadm reset

## Install cluster mới

master>

sudo kubeadm init --apiserver-advertise-address 192.168.2.101 --pod-network-cidr 10.1.0.0/16

…

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

kubectl apply -f https://raw.githubusercontent.com/cloudnativelabs/kube-router/master/daemonset/kubeadm-kuberouter.yaml

kubeadm token create --print-join-command

workers 1,2>

sudo kubeadm join 192.168.2.101:6443 --token b2b482.4jfbj61xlfxubev3 --discovery-token-ca-cert-hash sha256:6c9234fd7381ab8af2e1f15575587e8c1b0fc6094d13bb6fca6c6650ef2513f1

Fix lỗi coredns bị CrashLoopBackOff

Cách 3:

kubectl -n kube-system get deployment coredns -o yaml | \

sed 's/allowPrivilegeEscalation: false/allowPrivilegeEscalation: true/g' | \

kubectl apply -f -

<https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/troubleshooting-kubeadm/>

## Test cluster mới

kubectl create deployment nginx --image=nginx

kubectl describe deployment nginx

kubectl create service nodeport nginx --tcp=80:80

kubectl scale --replicas=2 deployment/nginx

kubectl get svc

wget http://h2.spark.org:32128

wget http://h3.spark.org:32128

#kubectl delete svc nginx

#kubectl delete deployment nginx

# Setup local registry

## Install service

Tạo self-cert

openssl req -newkey rsa:4096 -nodes -sha256 \

-keyout registry.key -x509 -days 365 \

-out registry.crt

Copy self-cert đến các node master và worker

sudo mkdir -p /etc/docker/certs.d/h1.spark.org:5000

sudo cp registry.crt /etc/docker/certs.d/h1.spark.org:5000/ca.crt

Tạo user để login

docker run \

--entrypoint htpasswd \

registry:latest -Bbn testuser testpassword > auth/htpasswd

Run service

docker run -d -p 5000:5000 --name registry \

-v /home/vagrant/registry:/certs \

-v /home/vagrant/images:/var/lib/registry \

-v /home/vagrant/auth:/auth \

-e "REGISTRY\_AUTH=htpasswd" \

-e "REGISTRY\_AUTH\_HTPASSWD\_REALM=Registry Realm" \

-e REGISTRY\_AUTH\_HTPASSWD\_PATH=/auth/htpasswd \

-e REGISTRY\_HTTP\_ADDR=0.0.0.0:5000 \

-e REGISTRY\_HTTP\_TLS\_CERTIFICATE=/certs/registry.crt \

-e REGISTRY\_HTTP\_TLS\_KEY=/certs/registry.key \

registry:latest

Debug service

docker logs registry

Stop service (optional)

docker stop registry

docker rm registry

## Test service

Push image lên repo

docker image pull nginx:1.13-alpine

docker tag nginx:1.13-alpine h1.spark.org:5000/nginx:1.13-alpine

docker login h1.spark.org:5000

docker push h1.spark.org:5000/nginx:1.13-alpine

Config k8s dùng local registry

Tạo config

kubectl create secret docker-registry local-registry \

--docker-server=h1.spark.org:5000 \

--docker-username=testuser \

--docker-password=testpassword \

--docker-email=test@h1.spark.org

Run service

vi nginx-deployment.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

labels:

app: nginx

spec:

replicas: 3

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx-001

image: h1.spark.org:5000/nginx:1.13-alpine

ports:

- containerPort: 80

imagePullSecrets:

- name: local-registry

kubectl create -f nginx-deployment.yaml

kubectl get deployment

#kubectl expose deployment/nginx-deployment --type=LoadBalancer

kubectl get service

Stop service

#kubectl delete service nginx-deployment

#kubectl create service nodeport nginx-deployment --tcp=80:80

#kubectl delete -f nginx-deployment.yaml

# Thêm forward domain cho DNS service của k8s

## Sửa Corefile của coredns

kubectl -n kube-system edit configmap coredns

Corefile: |

local.org:53 {

errors

cache 30

proxy . 192.168.1.104

}

hdfs.org:53 {

errors

cache 30

proxy . 192.168.1.104

}

.:53 {

errors

....

kubectl -n kube-system scale --replicas=0 deployment.apps/coredns

kubectl -n kube-system scale --replicas=2 deployment.apps/coredns

## Test dns

kubectl run -i --tty busybox --image=busybox --restart=Never -- sh

#ping h1.local.org

#exit

kubectl delete pod busybox

## Tìm hiểu thêm về hostname, DNS trên k8s

https://kubernetes.io/docs/concepts/services-networking/dns-pod-service/

https://supergiant.io/blog/kubernetes-dns-for-services-and-pods/

# Setup nginx ingress cho k8s services

## Install nginx ingress

kubectl apply -f <https://raw.githubusercontent.com/kubernetes/ingress-nginx/master/deploy/mandatory.yaml>

# dùng node-port

kubectl apply -f <https://raw.githubusercontent.com/kubernetes/ingress-nginx/master/deploy/provider/baremetal/service-nodeport.yaml>

kubectl get pods --all-namespaces -l app=ingress-nginx

kubectl -n ingress-nginx logs nginx-ingress-controller-5694ccb578-mfrrk

Reference: <https://github.com/kubernetes/ingress-nginx/blob/master/docs/deploy/index.md>

## Test ingress

Reference: <https://matthewpalmer.net/kubernetes-app-developer/articles/kubernetes-ingress-guide-nginx-example.html>

Tạo services

kubectl apply -f apple.yaml

kubectl apply -f banana.yaml

Tạo ingress rules

kubectl create -f ingress.yaml

Access service thông qua ingress

curl https://h2.spark.org:30778/banana

# Một số lệnh hay dùng

kubectl -n kube-system get all

kubectl get all --all-namespaces

kubectl get pod -o wide

kubectl get deployment -o wide

kubectl get svc -o wide

kubectl create -f ….yaml

kubectl delete -f … yaml

kubectl apply -f … yaml

kubectl exec -it pod… /bin/bash

kubectl logs -f pod…

# Viết yaml

## Tạo pod, deployment, service

Reference: <https://www.mirantis.com/blog/introduction-to-yaml-creating-a-kubernetes-deployment/>

kubectl logs rss-site-c88f9b65c-wqxpn rss-reader

kubectl exec -it rss-site-c88f9b65c-wqxpn --container rss-reader -- /bin/bash

kubectl delete -f deployment.yaml

## Gán label cho pod

<http://kubernetesbyexample.com/labels/>

<https://www.digitalocean.com/community/tutorials/webinar-series-a-closer-look-at-kubernetes>

# Setup NFS persistent volume

kubectl create -f …

## Run NFS pod

// 00\_nfs-server.yaml

# Note - an NFS server isn't really a Kubernetes

# concept. We're just creating it in Kubernetes

# for illustration and convenience. In practice,

# it might be run in some other system.

# Create a service to expose the NFS server

# to pods inside the cluster.

kind: Service

apiVersion: v1

metadata:

name: nfs-service

spec:

selector:

role: nfs

ports:

# Open the ports required by the NFS server

# Port 2049 for TCP

- name: tcp-2049

port: 2049

protocol: TCP

# Port 111 for UDP

- name: udp-111

port: 111

protocol: UDP

---

# Run the NFS server image in a pod that is

# exposed by the service.

kind: Pod

apiVersion: v1

metadata:

name: nfs-server-pod

labels:

role: nfs

spec:

containers:

- name: nfs-server-container

image: cpuguy83/nfs-server

securityContext:

privileged: true

args:

# Pass the paths to share to the Docker image

- /exports

## Tạo PV

// 01\_pv.yaml

apiVersion: v1

kind: PersistentVolume

metadata:

name: pv0001

spec:

capacity:

storage: 1Gi

accessModes:

- ReadWriteMany

nfs:

path: /

server: 10.99.192.18

---

kind: PersistentVolumeClaim

apiVersion: v1

metadata:

name: nfs

spec:

accessModes:

- ReadWriteMany

storageClassName: ""

resources:

requests:

storage: 1Gi

## Test mount PV

// 02\_pod.yaml

# Create a pod that reads and writes to the

# NFS server via an NFS volume.

kind: Pod

apiVersion: v1

metadata:

name: pod-using-nfs

spec:

# Add the server as an NFS volume for the pod

volumes:

- name: nfs-volume

persistentVolumeClaim:

claimName: nfs

# In this container, we'll mount the NFS volume

# and write the date to a file inside it.

containers:

- name: app

image: alpine

# Mount the NFS volume in the container

volumeMounts:

- name: nfs-volume

mountPath: /var/nfs

# Write to a file inside our NFS

command: ["/bin/sh"]

args: ["-c", "while true; do date >> /var/nfs/dates.txt; sleep 5; done"]

$ kubectl get pv

// 03\_pod.yaml

apiVersion: extensions/v1beta1

kind: Deployment

metadata:

name: nfs-busybox

spec:

replicas: 1

selector:

matchLabels:

name: nfs-busybox

template:

metadata:

labels:

name: nfs-busybox

spec:

containers:

- image: nginx

imagePullPolicy: Always

name: busybox

volumeMounts:

# name must match the volume name below

- name: my-pvc-nfs

mountPath: "/opt/data"

volumes:

- name: my-pvc-nfs

persistentVolumeClaim:

claimName: nfs

# Setup alluxio cluster trên k8s

## Build alluxio docker

vi Dockerfile

FROM centos:7.4.1708

#ARG ALLUXIO\_TARBALL=http://downloads.alluxio.org/downloads/files/1.8.1/alluxio-1.8.1-bin.tar.gz

RUN yum -y install epel-release java-1.8.0-openjdk-devel krb5-workstation which

RUN yum -y install net-tools nc vim nano

RUN ln -sf /usr/share/zoneinfo/Asia/Ho\_Chi\_Minh /etc/localtime

#ADD krb5.conf /etc/krb5.conf

#ADD ${ALLUXIO\_TARBALL} /opt/

#ADD alluxio-1.8.1-hadoop-2.7-bin.tar.gz /opt/

#RUN cd /opt && \

# (if ls | grep -q ".tar.gz"; then tar -xzf \*.tar.gz && rm \*.tar.gz; fi) && \

# ln -s alluxio-\* alluxio

ADD alluxio-1.8.1-hadoop-2.7 /opt/alluxio

#COPY conf /opt/alluxio/conf/

#COPY entrypoint.sh /

ADD common.sh start-master.sh start-worker.sh entrypoint.sh /

RUN chmod +x /common.sh /start-master.sh /start-worker.sh /entrypoint.sh

WORKDIR /opt/alluxio

#ENTRYPOINT ["/entrypoint.sh"]

vi entrypoint.sh

#!/usr/bin/env bash

#

# The Alluxio Open Foundation licenses this work under the Apache License, version 2.0

# (the "License"). You may not use this work except in compliance with the License, which is

# available at www.apache.org/licenses/LICENSE-2.0

#

# This software is distributed on an "AS IS" basis, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,

# either express or implied, as more fully set forth in the License.

#

# See the NOTICE file distributed with this work for information regarding copyright ownership.

#

set -e

NO\_FORMAT='--no-format'

function printUsage {

echo "Usage: COMMAND [COMMAND\_OPTIONS]"

echo

echo "COMMAND is one of:"

echo -e " master [--no-format] \t Start Alluxio master. If --no-format is specified, do not format"

echo -e " worker [--no-format] \t Start Alluxio worker. If --no-format is specified, do not format"

echo -e " proxy \t Start Alluxio proxy"

}

if [[ $# -lt 1 ]]; then

printUsage

exit 1

fi

service=$1

options=$2

# Only set ALLUXIO\_RAM\_FOLDER if tiered storage isn't explicitly configured

if [[ -z "${ALLUXIO\_WORKER\_TIEREDSTORE\_LEVEL0\_DIRS\_PATH}" ]]; then

# Docker will set this tmpfs up by default. Its size is configurable through the

# --shm-size argument to docker run

export ALLUXIO\_RAM\_FOLDER=${ALLUXIO\_RAM\_FOLDER:-/dev/shm}

fi

home=/opt/alluxio

cd ${home}

# List of environment variables which go in alluxio-env.sh instead of

# alluxio-site.properties

alluxio\_env\_vars=(

ALLUXIO\_CLASSPATH

ALLUXIO\_HOSTNAME

ALLUXIO\_JARS

ALLUXIO\_JAVA\_OPTS

ALLUXIO\_MASTER\_JAVA\_OPTS

ALLUXIO\_PROXY\_JAVA\_OPTS

ALLUXIO\_RAM\_FOLDER

ALLUXIO\_USER\_JAVA\_OPTS

ALLUXIO\_WORKER\_JAVA\_OPTS

)

for keyvaluepair in $(env); do

# split around the "="

key=$(echo ${keyvaluepair} | cut -d= -f1)

value=$(echo ${keyvaluepair} | cut -d= -f2-)

#if [[ "$key" == "ALLUXIO\_MASTER\_PORT" ]]; then

# continue

#fi

if [[ "${alluxio\_env\_vars[\*]}" =~ "${key}" ]]; then

echo "export ${key}=${value}" >> conf/alluxio-env.sh

else

# check if property name is valid

if confkey=$(bin/alluxio runClass alluxio.cli.GetConfKey ${key} 2> /dev/null); then

echo "${confkey}=${value}" >> conf/alluxio-site.properties

fi

fi

done

if [ "$ENABLE\_FUSE" = true ]; then

integration/fuse/bin/alluxio-fuse mount /alluxio-fuse /

fi

case ${service,,} in

master)

if [[ -n ${options} && ${options} != ${NO\_FORMAT} ]]; then

printUsage

exit 1

fi

if [[ ${options} != ${NO\_FORMAT} ]]; then

bin/alluxio formatMaster

fi

integration/docker/bin/alluxio-master.sh

;;

worker)

if [[ -n ${options} && ${options} != ${NO\_FORMAT} ]]; then

printUsage

exit 1

fi

if [[ ${options} != ${NO\_FORMAT} ]]; then

bin/alluxio formatWorker

fi

integration/docker/bin/alluxio-worker.sh

;;

proxy)

integration/docker/bin/alluxio-proxy.sh

;;

\*)

printUsage

exit 1

;;

esac

vi common.sh

#!/bin/bash

unset ALLUXIO\_MASTER\_PORT

unset ALLUXIO\_WORKER\_PORT

cp /conf/krb5.conf /etc/

cp -r /conf/conf /opt/alluxio/

vi start-master.sh

#!/bin/bash

. /common.sh

echo "$(hostname -i) $ALLUXIO\_MASTER\_HOSTNAME" >> /etc/hosts

/entrypoint.sh master

vi start-worker.sh

#!/bin/bash

. /common.sh

if ! getent hosts $ALLUXIO\_MASTER\_HOSTNAME; then

echo "=== Cannot resolve the DNS entry for spark-master. Has the service been created yet, and is SkyDNS functional?"

echo "=== See http://kubernetes.io/v1.1/docs/admin/dns.html for more details on DNS integration."

echo "=== Sleeping 10s before pod exit."

sleep 10

exit 0

fi

echo "alluxio.worker.hostname=$(hostname -i)" >> /opt/alluxio/conf/alluxio-site.properties

alluxio-conf

├── conf

│   ├── alluxio-env.sh.template

│   ├── alluxio-site.properties.template

│   ├── core-site.xml

│   ├── hdfs-site.xml

│   ├── mapred-site.xml

│   └── yarn-site.xml

└── krb5.conf

docker build -t alluxio-docker .

docker tag alluxio-docker:latest h1.spark.org:5000/alluxio-docker:latest

docker push h1.spark.org:5000/alluxio-docker:latest

## Copy alluxio-conf đến persistent volume

kubectl cp alluxio-conf nfs-server-pod:/exports/

kubectl exec -it nfs-server-pod sh

# ls -la /exports/alluxio-conf

## Copy keytab đến secret

kubectl create secret generic dp-sec --from-file=keytab=/home/vagrant/kerberos/data/keytab

## Run alluxio cluster

vi master.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: alluxio-master

labels:

app: alluxio-master

spec:

replicas: 1

selector:

matchLabels:

app: alluxio-master

template:

metadata:

labels:

app: alluxio-master

spec:

containers:

- name: alluxio-master

image: h1.spark.org:5000/alluxio-docker:latest

imagePullPolicy: Always

resources:

requests:

cpu: "0.5"

memory: "512M"

limits:

cpu: "1"

memory: "1024M"

command: ["/start-master.sh"]

env:

- name: ALLUXIO\_MASTER\_HOSTNAME

value: alluxio-master

- name: alluxio.master.keytab.file

value: /etc/keytab

- name: alluxio.master.principal

value: tuanndd@HDFS.ORG

- name: alluxio.security.login.impersonation.username

value: \_NONE\_

ports:

- containerPort: 19998

name: rpc

- containerPort: 19999

name: web

volumeMounts:

- name: vol-conf

mountPath: /conf

subPath: alluxio-conf

- name: vol-sec

mountPath: /etc/keytab

subPath: tuanndd.keytab

restartPolicy: Always

imagePullSecrets:

- name: local-registry

volumes:

- name: vol-conf

persistentVolumeClaim:

claimName: nfs

- name: vol-sec

secret:

secretName: dp-sec

items:

- key: keytab

path: tuanndd.keytab

---

kind: Service

apiVersion: v1

metadata:

name: alluxio-master

labels:

app: alluxio-master

spec:

ports:

- name: rpc

protocol: TCP

port: 19998

targetPort: 19998

- name: web

protocol: TCP

port: 19999

targetPort: 19999

selector:

app: alluxio-master

vi worker.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: alluxio-worker

labels:

app: alluxio-worker

spec:

replicas: 2

selector:

matchLabels:

app: alluxio-worker

template:

metadata:

labels:

app: alluxio-worker

spec:

containers:

- name: alluxio-worker

image: h1.spark.org:5000/alluxio-docker:latest

imagePullPolicy: Always

resources:

requests:

cpu: "0.5"

memory: "2G"

limits:

cpu: "1"

memory: "2G"

command: ["/start-worker.sh"]

env:

- name: ALLUXIO\_MASTER\_HOSTNAME

value: alluxio-master

- name: alluxio.worker.keytab.file

value: /etc/keytab

- name: alluxio.worker.principal

value: tuanndd@HDFS.ORG

- name: ALLUXIO\_WORKER\_MEMORY\_SIZE

value: 1G

volumeMounts:

- name: vol-conf

mountPath: /conf

subPath: alluxio-conf

- name: vol-sec

mountPath: /etc/keytab

subPath: tuanndd.keytab

- name: alluxio-ramdisk

mountPath: /dev/shm

restartPolicy: Always

imagePullSecrets:

- name: local-registry

volumes:

- name: vol-conf

persistentVolumeClaim:

claimName: nfs

- name: vol-sec

secret:

secretName: dp-sec

items:

- key: keytab

path: tuanndd.keytab

- name: alluxio-ramdisk

emptyDir:

medium: "Memory"

sizeLimit: "1G"

kubectl create -f master.yaml

kubectl create -f worker.yaml

kubectl get pod -o wide

## Mount HDFS

kubectl get pod -o wide

kubectl exec -it alluxio-worker-1-deployment-6988db748b-26dbk /bin/bash

# cd /opt/alluxio

# bin/alluxio fs mount /hdfs hdfs://h1.local.org:8020/

## Stop alluxio cluster (optional)

kubectl delete -f worker.yaml

kubectl delete -f master.yaml

# Setup Spark cluster trên k8s

## Build spark docker

vi Dockerfile

# inspired by https://hub.docker.com/r/wongnai/spark-standalone/

FROM centos:7.4.1708

RUN yum -y install epel-release java-1.8.0-openjdk-devel krb5-workstation which

RUN yum -y install net-tools nc vim nano

RUN ln -sf /usr/share/zoneinfo/Asia/Ho\_Chi\_Minh /etc/localtime

#ADD krb5.conf /etc/krb5.conf

ADD hadoop-2.7.3 /opt/hadoop

ADD spark-2.3.0-bin-hadoop2.7 /opt/spark

ADD alluxio-1.8.1-hadoop-2.7 /opt/alluxio

ARG CONDA\_PKG="Miniconda3-4.5.12-Linux-x86\_64.sh"

ARG CONDA\_DIR="/opt/anaconda3"

ADD $CONDA\_PKG /

RUN bash /$CONDA\_PKG -b -p $CONDA\_DIR

RUN rm -f /$CONDA\_PKG

ENV PATH $CONDA\_DIR/bin:$PATH

RUN conda create -n py3 python=3

ADD common.sh start-master start-worker /

#ADD spark-defaults.conf /opt/spark/conf/spark-defaults.conf

RUN chmod +x /common.sh /start-master /start-worker

ENV PATH=/opt/spark/bin:/opt/spark/sbin:$PATH

#EXPOSE 6066 7077 7078 8080 8081

WORKDIR /opt/spark

vi common.sh

#!/bin/bash

unset SPARK\_MASTER\_PORT

export LD\_LIBRARY\_PATH=${LD\_LIBRARY\_PATH}:/opt/hadoop/lib/native

cat > /opt/spark/conf/spark-defaults.conf << EOF

spark.master spark://$SPARK\_MASTER\_HOSTNAME:7077

spark.driver.extraLibraryPath /opt/hadoop/lib/native

spark.driver.extraClassPath /opt/alluxio/client/alluxio-1.8.1-client.jar

spark.executor.extraClassPath /opt/alluxio/client/alluxio-1.8.1-client.jar

EOF

vi start-master

#!/bin/sh

. /common.sh

echo "$(hostname -i) $SPARK\_MASTER\_HOSTNAME" >> /etc/hosts

/opt/spark/bin/spark-class org.apache.spark.deploy.master.Master --ip $SPARK\_MASTER\_HOSTNAME --port 7077 --webui-port 8080

#/opt/spark/sbin/start-master.sh --ip spark-master --port 7077

vi start-worker

#!/bin/sh

. /common.sh

#/opt/spark/sbin/start-slave.sh spark://spark-master:7077

if ! getent hosts $SPARK\_MASTER\_HOSTNAME; then

echo "=== Cannot resolve the DNS entry for spark-master. Has the service been created yet, and is SkyDNS functional?"

echo "=== See http://kubernetes.io/v1.1/docs/admin/dns.html for more details on DNS integration."

echo "=== Sleeping 10s before pod exit."

sleep 10

exit 0

fi

# Run spark-class directly so that when it exits (or crashes), the pod restarts.

/opt/spark/bin/spark-class org.apache.spark.deploy.worker.Worker spark://$SPARK\_MASTER\_HOSTNAME:7077 --webui-port 8081

docker build -t spark-docker .

docker tag spark-docker:latest h1.spark.org:5000/spark-docker:latest

docker push h1.spark.org:5000/spark-docker:latest

## Run spark cluster

vi master.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: spark-master

labels:

app: spark-master

spec:

replicas: 1

selector:

matchLabels:

app: spark-master

template:

metadata:

labels:

app: spark-master

spec:

containers:

- name: spark-master

image: h1.spark.org:5000/spark-docker:latest

imagePullPolicy: Always

resources:

requests:

cpu: 100m

command: ["/start-master"]

env:

- name: SPARK\_MASTER\_HOSTNAME

value: spark-master

ports:

- containerPort: 7077

name: spark

- containerPort: 8080

name: http

restartPolicy: Always

imagePullSecrets:

- name: local-registry

---

kind: Service

apiVersion: v1

metadata:

name: spark-master

labels:

app: spark-master

spec:

ports:

- port: 7077

targetPort: 7077

name: spark

- port: 8080

targetPort: 8080

name: http

selector:

app: spark-master

vi worker.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: spark-worker

labels:

app: spark-worker

spec:

replicas: 2

selector:

matchLabels:

app: spark-worker

template:

metadata:

labels:

app: spark-worker

spec:

containers:

- name: spark-worker

image: h1.spark.org:5000/spark-docker:latest

imagePullPolicy: Always

resources:

requests:

cpu: 100m

command: ["/start-worker"]

env:

- name: SPARK\_MASTER\_HOSTNAME

value: spark-master

ports:

- containerPort: 8081

restartPolicy: Always

imagePullSecrets:

- name: local-registry

vi ui-proxy.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: spark-ui-proxy

labels:

app: spark-ui-proxy

spec:

replicas: 1

selector:

matchLabels:

app: spark-ui-proxy

template:

metadata:

labels:

app: spark-ui-proxy

spec:

containers:

- name: spark-ui-proxy

image: elsonrodriguez/spark-ui-proxy:1.0

ports:

- containerPort: 80

resources:

requests:

cpu: 100m

args:

- spark-master:8080

livenessProbe:

httpGet:

path: /

port: 80

initialDelaySeconds: 120

timeoutSeconds: 5

---

kind: Service

apiVersion: v1

metadata:

name: spark-ui-proxy

labels:

app: spark-ui-proxy

spec:

ports:

- port: 80

targetPort: 80

selector:

app: spark-ui-proxy

type: NodePort

kubectl create -f master.yaml

kubectl create -f worker.yaml

kubectl create -f ui-proxy.yaml

kubectl get pod -o wide

kubectl get svc -o wide

## Test spark

kubectl get pod -o wide

kubectl exec -it spark-worker-deployment-b7b6467b9-ff4rn /bin/bash

# spark-shell

> sc.setLogLevel("INFO")

> sc.textFile("alluxio://alluxio-master:19998/hdfs/tmp/LICENSE").count()

val NUM\_SAMPLES = 1000000

val count1 = sc.parallelize(1 to NUM\_SAMPLES).filter { \_ =>

val x = math.random

val y = math.random

x\*x + y\*y < 1

}.count()

println(s"Pi is roughly ${4.0 \* count1 / NUM\_SAMPLES}")

wget <http://h2.spark.org:32656>

## Stop spark cluster (optional)

kubectl delete -f ui-proxy.yaml

kubectl delete -f worker.yaml

kubectl delete -f

# Setup Jupyter

## Build jupyter docker

docker build -t jupyter-docker .

docker tag jupyter-docker:latest h1.spark.org:5000/jupyter-docker:latest

docker push h1.spark.org:5000/jupyter-docker:latest

vi Dockerfile

# inspired by https://hub.docker.com/r/wongnai/spark-standalone/

FROM centos:7.4.1708

#ARG JUPYTER\_USER="jupyter"

#ARG JUPYTER\_UID="1000"

#ARG JUPYTER\_GID="1000"

#USER root

RUN yum -y install epel-release java-1.8.0-openjdk-devel krb5-workstation which

RUN yum -y install net-tools nc vim nano

RUN yum -y install bzip2

RUN ln -sf /usr/share/zoneinfo/Asia/Ho\_Chi\_Minh /etc/localtime

ADD hadoop-2.7.3 /opt/hadoop

ADD spark-2.3.0-bin-hadoop2.7 /opt/spark

ADD alluxio-1.8.1-hadoop-2.7 /opt/alluxio

ADD apache-hive-1.2.2-bin /opt/hive

ARG CONDA\_PKG="Miniconda3-4.5.12-Linux-x86\_64.sh"

ARG CONDA\_DIR="/opt/anaconda3"

ADD $CONDA\_PKG /

RUN bash /$CONDA\_PKG -b -p $CONDA\_DIR

RUN rm -f /$CONDA\_PKG

ENV PATH $CONDA\_DIR/bin:$PATH

RUN conda create -n py3 python=3

RUN conda install -n py3 jupyter ipykernel -y

RUN conda install -n py3 PyHamcrest -y

RUN source /opt/anaconda3/bin/activate py3 && \

pip install toree && \

jupyter toree install --spark\_home=/opt/spark

#RUN conda install jupyter ipykernel -y

#RUN conda create -n py3 python=3.5

#RUN /opt/anaconda3/bin/conda install -n py3 ipykernel -y

#RUN source /opt/anaconda3/bin/activate py3 && \

# ipython kernel install

# pip install jupyter

#USER $JUPYTER\_UID

#RUN conda run -n py3 -- ipython kernel install

##ADD krb5.conf /etc/krb5.conf

#ADD alluxio-1.8.1-hadoop-2.7 /opt/alluxio

#ADD spark-2.0.1-bin-hadoop2.7 /opt/spark

#ADD hadoop-2.7.3 /opt/hadoop

ADD common.sh start-jupyter /

#ADD spark-defaults.conf /opt/spark/conf/spark-defaults.conf

RUN chmod +x /common.sh /start-jupyter

ENV PATH=/opt/hadoop/bin:/opt/hive/bin:/opt/spark/bin:$PATH

#EXPOSE 6066 7077 7078 8080 8081

#WORKDIR /opt/spark

#ENTRYPOINT ["/opt/spark/bin/spark-class"]

#CMD ["org.apache.spark.deploy.master.Master", "--ip spark-master", "--port 7077", "--webui-port 8080"]

vi common.sh

#!/bin/bash

#unset SPARK\_MASTER\_PORT

export LD\_LIBRARY\_PATH=${LD\_LIBRARY\_PATH}:/opt/hadoop/lib/native

cat > /opt/spark/conf/spark-defaults.conf << EOF

spark.master SPARK\_MASTER\_URL

spark.driver.extraLibraryPath /opt/hadoop/lib/native

spark.driver.extraClassPath /opt/alluxio/client/alluxio-1.8.1-client.jar

spark.executor.extraClassPath /opt/alluxio/client/alluxio-1.8.1-client.jar

spark.driver.host $(hostname -i)

EOF

cp -f /opt/hadoop-conf/krb5.conf /etc/

kinit -kt /etc/keytab $KRB\_USERNAME

cp /opt/hadoop-conf/core-site.xml /opt/hive/conf/

cat > /opt/hive/conf/hive-env.sh << EOF

export HADOOP\_HOME=/opt/hadoop

export JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.212.b04-0.el7\_6.x86\_64

EOF

mkdir -p /opt/anaconda3/envs/py3/share/jupyter/kernels/py3spark2

cat > /opt/anaconda3/envs/py3/share/jupyter/kernels/py3spark2/kernel.json << EOF

{

"argv": [

"/opt/anaconda3/envs/py3/bin/python",

"-m",

"ipykernel\_launcher",

"-f",

"{connection\_file}"

],

"display\_name": "Python 3.5 + PySpark(spark-2.3.0-bin-hadoop2.7)",

"language": "python",

"env": {

"JAVA\_HOME": "/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.212.b04-0.el7\_6.x86\_64",

"HADOOP\_CLIENT\_OPTS": "-Djava.net.preferIPv4Stack=true",

"HADOOP\_CONF\_DIR": "/opt/hadoop-conf",

"SPARK\_HOME": "/opt/spark",

"PYSPARK\_PYTHON": "/opt/anaconda3/envs/py3/bin/python",

"PYSPARK\_SUBMIT\_ARGS": " --master $SPARK\_MASTER\_URL pyspark-shell",

"PYTHONPATH": "/opt/spark/python/lib/py4j-0.10.6-src.zip:/opt/spark/python/",

"PYTHONSTARTUP": "/opt/spark/python/pyspark/shell.py"

}

}

EOF

cat > /usr/local/share/jupyter/kernels/apache\_toree\_scala/kernel.json << EOF

{

"argv": [

"/usr/local/share/jupyter/kernels/apache\_toree\_scala/bin/run.sh",

"--profile",

"{connection\_file}"

],

"env": {

"DEFAULT\_INTERPRETER": "Scala",

"\_\_TOREE\_SPARK\_OPTS\_\_": "--master $SPARK\_MASTER\_URL --deploy-mode client",

"\_\_TOREE\_OPTS\_\_": "",

"SPARK\_HOME": "/opt/spark",

"PYTHONPATH": "/opt/spark/python:/opt/spark/python/lib/py4j-0.10.6-src.zip",

"PYTHON\_EXEC": "/opt/anaconda3/envs/py3/bin/python"

},

"display\_name": "Apache Toree - Scala",

"language": "scala",

"interrupt\_mode": "signal",

"metadata": {}

}

EOF

vi start-juyter

#!/bin/bash

. /common.sh

source /opt/anaconda3/bin/activate py3 && \

jupyter notebook --allow-root --no-browser --ip='0.0.0.0' --NotebookApp.token='' --NotebookApp.password='' --NotebookApp.quit\_button=False

## Copy hadoop-conf đến persistent volume

kubectl cp hadoop-conf nfs-server-pod:/exports/

kubectl exec -it nfs-server-pod sh

# ls -la /exports/hadoop-conf

## Copy keytab đến secret

kubectl create secret generic dp-sec --from-file=keytab=/home/vagrant/kerberos/data/keytab

## Start jupyter

vi jupyter.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: jupyter

labels:

app: jupyter

spec:

replicas: 1

selector:

matchLabels:

app: jupyter

template:

metadata:

labels:

app: jupyter

spec:

containers:

- name: jupyter

image: h1.spark.org:5000/jupyter-docker:latest

imagePullPolicy: Always

resources:

requests:

cpu: 100m

#command: ["/start-jupyter"]

command: ["sleep", "3600"]

env:

- name: KRB\_USERNAME

value: tuanndd@HDFS.ORG

- name: SPARK\_MASTER\_URL

value: spark://spark-master:7077

- name: MY\_POD\_IP

valueFrom:

fieldRef:

fieldPath: status.podIP

- name: HADOOP\_HOME

value: /opt/hadoop

- name: JAVA\_HOME

value: /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.212.b04-0.el7\_6.x86\_64

- name: HIVE\_HOME

value: /opt/hive

- name: SPARK\_HOME

value: /opt/spark

ports:

- containerPort: 8888

volumeMounts:

- name: vol-conf

mountPath: /opt/hadoop-conf

subPath: hadoop-conf

- name: vol-sec

mountPath: /etc/keytab

subPath: tuanndd.keytab

restartPolicy: Always

imagePullSecrets:

- name: local-registry

volumes:

- name: vol-conf

persistentVolumeClaim:

claimName: nfs

- name: vol-sec

secret:

secretName: dp-sec

items:

- key: keytab

path: tuanndd.keytab

---

kind: Service

apiVersion: v1

metadata:

name: jupyter

labels:

app: jupyter

spec:

ports:

- port: 8888

targetPort: 8888

selector:

app: jupyter

type: NodePort

kubectl create -f jypyter.yaml

## Test jupyter

<http://h4.spark.org:31319>

**Terminal**>

hadoop fs -ls /

beeline -u "jdbc:hive2://h2.local.org:10000/default;principal=hive/\_HOST@HDFS.ORG"

show tables;

…

SELECT d.driverId, d.name, t.total\_hours, t.total\_miles from drivers d

JOIN (SELECT driverId, sum(hours\_logged)total\_hours, sum(miles\_logged)total\_miles FROM timesheet GROUP BY driverId ) t

ON (d.driverId = t.driverId)

…

Reference: <https://hortonworks.com/tutorial/how-to-process-data-with-apache-hive/>

**Toree**>

sc

sc.setLogLevel("INFO")

sc.getConf.getAll.foreach(println)

sc.textFile("alluxio://alluxio-master:19998/hdfs/tmp/LICENSE").count()

val NUM\_SAMPLES = 1000000

val count1 = sc.parallelize(1 to NUM\_SAMPLES).filter { \_ =>

val x = math.random

val y = math.random

x\*x + y\*y < 1

}.count()

println(s"Pi is roughly ${4.0 \* count1 / NUM\_SAMPLES}")

**PySpark**>

from pyspark import SparkContext

from pyspark import SparkConf

import random

NUM\_SAMPLES = 100000

def inside(p):

x, y = random.random(), random.random()

return x\*x + y\*y < 1

count = sc.parallelize(range(0, NUM\_SAMPLES)).filter(inside).count()

pi = 4 \* count / NUM\_SAMPLES

print("Pi is roughly", pi)

sc.textFile("alluxio://alluxio-master:19998/hdfs/tmp/LICENSE").count()

## Stop jupyter (optional)

kubectl delete –f jupyter.yaml

# Setup Zeppelin

## Build zeppelin docker

vi Dockerfile

FROM centos:7.4.1708

RUN yum -y install epel-release java-1.8.0-openjdk-devel krb5-workstation which

RUN yum -y install net-tools nc vim nano

RUN ln -sf /usr/share/zoneinfo/Asia/Ho\_Chi\_Minh /etc/localtime

ADD zeppelin-0.8.1-bin-all /opt/zeppelin

ADD spark-2.0.1-bin-hadoop2.7 /opt/spark

ADD alluxio-1.8.1-hadoop-2.7 /opt/alluxio

ADD apache-hive-1.2.2-bin /opt/hive

COPY zeppelin-env.sh /opt/zeppelin/conf/

COPY interpreter.json /opt/zeppelin/conf/

ADD common.sh start-zeppelin /

RUN chmod +x /common.sh /start-zeppelin

WORKDIR /opt/zeppelin

vi common.sh

#!/bin/bash

unset ZEPPELIN\_PORT

cat > /opt/spark/conf/spark-defaults.conf << EOF

spark.master $SPARK\_MASTER\_URL

spark.driver.extraLibraryPath /opt/hadoop/lib/native

spark.driver.extraClassPath /opt/alluxio/client/alluxio-1.8.1-client.jar

spark.executor.extraClassPath /opt/alluxio/client/alluxio-1.8.1-client.jar

EOF

cat > /opt/zeppelin/conf/zeppelin-site.xml << EOF

<configuration>

<property>

<name>hadoop.security.authentication</name>

<value>kerberos</value>

</property>

</configuration>

EOF

cp /opt/hive/lib/hive-jdbc-1.2.2-standalone.jar /opt/zeppelin/interpreter/jdbc/

cp /opt/hive/lib/hive-shims-1.2.2.jar /opt/zeppelin/interpreter/jdbc/

cp /opt/hive/lib/commons-collections-3.2.2.jar /opt/zeppelin/interpreter/jdbc/

cp /opt/zeppelin/lib/hadoop-auth-2.7.3.jar /opt/zeppelin/interpreter/jdbc/

cp /opt/zeppelin/lib/hadoop-common-2.7.3.jar /opt/zeppelin/interpreter/jdbc/

vi start-zeppelin

#!/bin/bash

. /common.sh

cp /opt/hadoop-conf/krb5.conf /etc/

kinit -kt /etc/keytab $KRB\_USERNAME

cp /opt/alluxio/client/\*.jar /opt/zeppelin/lib/interpreter/

sed -i "s|local\[\\*\]|${SPARK\_MASTER\_URL}|g" /opt/zeppelin/conf/interpreter.json

/opt/zeppelin/bin/zeppelin.sh

vi zeppelin-env.sh

export HADOOP\_CONF\_DIR=/opt/hadoop-conf

export SPARK\_HOME=/opt/spark

vi interpreter.json

{

"interpreterSettings": {

"spark": {

"id": "spark",

"name": "spark",

"group": "spark",

"properties": {

"spark.executor.memory": {

"name": "spark.executor.memory",

"value": "",

"type": "string"

},

"zeppelin.spark.sql.interpolation": {

"name": "zeppelin.spark.sql.interpolation",

"value": false,

"type": "checkbox"

},

"zeppelin.spark.concurrentSQL": {

"name": "zeppelin.spark.concurrentSQL",

"value": false,

"type": "checkbox"

},

"zeppelin.R.knitr": {

"name": "zeppelin.R.knitr",

"value": true,

"type": "checkbox"

},

"zeppelin.R.cmd": {

"name": "zeppelin.R.cmd",

"value": "R",

"type": "string"

},

"spark.app.name": {

"name": "spark.app.name",

"value": "Zeppelin",

"type": "string"

},

"zeppelin.R.image.width": {

"name": "zeppelin.R.image.width",

"value": "100%",

"type": "number"

},

"zeppelin.spark.importImplicit": {

"name": "zeppelin.spark.importImplicit",

"value": true,

"type": "checkbox"

},

"zeppelin.dep.additionalRemoteRepository": {

"name": "zeppelin.dep.additionalRemoteRepository",

"value": "spark-packages,http://dl.bintray.com/spark-packages/maven,false;",

"type": "textarea"

},

"zeppelin.spark.maxResult": {

"name": "zeppelin.spark.maxResult",

"value": "1000",

"type": "number"

},

"zeppelin.spark.ui.hidden": {

"name": "zeppelin.spark.ui.hidden",

"value": false,

"type": "checkbox"

},

"master": {

"name": "master",

"value": "local[\*]",

"type": "string"

},

"zeppelin.pyspark.python": {

"name": "zeppelin.pyspark.python",

"value": "python",

"type": "string"

},

"args": {

"name": "args",

"value": "",

"type": "textarea"

},

"zeppelin.spark.enableSupportedVersionCheck": {

"name": "zeppelin.spark.enableSupportedVersionCheck",

"value": true,

"type": "checkbox"

},

"zeppelin.spark.useNew": {

"name": "zeppelin.spark.useNew",

"value": true,

"type": "checkbox"

},

"zeppelin.dep.localrepo": {

"name": "zeppelin.dep.localrepo",

"value": "local-repo",

"type": "string"

},

"zeppelin.pyspark.useIPython": {

"name": "zeppelin.pyspark.useIPython",

"value": true,

"type": "checkbox"

},

"zeppelin.spark.sql.stacktrace": {

"name": "zeppelin.spark.sql.stacktrace",

"value": false,

"type": "checkbox"

},

"zeppelin.spark.useHiveContext": {

"name": "zeppelin.spark.useHiveContext",

"value": true,

"type": "checkbox"

},

"zeppelin.spark.uiWebUrl": {

"name": "zeppelin.spark.uiWebUrl",

"value": "",

"type": "string"

},

"zeppelin.R.render.options": {

"name": "zeppelin.R.render.options",

"value": "out.format \u003d \u0027html\u0027, comment \u003d NA, echo \u003d FALSE, results \u003d \u0027asis\u0027, message \u003d F, warning \u003d F, fig.retina \u003d 2",

"type": "textarea"

},

"zeppelin.spark.printREPLOutput": {

"name": "zeppelin.spark.printREPLOutput",

"value": true,

"type": "checkbox"

},

"spark.cores.max": {

"name": "spark.cores.max",

"value": "",

"type": "number"

}

},

"status": "READY",

"interpreterGroup": [

{

"name": "spark",

"class": "org.apache.zeppelin.spark.SparkInterpreter",

"defaultInterpreter": true,

"editor": {

"language": "scala",

"editOnDblClick": false,

"completionKey": "TAB",

"completionSupport": true

}

},

{

"name": "sql",

"class": "org.apache.zeppelin.spark.SparkSqlInterpreter",

"defaultInterpreter": false,

"editor": {

"language": "sql",

"editOnDblClick": false,

"completionKey": "TAB",

"completionSupport": true

}

},

{

"name": "dep",

"class": "org.apache.zeppelin.spark.DepInterpreter",

"defaultInterpreter": false,

"editor": {

"language": "scala",

"editOnDblClick": false,

"completionKey": "TAB",

"completionSupport": true

}

},

{

"name": "pyspark",

"class": "org.apache.zeppelin.spark.PySparkInterpreter",

"defaultInterpreter": false,

"editor": {

"language": "python",

"editOnDblClick": false,

"completionKey": "TAB",

"completionSupport": true

}

},

{

"name": "ipyspark",

"class": "org.apache.zeppelin.spark.IPySparkInterpreter",

"defaultInterpreter": false,

"editor": {

"language": "python",

"editOnDblClick": false,

"completionSupport": true

}

},

{

"name": "r",

"class": "org.apache.zeppelin.spark.SparkRInterpreter",

"defaultInterpreter": false,

"editor": {

"language": "r",

"editOnDblClick": false,

"completionSupport": false

}

}

],

"dependencies": [],

"option": {

"remote": true,

"port": -1,

"isExistingProcess": false,

"setPermission": false,

"owners": [],

"isUserImpersonate": false

}

},

"python": {

"id": "python",

"name": "python",

"group": "python",

"properties": {

"zeppelin.python": {

"name": "zeppelin.python",

"value": "python",

"type": "string"

},

"zeppelin.ipython.launch.timeout": {

"name": "zeppelin.ipython.launch.timeout",

"value": "30000",

"type": "number"

},

"zeppelin.python.useIPython": {

"name": "zeppelin.python.useIPython",

"value": true,

"type": "checkbox"

},

"zeppelin.ipython.grpc.message\_size": {

"name": "zeppelin.ipython.grpc.message\_size",

"value": "33554432",

"type": "number"

},

"zeppelin.python.maxResult": {

"name": "zeppelin.python.maxResult",

"value": "1000",

"type": "number"

}

},

"status": "READY",

"interpreterGroup": [

{

"name": "python",

"class": "org.apache.zeppelin.python.PythonInterpreter",

"defaultInterpreter": false,

"editor": {

"language": "python",

"editOnDblClick": false,

"completionSupport": true

}

},

{

"name": "ipython",

"class": "org.apache.zeppelin.python.IPythonInterpreter",

"defaultInterpreter": false,

"editor": {

"language": "python",

"editOnDblClick": false,

"completionKey": "TAB",

"completionSupport": true

}

},

{

"name": "sql",

"class": "org.apache.zeppelin.python.PythonInterpreterPandasSql",

"defaultInterpreter": false,

"editor": {

"language": "sql",

"editOnDblClick": false,

"completionKey": "TAB",

"completionSupport": false

}

},

{

"name": "conda",

"class": "org.apache.zeppelin.python.PythonCondaInterpreter",

"defaultInterpreter": false,

"editor": {

"language": "sh",

"editOnDblClick": false,

"completionSupport": false

}

},

{

"name": "docker",

"class": "org.apache.zeppelin.python.PythonDockerInterpreter",

"defaultInterpreter": false,

"editor": {

"language": "sh",

"editOnDblClick": false,

"completionSupport": false

}

}

],

"dependencies": [],

"option": {

"remote": true,

"port": -1,

"isExistingProcess": false,

"setPermission": false,

"owners": [],

"isUserImpersonate": false

}

}

},

"interpreterBindings": {},

"interpreterRepositories": [

{

"id": "central",

"type": "default",

"url": "http://repo1.maven.org/maven2/",

"releasePolicy": {

"enabled": true,

"updatePolicy": "daily",

"checksumPolicy": "warn"

},

"snapshotPolicy": {

"enabled": true,

"updatePolicy": "daily",

"checksumPolicy": "warn"

},

"mirroredRepositories": [],

"repositoryManager": false

},

{

"id": "local",

"type": "default",

"url": "file:///root/.m2/repository",

"releasePolicy": {

"enabled": true,

"updatePolicy": "daily",

"checksumPolicy": "warn"

},

"snapshotPolicy": {

"enabled": true,

"updatePolicy": "daily",

"checksumPolicy": "warn"

},

"mirroredRepositories": [],

"repositoryManager": false

}

]

}

hadoop-conf

├── capacity-scheduler.xml

├── commons-logging.properties

├── configuration.xsl

├── container-executor.cfg

├── core-site.xml

├── dfs.exclude

├── hadoop-env.cmd

├── hadoop-env.sh

├── hadoop-metrics2.properties

├── hadoop-metrics.properties

├── hadoop-policy.xml

├── hdfs\_dn\_jaas.conf

├── hdfs\_nn\_jaas.conf

├── hdfs-site.xml

├── health\_check

├── kms-acls.xml

├── kms-env.sh

├── kms-log4j.properties

├── kms-site.xml

├── log4j.properties

├── mapred-env.cmd

├── mapred-env.sh

├── mapred\_jaas.conf

├── mapred-queues.xml.template

├── mapred-site.xml

├── mapred-site.xml.template

├── secure

│   └── ssl-client.xml

├── slaves

├── ssl-client.xml

├── ssl-client.xml.example

├── ssl-server.xml

├── ssl-server.xml.example

├── taskcontroller.cfg

├── task-log4j.properties

├── topology\_mappings.data

├── topology\_script.py

├── yarn\_ats\_jaas.conf

├── yarn-env.cmd

├── yarn-env.sh

├── yarn\_jaas.conf

├── yarn\_nm\_jaas.conf

└── yarn-site.xml

docker build -t zeppelin-docker .

docker tag zeppelin-docker:latest h1.spark.org:5000/zeppelin-docker:latest

docker push h1.spark.org:5000/zeppelin-docker:latest

## Copy hadoop-conf đến persistent volume

kubectl cp hadoop-conf nfs-server-pod:/exports/

kubectl exec -it nfs-server-pod sh

# ls -la /exports/hadoop-conf

## Copy keytab đến secret

kubectl create secret generic dp-sec --from-file=keytab=/home/vagrant/kerberos/data/keytab

## Start zeppelin

vi zeppelin.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: zeppelin

labels:

app: zeppelin

spec:

replicas: 1

selector:

matchLabels:

app: zeppelin

template:

metadata:

labels:

app: zeppelin

spec:

containers:

- name: zeppelin

image: h1.spark.org:5000/zeppelin-docker:latest

imagePullPolicy: Always

resources:

requests:

cpu: 100m

command: ["/start-zeppelin"]

#command: ["sleep", " infinity"]

env:

- name: KRB\_USERNAME

value: tuanndd@HDFS.ORG

- name: SPARK\_MASTER\_URL

value: spark://spark-master:7077

ports:

- containerPort: 8080

volumeMounts:

- name: vol-conf

mountPath: /opt/hadoop-conf

subPath: hadoop-conf

- name: vol-sec

mountPath: /etc/keytab

subPath: tuanndd.keytab

restartPolicy: Always

imagePullSecrets:

- name: local-registry

volumes:

- name: vol-conf

persistentVolumeClaim:

claimName: nfs

- name: vol-sec

secret:

secretName: dp-sec

items:

- key: keytab

path: tuanndd.keytab

---

kind: Service

apiVersion: v1

metadata:

name: zeppelin

labels:

app: zeppelin

spec:

ports:

- port: 80

targetPort: 8080

selector:

app: zeppelin

type: NodePort

kubectl create -f zeppelin.yaml

kubectl get pod -o wide

kubectl get svc

## Test zepplin

<http://h4.spark.org:31319>

SPARK>

sc

sc.setLogLevel("INFO")

sc.getConf.getAll.foreach(println)

sc.textFile("alluxio://alluxio-master:19998/hdfs/tmp/LICENSE").count()

val NUM\_SAMPLES = 1000000

val count1 = sc.parallelize(1 to NUM\_SAMPLES).filter { \_ =>

val x = math.random

val y = math.random

x\*x + y\*y < 1

}.count()

println(s"Pi is roughly ${4.0 \* count1 / NUM\_SAMPLES}")

HIVE>

Tạo “jdbc” interpreter với cầu hình:

zeppelin.jdbc.auth.kerberos.proxy.enable=false

zeppelin.jdbc.auth.type=KERBEROS

zeppelin.jdbc.keytab.location=/etc/keytab

zeppelin.jdbc.principal=tuanndd@HDFS.ORG

hive.driver=org.apache.hive.jdbc.HiveDriver

hive.url=jdbc:hive2://h2.local.org:10000/default;principal=hive/\_HOST@HDFS.ORG

Test query:

%jdbc(hive)

show tables

%jdbc(hive)

SELECT d.driverId, d.name, t.total\_hours, t.total\_miles from drivers d

JOIN (SELECT driverId, sum(hours\_logged)total\_hours, sum(miles\_logged)total\_miles FROM timesheet GROUP BY driverId ) t

ON (d.driverId = t.driverId)

## Stop zeppelin (optional)

kubectl delete –f zeppelin.yaml

# Setup airflow

## Build airflow docker

docker build -t airflow-docker .

docker tag airflow-docker:latest h1.spark.org:5000/airflow-docker:latest

docker push h1.spark.org:5000/airflow-docker:latest

|  |  |
| --- | --- |
| Dockerfile | #!/bin/bash  docker build -t airflow-docker .  docker tag airflow-docker:latest h1.spark.org:5000/airflow-docker:latest  docker push h1.spark.org:5000/airflow-docker:latest  [vagrant@h1 airflow-docker]$ cat Dockerfile  # inspired by https://hub.docker.com/r/wongnai/spark-standalone/  FROM centos:7.4.1708  RUN yum -y install epel-release java-1.8.0-openjdk-devel krb5-workstation which  RUN yum -y install net-tools nc vim nano  RUN yum -y install bzip2  RUN ln -sf /usr/share/zoneinfo/Asia/Ho\_Chi\_Minh /etc/localtime  RUN yum -y groupinstall 'Development Tools'  RUN yum -y install python-devel python-pip postgresql-devel  #RUN pip install -U pip setuptools wheel  #RUN pip install apache-airflow[kerberos]  ADD Miniconda3-latest-Linux-x86\_64.sh /  ADD requirements.txt /  RUN bash /Miniconda3-latest-Linux-x86\_64.sh -b -p /opt/anaconda3  RUN rm -rf /Miniconda3-latest-Linux-x86\_64.sh  RUN source /opt/anaconda3/bin/activate && \  conda create -n airflow python=3 && \  conda activate airflow && \  pip install -U pip && \  pip install -U pip setuptools wheel && \  pip install -r requirements.txt  #ENV AIRFLOW\_GPL\_UNIDECODE yes  #RUN yum -y install postgresql-devel  #COPY requirements.txt .  #RUN pip install -r requirements.txt  ADD common.sh start.sh create-admin-user.py /  #ADD spark-defaults.conf /opt/spark/conf/spark-defaults.conf  RUN chmod +x /common.sh /start.sh /create-admin-user.py  #ARG AIRFLOW\_HOME=/usr/local/airflow  #COPY airflow.cfg ${AIRFLOW\_HOME}/airflow.cfg  COPY airflow.cfg /usr/local/airflow/airflow.cfg  #RUN useradd -ms /bin/bash -d ${AIRFLOW\_HOME} airflow  #RUN chown -R airflow: ${AIRFLOW\_HOME}  #USER airflow  ADD hadoop-2.7.3 /opt/hadoop  #WORKDIR ${AIRFLOW\_HOME}  WORKDIR /usr/local/airflow  ENV PATH=/opt/hadoop/bin:/opt/anaconda3/bin/:$PATH  #ENV PATH=/opt/spark/bin:/opt/spark/sbin:$PATH  #EXPOSE 6066 7077 7078 8080 8081  #WORKDIR /  #ENTRYPOINT ["/opt/spark/bin/spark-class"]  #CMD ["org.apache.spark.deploy.master.Master", "--ip spark-master", "--port 7077", "--webui-port 8080"] |
| requirements.txt | pytz  pyOpenSSL  ndg-httpsclient  pyasn1  redis  celery  psycopg2-binary  apache-airflow[celery,redis,postgres,crypto,password] |
| common.sh | #!/bin/bash  cat > /etc/resolv.conf << EOF  nameserver 10.96.0.10  search default.svc.cluster.local svc.cluster.local cluster.local  EOF  #unset POSTGRES\_PORT  #unset REDIS\_PORT  unset FLOWER\_PORT  #kinit -kt /etc/keytab $KRB\_USERNAME  source /opt/anaconda3/bin/activate airflow |
| start.sh | #!/usr/bin/env bash  . /common.sh  TRY\_LOOP="20"  : "${REDIS\_HOST:="redis"}"  : "${REDIS\_PORT:="6379"}"  : "${REDIS\_PASSWORD:=""}"  : "${POSTGRES\_HOST:="postgres"}"  : "${POSTGRES\_PORT:="5432"}"  : "${POSTGRES\_USER:="airflow"}"  : "${POSTGRES\_PASSWORD:="airflow"}"  : "${POSTGRES\_DB:="airflow"}"  # Defaults and back-compat  : "${AIRFLOW\_\_CORE\_\_FERNET\_KEY:=${FERNET\_KEY:=$(python -c "from cryptography.fernet import Fernet; FERNET\_KEY = Fernet.generate\_key().decode(); print(FERNET\_KEY)")}}"  : "${AIRFLOW\_\_CORE\_\_EXECUTOR:=${EXECUTOR:-Sequential}Executor}"  export \  AIRFLOW\_\_CELERY\_\_BROKER\_URL \  AIRFLOW\_\_CELERY\_\_RESULT\_BACKEND \  AIRFLOW\_\_CORE\_\_EXECUTOR \  AIRFLOW\_\_CORE\_\_FERNET\_KEY \  AIRFLOW\_\_CORE\_\_LOAD\_EXAMPLES \  AIRFLOW\_\_CORE\_\_SQL\_ALCHEMY\_CONN \  # Load DAGs exemples (default: Yes)  if [[ -z "$AIRFLOW\_\_CORE\_\_LOAD\_EXAMPLES" && "${LOAD\_EX:=n}" == n ]]  then  AIRFLOW\_\_CORE\_\_LOAD\_EXAMPLES=False  fi  # Install custom python package if requirements.txt is present  #if [ -e "/requirements.txt" ]; then  # $(which pip) install --user -r /requirements.txt  #fi  if [ -n "$REDIS\_PASSWORD" ]; then  REDIS\_PREFIX=:${REDIS\_PASSWORD}@  else  REDIS\_PREFIX=  fi  wait\_for\_port() {  local name="$1" host="$2" port="$3"  local j=0  while ! nc -z "$host" "$port" >/dev/null 2>&1 < /dev/null; do  j=$((j+1))  if [ $j -ge $TRY\_LOOP ]; then  echo >&2 "$(date) - $host:$port still not reachable, giving up"  exit 1  fi  echo "$(date) - waiting for $name... $j/$TRY\_LOOP"  sleep 5  done  }  if [ "$AIRFLOW\_\_CORE\_\_EXECUTOR" != "SequentialExecutor" ]; then  AIRFLOW\_\_CORE\_\_SQL\_ALCHEMY\_CONN="postgresql+psycopg2://$POSTGRES\_USER:$POSTGRES\_PASSWORD@$POSTGRES\_HOST:$POSTGRES\_PORT/$POSTGRES\_DB"  AIRFLOW\_\_CELERY\_\_RESULT\_BACKEND="db+postgresql://$POSTGRES\_USER:$POSTGRES\_PASSWORD@$POSTGRES\_HOST:$POSTGRES\_PORT/$POSTGRES\_DB"  wait\_for\_port "Postgres" "$POSTGRES\_HOST" "$POSTGRES\_PORT"  fi  if [ "$AIRFLOW\_\_CORE\_\_EXECUTOR" = "CeleryExecutor" ]; then  AIRFLOW\_\_CELERY\_\_BROKER\_URL="redis://$REDIS\_PREFIX$REDIS\_HOST:$REDIS\_PORT/1"  wait\_for\_port "Redis" "$REDIS\_HOST" "$REDIS\_PORT"  fi  case "$1" in  webserver)  airflow initdb  python /create-admin-user.py  if [ "$AIRFLOW\_\_CORE\_\_EXECUTOR" = "LocalExecutor" ]; then  # With the "Local" executor it should all run in one container.  airflow scheduler &  fi  exec airflow webserver  ;;  worker|scheduler)  # To give the webserver time to run initdb.  sleep 10  airflow kerberos &  exec airflow "$@"  ;;  flower)  sleep 10  exec airflow "$@"  ;;  version)  exec airflow "$@"  ;;  \*)  # The command is something like bash, not an airflow subcommand. Just run it in the right environment.  exec "$@"  ;;  esac |
| create-admin-user.py | import os  import airflow  from airflow import models, settings  from airflow.contrib.auth.backends.password\_auth import PasswordUser  user = PasswordUser(models.User())  user.username = os.environ['ADMIN\_USERNAME']  user.email = os.environ['ADMIN\_EMAIL']  user.password = os.environ['ADMIN\_PASSWORD']  session = settings.Session()  session.add(user)  session.commit()  session.close() |
| airflow.cfg | [core]  # The home folder for airflow, default is ~/airflow  #airflow\_home = /usr/local/airflow  # The folder where your airflow pipelines live, most likely a  # subfolder in a code repository  # This path must be absolute  dags\_folder = /usr/local/airflow/dags  # The folder where airflow should store its log files  # This path must be absolute  base\_log\_folder = /usr/local/airflow/logs  # Airflow can store logs remotely in AWS S3, Google Cloud Storage or Elastic Search.  # Users must supply an Airflow connection id that provides access to the storage  # location. If remote\_logging is set to true, see UPDATING.md for additional  # configuration requirements.  remote\_logging = False  remote\_log\_conn\_id =  remote\_base\_log\_folder =  encrypt\_s3\_logs = False  # Logging level  logging\_level = INFO  fab\_logging\_level = WARN  # Logging class  # Specify the class that will specify the logging configuration  # This class has to be on the python classpath  # logging\_config\_class = my.path.default\_local\_settings.LOGGING\_CONFIG  logging\_config\_class =  # Log format  # we need to escape the curly braces by adding an additional curly brace  log\_format = [%%(asctime)s] {{%%(filename)s:%%(lineno)d}} %%(levelname)s - %%(message)s  simple\_log\_format = %%(asctime)s %%(levelname)s - %%(message)s  # Log filename format  # we need to escape the curly braces by adding an additional curly brace  log\_filename\_template = {{ ti.dag\_id }}/{{ ti.task\_id }}/{{ ts }}/{{ try\_number }}.log  log\_processor\_filename\_template = {{ filename }}.log  dag\_processor\_manager\_log\_location = /usr/local/airflow/logs/dag\_processor\_manager/dag\_processor\_manager.log  # Hostname by providing a path to a callable, which will resolve the hostname  hostname\_callable = socket:getfqdn  # Default timezone in case supplied date times are naive  # can be utc (default), system, or any IANA timezone string (e.g. Europe/Amsterdam)  default\_timezone = utc  # The executor class that airflow should use. Choices include  # SequentialExecutor, LocalExecutor, CeleryExecutor, DaskExecutor  executor = CeleryExecutor  # The SqlAlchemy connection string to the metadata database.  # SqlAlchemy supports many different database engine, more information  # their website  # sql\_alchemy\_conn = sqlite:////tmp/airflow.db  #sql\_alchemy\_conn=postgresql://airflow:airflow@postgres/airflow  # If SqlAlchemy should pool database connections.  sql\_alchemy\_pool\_enabled = True  # The encoding for the databases  sql\_engine\_encoding = utf-8  # The SqlAlchemy pool size is the maximum number of database connections  # in the pool. 0 indicates no limit.  sql\_alchemy\_pool\_size = 5  # The SqlAlchemy pool recycle is the number of seconds a connection  # can be idle in the pool before it is invalidated. This config does  # not apply to sqlite. If the number of DB connections is ever exceeded,  # a lower config value will allow the system to recover faster.  sql\_alchemy\_pool\_recycle = 1800  # How many seconds to retry re-establishing a DB connection after  # disconnects. Setting this to 0 disables retries.  sql\_alchemy\_reconnect\_timeout = 300  # The schema to use for the metadata database  # SqlAlchemy supports databases with the concept of multiple schemas.  sql\_alchemy\_schema =  # The amount of parallelism as a setting to the executor. This defines  # the max number of task instances that should run simultaneously  # on this airflow installation  parallelism = 32  # The number of task instances allowed to run concurrently by the scheduler  dag\_concurrency = 16  # Are DAGs paused by default at creation  dags\_are\_paused\_at\_creation = True  # When not using pools, tasks are run in the "default pool",  # whose size is guided by this config element  non\_pooled\_task\_slot\_count = 128  # The maximum number of active DAG runs per DAG  max\_active\_runs\_per\_dag = 16  # Whether to load the examples that ship with Airflow. It's good to  # get started, but you probably want to set this to False in a production  # environment  load\_examples = False  # Where your Airflow plugins are stored  plugins\_folder = /usr/local/airflow/plugins  # Secret key to save connection passwords in the db  fernet\_key = $FERNET\_KEY  # Whether to disable pickling dags  donot\_pickle = False  # How long before timing out a python file import while filling the DagBag  dagbag\_import\_timeout = 30  # The class to use for running task instances in a subprocess  #task\_runner = BashTaskRunner  task\_runner = StandardTaskRunner  # If set, tasks without a `run\_as\_user` argument will be run with this user  # Can be used to de-elevate a sudo user running Airflow when executing tasks  default\_impersonation =  # What security module to use (for example kerberos):  security =  # If set to False enables some unsecure features like Charts and Ad Hoc Queries.  # In 2.0 will default to True.  secure\_mode = False  # Turn unit test mode on (overwrites many configuration options with test  # values at runtime)  unit\_test\_mode = False  # Name of handler to read task instance logs.  # Default to use task handler.  task\_log\_reader = task  # Whether to enable pickling for xcom (note that this is insecure and allows for  # RCE exploits). This will be deprecated in Airflow 2.0 (be forced to False).  enable\_xcom\_pickling = True  # When a task is killed forcefully, this is the amount of time in seconds that  # it has to cleanup after it is sent a SIGTERM, before it is SIGKILLED  killed\_task\_cleanup\_time = 60  # Whether to override params with dag\_run.conf. If you pass some key-value pairs through `airflow backfill -c` or  # `airflow trigger\_dag -c`, the key-value pairs will override the existing ones in params.  dag\_run\_conf\_overrides\_params = False  # Worker initialisation check to validate Metadata Database connection  worker\_precheck = False  [cli]  # In what way should the cli access the API. The LocalClient will use the  # database directly, while the json\_client will use the api running on the  # webserver  api\_client = airflow.api.client.local\_client  # If you set web\_server\_url\_prefix, do NOT forget to append it here, ex:  # endpoint\_url = http://localhost:8080/myroot  # So api will look like: http://localhost:8080/myroot/api/experimental/...  endpoint\_url = http://localhost:8080  [api]  # How to authenticate users of the API  auth\_backend = airflow.api.auth.backend.default  [lineage]  # what lineage backend to use  backend =  [atlas]  sasl\_enabled = False  host =  port = 21000  username =  password =  [operators]  # The default owner assigned to each new operator, unless  # provided explicitly or passed via `default\_args`  default\_owner = Airflow  default\_cpus = 1  default\_ram = 512  default\_disk = 512  default\_gpus = 0  [hive]  # Default mapreduce queue for HiveOperator tasks  default\_hive\_mapred\_queue =  # Template for mapred\_job\_name in HiveOperator, supports the following named parameters:  # hostname, dag\_id, task\_id, execution\_date  mapred\_job\_name\_template = Airflow HiveOperator task for {hostname}.{dag\_id}.{task\_id}.{execution\_date}  [webserver]  # The base url of your website as airflow cannot guess what domain or  # cname you are using. This is used in automated emails that  # airflow sends to point links to the right web server  base\_url = http://localhost:8080  # The ip specified when starting the web server  web\_server\_host = 0.0.0.0  # The port on which to run the web server  web\_server\_port = 8080  # Paths to the SSL certificate and key for the web server. When both are  # provided SSL will be enabled. This does not change the web server port.  web\_server\_ssl\_cert =  web\_server\_ssl\_key =  # Number of seconds the webserver waits before killing gunicorn master that doesn't respond  web\_server\_master\_timeout = 120  # Number of seconds the gunicorn webserver waits before timing out on a worker  web\_server\_worker\_timeout = 120  # Number of workers to refresh at a time. When set to 0, worker refresh is  # disabled. When nonzero, airflow periodically refreshes webserver workers by  # bringing up new ones and killing old ones.  worker\_refresh\_batch\_size = 1  # Number of seconds to wait before refreshing a batch of workers.  worker\_refresh\_interval = 30  # Secret key used to run your flask app  secret\_key = temporary\_key  # Number of workers to run the Gunicorn web server  workers = 4  # The worker class gunicorn should use. Choices include  # sync (default), eventlet, gevent  worker\_class = sync  # Log files for the gunicorn webserver. '-' means log to stderr.  access\_logfile = -  error\_logfile = -  # Expose the configuration file in the web server  # This is only applicable for the flask-admin based web UI (non FAB-based).  # In the FAB-based web UI with RBAC feature,  # access to configuration is controlled by role permissions.  expose\_config = True  # Set to true to turn on authentication:  # https://airflow.incubator.apache.org/security.html#web-authentication  authenticate = True  auth\_backend = airflow.contrib.auth.backends.password\_auth  # Filter the list of dags by owner name (requires authentication to be enabled)  filter\_by\_owner = False  # Filtering mode. Choices include user (default) and ldapgroup.  # Ldap group filtering requires using the ldap backend  #  # Note that the ldap server needs the "memberOf" overlay to be set up  # in order to user the ldapgroup mode.  owner\_mode = user  # Default DAG view. Valid values are:  # tree, graph, duration, gantt, landing\_times  dag\_default\_view = tree  # Default DAG orientation. Valid values are:  # LR (Left->Right), TB (Top->Bottom), RL (Right->Left), BT (Bottom->Top)  dag\_orientation = LR  # Puts the webserver in demonstration mode; blurs the names of Operators for  # privacy.  demo\_mode = False  # The amount of time (in secs) webserver will wait for initial handshake  # while fetching logs from other worker machine  log\_fetch\_timeout\_sec = 5  # By default, the webserver shows paused DAGs. Flip this to hide paused  # DAGs by default  hide\_paused\_dags\_by\_default = False  # Consistent page size across all listing views in the UI  page\_size = 100  # Use FAB-based webserver with RBAC feature  rbac = False  # Define the color of navigation bar  navbar\_color = #007A87  # Default dagrun to show in UI  default\_dag\_run\_display\_number = 25  [email]  email\_backend = airflow.utils.email.send\_email\_smtp  [smtp]  # If you want airflow to send emails on retries, failure, and you want to use  # the airflow.utils.email.send\_email\_smtp function, you have to configure an  # smtp server here  smtp\_host = localhost  smtp\_starttls = True  smtp\_ssl = False  # Uncomment and set the user/pass settings if you want to use SMTP AUTH  # smtp\_user = airflow  # smtp\_password = airflow  smtp\_port = 25  smtp\_mail\_from = airflow@example.com  [celery]  # This section only applies if you are using the CeleryExecutor in  # [core] section above  # The app name that will be used by celery  celery\_app\_name = airflow.executors.celery\_executor  # The concurrency that will be used when starting workers with the  # "airflow worker" command. This defines the number of task instances that  # a worker will take, so size up your workers based on the resources on  # your worker box and the nature of your tasks  worker\_concurrency = 16  # When you start an airflow worker, airflow starts a tiny web server  # subprocess to serve the workers local log files to the airflow main  # web server, who then builds pages and sends them to users. This defines  # the port on which the logs are served. It needs to be unused, and open  # visible from the main web server to connect into the workers.  worker\_log\_server\_port = 8793  # The Celery broker URL. Celery supports RabbitMQ, Redis and experimentally  # a sqlalchemy database. Refer to the Celery documentation for more  # information.  #broker\_url = redis://redis:6379/1  # Another key Celery setting  #result\_backend = db+postgresql://airflow:airflow@postgres/airflow  # Celery Flower is a sweet UI for Celery. Airflow has a shortcut to start  # it `airflow flower`. This defines the IP that Celery Flower runs on  flower\_host = 0.0.0.0  # The root URL for Flower  # Ex: flower\_url\_prefix = /flower  flower\_url\_prefix =  # This defines the port that Celery Flower runs on  flower\_port = 5555  # Default queue that tasks get assigned to and that worker listen on.  default\_queue = default  # Import path for celery configuration options  celery\_config\_options = airflow.config\_templates.default\_celery.DEFAULT\_CELERY\_CONFIG  # In case of using SSL  ssl\_active = False  ssl\_key =  ssl\_cert =  ssl\_cacert =  [celery\_broker\_transport\_options]  # This section is for specifying options which can be passed to the  # underlying celery broker transport. See:  # http://docs.celeryproject.org/en/latest/userguide/configuration.html#std:setting-broker\_transport\_options  # The visibility timeout defines the number of seconds to wait for the worker  # to acknowledge the task before the message is redelivered to another worker.  # Make sure to increase the visibility timeout to match the time of the longest  # ETA you're planning to use.  #  # visibility\_timeout is only supported for Redis and SQS celery brokers.  # See:  # http://docs.celeryproject.org/en/master/userguide/configuration.html#std:setting-broker\_transport\_options  #  #visibility\_timeout = 21600  [dask]  # This section only applies if you are using the DaskExecutor in  # [core] section above  # The IP address and port of the Dask cluster's scheduler.  cluster\_address = 127.0.0.1:8786  # TLS/ SSL settings to access a secured Dask scheduler.  tls\_ca =  tls\_cert =  tls\_key =  [scheduler]  # Task instances listen for external kill signal (when you clear tasks  # from the CLI or the UI), this defines the frequency at which they should  # listen (in seconds).  job\_heartbeat\_sec = 5  # The scheduler constantly tries to trigger new tasks (look at the  # scheduler section in the docs for more information). This defines  # how often the scheduler should run (in seconds).  scheduler\_heartbeat\_sec = 5  # after how much time should the scheduler terminate in seconds  # -1 indicates to run continuously (see also num\_runs)  run\_duration = -1  # after how much time a new DAGs should be picked up from the filesystem  min\_file\_process\_interval = 0  # How often (in seconds) to scan the DAGs directory for new files. Default to 5 minutes.  dag\_dir\_list\_interval = 300  # How often should stats be printed to the logs  print\_stats\_interval = 30  child\_process\_log\_directory = /usr/local/airflow/logs/scheduler  # Local task jobs periodically heartbeat to the DB. If the job has  # not heartbeat in this many seconds, the scheduler will mark the  # associated task instance as failed and will re-schedule the task.  scheduler\_zombie\_task\_threshold = 300  # Turn off scheduler catchup by setting this to False.  # Default behavior is unchanged and  # Command Line Backfills still work, but the scheduler  # will not do scheduler catchup if this is False,  # however it can be set on a per DAG basis in the  # DAG definition (catchup)  catchup\_by\_default = True  # This changes the batch size of queries in the scheduling main loop.  # This depends on query length limits and how long you are willing to hold locks.  # 0 for no limit  max\_tis\_per\_query = 512  # Statsd (https://github.com/etsy/statsd) integration settings  statsd\_on = False  statsd\_host = localhost  statsd\_port = 8125  statsd\_prefix = airflow  # The scheduler can run multiple threads in parallel to schedule dags.  # This defines how many threads will run.  max\_threads = 2  authenticate = False  # Turn off scheduler use of cron intervals by setting this to False.  # DAGs submitted manually in the web UI or with trigger\_dag will still run.  use\_job\_schedule = True  [ldap]  # set this to ldaps://<your.ldap.server>:<port>  uri =  user\_filter = objectClass=\*  user\_name\_attr = uid  group\_member\_attr = memberOf  superuser\_filter =  data\_profiler\_filter =  bind\_user = cn=Manager,dc=example,dc=com  bind\_password = insecure  basedn = dc=example,dc=com  cacert = /etc/ca/ldap\_ca.crt  search\_scope = LEVEL  [mesos]  # Mesos master address which MesosExecutor will connect to.  master = localhost:5050  # The framework name which Airflow scheduler will register itself as on mesos  framework\_name = Airflow  # Number of cpu cores required for running one task instance using  # 'airflow run <dag\_id> <task\_id> <execution\_date> --local -p <pickle\_id>'  # command on a mesos slave  task\_cpu = 1  # Memory in MB required for running one task instance using  # 'airflow run <dag\_id> <task\_id> <execution\_date> --local -p <pickle\_id>'  # command on a mesos slave  task\_memory = 256  # Enable framework checkpointing for mesos  # See http://mesos.apache.org/documentation/latest/slave-recovery/  checkpoint = False  # Failover timeout in milliseconds.  # When checkpointing is enabled and this option is set, Mesos waits  # until the configured timeout for  # the MesosExecutor framework to re-register after a failover. Mesos  # shuts down running tasks if the  # MesosExecutor framework fails to re-register within this timeframe.  # failover\_timeout = 604800  # Enable framework authentication for mesos  # See http://mesos.apache.org/documentation/latest/configuration/  authenticate = False  # Mesos credentials, if authentication is enabled  # default\_principal = admin  # default\_secret = admin  # Optional Docker Image to run on slave before running the command  # This image should be accessible from mesos slave i.e mesos slave  # should be able to pull this docker image before executing the command.  # docker\_image\_slave = puckel/docker-airflow  [kerberos]  #ccache = /tmp/krb5cc\_0  # gets augmented with fqdn  #principal = tuanndd@HDFS.ORG  reinit\_frequency = 3600  kinit\_path = kinit  #keytab = /etc/keytab  [github\_enterprise]  api\_rev = v3  [admin]  # UI to hide sensitive variable fields when set to True  hide\_sensitive\_variable\_fields = True  [elasticsearch]  elasticsearch\_host =  # we need to escape the curly braces by adding an additional curly brace  elasticsearch\_log\_id\_template = {dag\_id}-{task\_id}-{execution\_date}-{try\_number}  elasticsearch\_end\_of\_log\_mark = end\_of\_log  [kubernetes]  # The repository, tag and imagePullPolicy of the Kubernetes Image for the Worker to Run  worker\_container\_repository =  worker\_container\_tag =  worker\_container\_image\_pull\_policy = IfNotPresent  worker\_dags\_folder =  # If True (default), worker pods will be deleted upon termination  delete\_worker\_pods = True  # The Kubernetes namespace where airflow workers should be created. Defaults to `default`  namespace = default  # The name of the Kubernetes ConfigMap Containing the Airflow Configuration (this file)  airflow\_configmap =  # For either git sync or volume mounted DAGs, the worker will look in this subpath for DAGs  dags\_volume\_subpath =  # For DAGs mounted via a volume claim (mutually exclusive with volume claim)  dags\_volume\_claim =  # For volume mounted logs, the worker will look in this subpath for logs  logs\_volume\_subpath =  # A shared volume claim for the logs  logs\_volume\_claim =  # Git credentials and repository for DAGs mounted via Git (mutually exclusive with volume claim)  git\_repo =  git\_branch =  git\_user =  git\_password =  git\_subpath =  # For cloning DAGs from git repositories into volumes: https://github.com/kubernetes/git-sync  git\_sync\_container\_repository = gcr.io/google-containers/git-sync-amd64  git\_sync\_container\_tag = v2.0.5  git\_sync\_init\_container\_name = git-sync-clone  # The name of the Kubernetes service account to be associated with airflow workers, if any.  # Service accounts are required for workers that require access to secrets or cluster resources.  # See the Kubernetes RBAC documentation for more:  # https://kubernetes.io/docs/admin/authorization/rbac/  worker\_service\_account\_name =  # Any image pull secrets to be given to worker pods, If more than one secret is  # required, provide a comma separated list: secret\_a,secret\_b  image\_pull\_secrets =  # GCP Service Account Keys to be provided to tasks run on Kubernetes Executors  # Should be supplied in the format: key-name-1:key-path-1,key-name-2:key-path-2  gcp\_service\_account\_keys =  # Use the service account kubernetes gives to pods to connect to kubernetes cluster.  # It's intended for clients that expect to be running inside a pod running on kubernetes.  # It will raise an exception if called from a process not running in a kubernetes environment.  in\_cluster = True  [kubernetes\_node\_selectors]  # The Key-value pairs to be given to worker pods.  # The worker pods will be scheduled to the nodes of the specified key-value pairs.  # Should be supplied in the format: key = value  [kubernetes\_secrets]  # The scheduler mounts the following secrets into your workers as they are launched by the  # scheduler. You may define as many secrets as needed and the kubernetes launcher will parse the  # defined secrets and mount them as secret environment variables in the launched workers.  # Secrets in this section are defined as follows  # <environment\_variable\_mount> = <kubernetes\_secret\_object>:<kubernetes\_secret\_key>  #  # For example if you wanted to mount a kubernetes secret key named `postgres\_password` from the  # kubernetes secret object `airflow-secret` as the environment variable `POSTGRES\_PASSWORD` into  # your workers you would follow the following format:  # POSTGRES\_PASSWORD = airflow-secret:postgres\_credentials  #  # Additionally you may override worker airflow settings with the AIRFLOW\_\_<SECTION>\_\_<KEY>  # formatting as supported by airflow normally. |
|  |  |

|  |  |
| --- | --- |
| postgres.yaml | apiVersion: extensions/v1beta1  kind: Deployment  metadata:  name: postgres  labels:  app: postgres  spec:  replicas: 1  template:  metadata:  labels:  app: postgres  spec:  containers:  - name: postgres  image: postgres:9.6  imagePullPolicy: "IfNotPresent"  ports:  - containerPort: 5432  env:  - name: POSTGRES\_USER  value: airflow  - name: POSTGRES\_PASSWORD  value: airflow  - name: POSTGRES\_DB  value: airflow  volumeMounts:  - mountPath: /var/lib/postgresql/data  name: postgredb  volumes:  - name: postgredb  emptyDir: {}  ---  kind: Service  apiVersion: v1  metadata:  name: postgres  labels:  app: postgres  spec:  ports:  - port: 5432  targetPort: 5432  name: postgres  selector:  app: postgres |
| redis.yaml | apiVersion: extensions/v1beta1  kind: Deployment  metadata:  name: redis  labels:  app: redis  spec:  replicas: 1  template:  metadata:  labels:  app: redis  spec:  containers:  - name: redis  image: redis:3.2.7  imagePullPolicy: "IfNotPresent"  command: ["redis-server", "--requirepass", "$(REDIS\_PASS)"]  ports:  - containerPort: 6379  env:  - name: REDIS\_PASS  value: password  ---  kind: Service  apiVersion: v1  metadata:  name: redis  labels:  app: redis  spec:  ports:  - port: 6379  targetPort: 6379  name: redis  selector:  app: redis |
| webserver.yaml | apiVersion: apps/v1  kind: Deployment  metadata:  name: webserver  labels:  app: webserver  spec:  replicas: 1  selector:  matchLabels:  app: webserver  template:  metadata:  labels:  app: webserver  spec:  containers:  - name: webserver  image: h1.spark.org:5000/airflow-docker:latest  imagePullPolicy: Always  resources:  requests:  cpu: 100m  memory: 1G  command: ["/start.sh", "webserver"]  #command: ["sleep", "infinity"]  volumeMounts:  - mountPath: /usr/local/airflow/dags  name: vol-nfs  subPath: dags  - mountPath: /usr/local/airflow/logs  name: vol-nfs  subPath: logs  - mountPath: /etc/krb5.conf  name: vol-nfs  subPath: krb5.conf  - name: vol-nfs  mountPath: /opt/hadoop-conf  subPath: hadoop-conf  - name: vol-sec  mountPath: /etc/keytab  subPath: tuanndd.keytab  env:  - name: HADOOP\_HOME  value: /opt/hadoop  - name: HADOOP\_CONF\_DIR  value: /opt/hadoop-conf  - name: JAVA\_HOME  value: /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.212.b04-0.el7\_6.x86\_64  - name: KRB\_USERNAME  value: tuanndd@HDFS.ORG  - name: ADMIN\_USERNAME  value: admin  - name: ADMIN\_EMAIL  value: admin@example.com  - name: ADMIN\_PASSWORD  value: password  - name: AIRFLOW\_HOME  value: /usr/local/airflow  - name: LOAD\_EX  value: "n"  - name: FERNET\_KEY  value: "46BKJoQYlPPOexq0OhDZnIlNepKFf87WFwLbfzqDDho="  - name: EXECUTOR  value: Celery  - name: REDIS\_HOST  value: redis  - name: REDIS\_PORT  value: "6379"  - name: REDIS\_PASSWORD  value: password  - name: POSTGRES\_HOST  value: postgres  - name: POSTGRES\_PORT  value: "5432"  - name: POSTGRES\_DB  value: airflow  - name: POSTGRES\_USER  value: airflow  - name: POSTGRES\_PASSWORD  value: airflow  ports:  - containerPort: 8080  name: http  restartPolicy: Always  imagePullSecrets:  - name: local-registry  volumes:  - name: vol-sec  secret:  secretName: dp-sec  items:  - key: keytab  path: tuanndd.keytab  - name: vol-nfs  persistentVolumeClaim:  claimName: nfs  ---  kind: Service  apiVersion: v1  metadata:  name: webserver  labels:  app: webserver  spec:  ports:  - port: 8080  targetPort: 8080  name: http  selector:  app: webserver  type: NodePort |
| flower.yaml | apiVersion: apps/v1  kind: Deployment  metadata:  name: flower  labels:  app: flower  spec:  replicas: 1  selector:  matchLabels:  app: flower  template:  metadata:  labels:  app: flower  spec:  containers:  - name: flower  image: h1.spark.org:5000/airflow-docker:latest  imagePullPolicy: Always  resources:  requests:  cpu: 100m  memory: 1G  command: ["/start.sh", "flower"]  #command: ["sleep", "infinity"]  volumeMounts:  - mountPath: /usr/local/airflow/dags  name: dags  subPath: dags  env:  - name: AIRFLOW\_HOME  value: /usr/local/airflow  - name: EXECUTOR  value: Celery  - name: REDIS\_HOST  value: redis  - name: REDIS\_PORT  value: "6379"  - name: REDIS\_PASSWORD  value: password  - name: POSTGRES\_HOST  value: postgres  - name: POSTGRES\_PORT  value: "5432"  - name: POSTGRES\_DB  value: airflow  - name: POSTGRES\_USER  value: airflow  - name: POSTGRES\_PASSWORD  value: airflow  ports:  - containerPort: 5555  name: http  restartPolicy: Always  imagePullSecrets:  - name: local-registry  volumes:  - name: dags  persistentVolumeClaim:  claimName: nfs  ---  kind: Service  apiVersion: v1  metadata:  name: flower  labels:  app: flower  spec:  ports:  - port: 5555  targetPort: 5555  name: http  selector:  app: flower  type: NodePort |
| scheduler.yaml | apiVersion: apps/v1  kind: Deployment  metadata:  name: scheduler  labels:  app: scheduler  spec:  replicas: 1  selector:  matchLabels:  app: scheduler  template:  metadata:  labels:  app: scheduler  spec:  containers:  - name: scheduler  image: h1.spark.org:5000/airflow-docker:latest  imagePullPolicy: Always  resources:  requests:  cpu: 100m  memory: 1G  command: ["/start.sh", "scheduler"]  #command: ["sleep", "infinity"]  volumeMounts:  - mountPath: /usr/local/airflow/dags  name: vol-nfs  subPath: dags  - mountPath: /usr/local/airflow/logs  name: vol-nfs  subPath: logs  - mountPath: /etc/krb5.conf  name: vol-nfs  subPath: krb5.conf  - name: vol-nfs  mountPath: /opt/hadoop-conf  subPath: hadoop-conf  - name: vol-sec  mountPath: /etc/keytab  subPath: tuanndd.keytab  env:  - name: AIRFLOW\_HOME  value: /usr/local/airflow  - name: LOAD\_EX  value: "n"  - name: FERNET\_KEY  value: "46BKJoQYlPPOexq0OhDZnIlNepKFf87WFwLbfzqDDho="  - name: EXECUTOR  value: Celery  - name: REDIS\_HOST  value: redis  - name: REDIS\_PORT  value: "6379"  - name: REDIS\_PASSWORD  value: password  - name: POSTGRES\_HOST  value: postgres  - name: POSTGRES\_PORT  value: "5432"  - name: POSTGRES\_DB  value: airflow  - name: POSTGRES\_USER  value: airflow  - name: POSTGRES\_PASSWORD  value: airflow  - name: AIRFLOW\_\_KERBEROS\_\_CCACHE  value: /tmp/krb5cc\_0  - name: AIRFLOW\_\_KERBEROS\_\_PRINCIPAL  value: tuanndd@HDFS.ORG  - name: AIRFLOW\_\_KERBEROS\_\_KEYTAB  value: /etc/keytab  restartPolicy: Always  imagePullSecrets:  - name: local-registry  volumes:  - name: vol-sec  secret:  secretName: dp-sec  items:  - key: keytab  path: tuanndd.keytab  - name: vol-nfs  persistentVolumeClaim:  claimName: nfs |
| worker.yaml | apiVersion: apps/v1  kind: Deployment  metadata:  name: worker  labels:  app: worker  spec:  replicas: 2  selector:  matchLabels:  app: worker  template:  metadata:  labels:  app: worker  spec:  containers:  - name: worker  image: h1.spark.org:5000/airflow-docker:latest  imagePullPolicy: Always  resources:  requests:  cpu: 100m  memory: 1G  command: ["/start.sh", "worker"]  #command: ["sleep", "infinity"]  volumeMounts:  - mountPath: /usr/local/airflow/dags  name: vol-nfs  subPath: dags  - mountPath: /usr/local/airflow/logs  name: vol-nfs  subPath: logs  - mountPath: /etc/krb5.conf  name: vol-nfs  subPath: krb5.conf  - name: vol-nfs  mountPath: /opt/spark  subPath: spark-2.3.0-bin-hadoop2.7  - name: vol-nfs  mountPath: /opt/hadoop-conf  subPath: hadoop-conf  - name: vol-sec  mountPath: /etc/keytab  subPath: tuanndd.keytab  ports:  - containerPort: 8793  name: airflow  env:  - name: HADOOP\_HOME  value: /opt/hadoop  - name: HADOOP\_CONF\_DIR  value: /opt/hadoop-conf  - name: SPARK\_HOME  value: /opt/spark  - name: JAVA\_HOME  value: /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.212.b04-0.el7\_6.x86\_64  - name: KRB\_USERNAME  value: tuanndd@HDFS.ORG  - name: AIRFLOW\_HOME  value: /usr/local/airflow  - name: FERNET\_KEY  value: "46BKJoQYlPPOexq0OhDZnIlNepKFf87WFwLbfzqDDho="  - name: EXECUTOR  value: Celery  - name: REDIS\_HOST  value: redis  - name: REDIS\_PORT  value: "6379"  - name: REDIS\_PASSWORD  value: password  - name: POSTGRES\_HOST  value: postgres  - name: POSTGRES\_PORT  value: "5432"  - name: POSTGRES\_DB  value: airflow  - name: POSTGRES\_USER  value: airflow  - name: POSTGRES\_PASSWORD  value: airflow  - name: C\_FORCE\_ROOT  value: "true"  - name: AIRFLOW\_\_KERBEROS\_\_CCACHE  value: /tmp/krb5cc\_0  - name: AIRFLOW\_\_KERBEROS\_\_PRINCIPAL  value: tuanndd@HDFS.ORG  - name: AIRFLOW\_\_KERBEROS\_\_KEYTAB  value: /etc/keytab  restartPolicy: Always  imagePullSecrets:  - name: local-registry  volumes:  - name: vol-sec  secret:  secretName: dp-sec  items:  - key: keytab  path: tuanndd.keytab  - name: vol-nfs  persistentVolumeClaim:  claimName: nfs |

## Copy hadoop-conf đến persistent volume

kubectl cp hadoop-conf nfs-server-pod:/exports/

kubectl exec -it nfs-server-pod sh

# ls -la /exports/hadoop-conf

## Copy keytab đến secret

kubectl create secret generic dp-sec --from-file=keytab=/home/vagrant/kerberos/data/keytab

## Start airflow

kubectl create -f postgres.yaml

kubectl create -f redis.yaml

kubectl create -f webserver.yaml

kubectl create -f flower.yaml

kubectl create -f scheduler.yaml

kubectl create -f worker.yaml

## Test airflow

/usr/local/airflow/dags/tutorial.py

"""

Code that goes along with the Airflow tutorial located at:

https://github.com/airbnb/airflow/blob/master/airflow/example\_dags/tutorial.py

"""

from airflow import DAG

from airflow.operators.bash\_operator import BashOperator

from datetime import datetime, timedelta

default\_args = {

'owner': 'airflow',

'depends\_on\_past': False,

'start\_date': datetime(2015, 6, 1),

'email': ['airflow@example.com'],

'email\_on\_failure': False,

'email\_on\_retry': False,

'retries': 1,

'retry\_delay': timedelta(minutes=5),

# 'queue': 'bash\_queue',

# 'pool': 'backfill',

# 'priority\_weight': 10,

# 'end\_date': datetime(2016, 1, 1),

}

dag = DAG('tutorial', default\_args=default\_args)

# t1, t2 and t3 are examples of tasks created by instantiating operators

t1 = BashOperator(

task\_id='print\_date',

bash\_command='date && klist',

dag=dag)

t2 = BashOperator(

task\_id='sleep',

bash\_command='sleep 5',

retries=3,

dag=dag)

templated\_command = """

{% for i in range(5) %}

echo "{{ ds }}"

echo "{{ macros.ds\_add(ds, 7)}}"

echo "{{ params.my\_param }}"

{% endfor %}

"""

t3 = BashOperator(

task\_id='templated',

bash\_command=templated\_command,

params={'my\_param': 'Parameter I passed in'},

dag=dag)

t2.set\_upstream(t1)

t3.set\_upstream(t1)

## Stop airflow (optional)

kubectl delete -f worker.yaml

kubectl delete -f scheduler.yaml

kubectl delete -f flower.yaml

kubectl delete -f webserver.yaml

kubectl create -f redis.yaml

kubectl create -f postgres.yaml

# Quản lý cấp phát resource trên k8s

## Quản lý memory & cpu của pod

Đơn vị đo lường:

cpu: milicore (vd: 2000m = 2 full core cpu, 250m = 1/4 core)

mem: byte (vd: 32Mi)

Memory không thể không hồi sau khi đã cấp

CPU có tính "compressible" (thu hồi lại được)

Dùng request & limit để giới hạn resource của 1 pod cụ thể

Dùng ResourceQuotas để giới hạn tổng resource của 1 namespace (vd: production, development)

Dùng LimitRanger để giới hạn resource cho mỗi pod trong 1 namespace

<https://cloud.google.com/blog/products/gcp/kubernetes-best-practices-resource-requests-and-limits>

Demo: <https://www.alibabacloud.com/blog/how-to-set-constraints-on-kubernetes-resources_594313>

## Autoscale

HPA: horizontal scale (thêm/remove pod)

VPA: vertical scale (change cpu, mem của pod, yêu cầu: restart lại pod)

CA: cluster scale (thêm node khi có nhiều pod ở trạng thái pending)

<https://medium.com/magalix/kubernetes-autoscaling-101-cluster-autoscaler-horizontal-pod-autoscaler-and-vertical-pod-2a441d9ad231>

# Cronjob

<https://kubernetes.io/docs/tasks/job/automated-tasks-with-cron-jobs/>

<https://hub.docker.com/r/pietervogelaar/kubernetes-job-monitor/>

# Monitor

## kubectl top

git clone https://github.com/kubernetes-incubator/metrics-server.git

cd metrics-server

vi deploy/1.8+/metrics-server-deployment.yaml

command:

- /metrics-server

- --kubelet-insecure-tls

- --kubelet-preferred-address-types=InternalIP

kubectl apply -f deploy/1.8+/

kubectl top node

kubectl top pod

<https://medium.com/@cagri.ersen/kubernetes-metrics-server-installation-d93380de008>

## Check readiness & liveness probe

<https://cloud.google.com/blog/products/gcp/kubernetes-best-practices-setting-up-health-checks-with-readiness-and-liveness-probes>

## Dùng Kubernetes dashboard

<https://xuri.me/2019/01/23/deploy-the-kubernetes-web-ui-dashboard.html>

wget <https://raw.githubusercontent.com/kubernetes/dashboard/v1.10.1/src/deploy/recommended/kubernetes-dashboard.yaml>

vi kubernetes-dashboard.yaml

kind: Service

apiVersion: v1

metadata:

labels:

k8s-app: kubernetes-dashboard

name: kubernetes-dashboard

namespace: kube-system

spec:

ports:

- port: 443

targetPort: 8443

selector:

k8s-app: kubernetes-dashboard

type: **NodePort**

# Tạo user admin-user, gán vào group cluster-admin

vi dashboard-adminuser.yaml

apiVersion: v1

kind: ServiceAccount

metadata:

name: admin-user

namespace: kube-system

---

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

name: admin-user

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: cluster-admin

subjects:

- kind: ServiceAccount

name: admin-user

namespace: kube-system

kubectl create -f kubernetes-dashboard.yaml

kubectl create -f dashboard-adminuser.yaml

# Lấy token của user admin-user

kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep admin-user | awk '{print $1}')

kubectl get pod -o wide

kubectl get svc

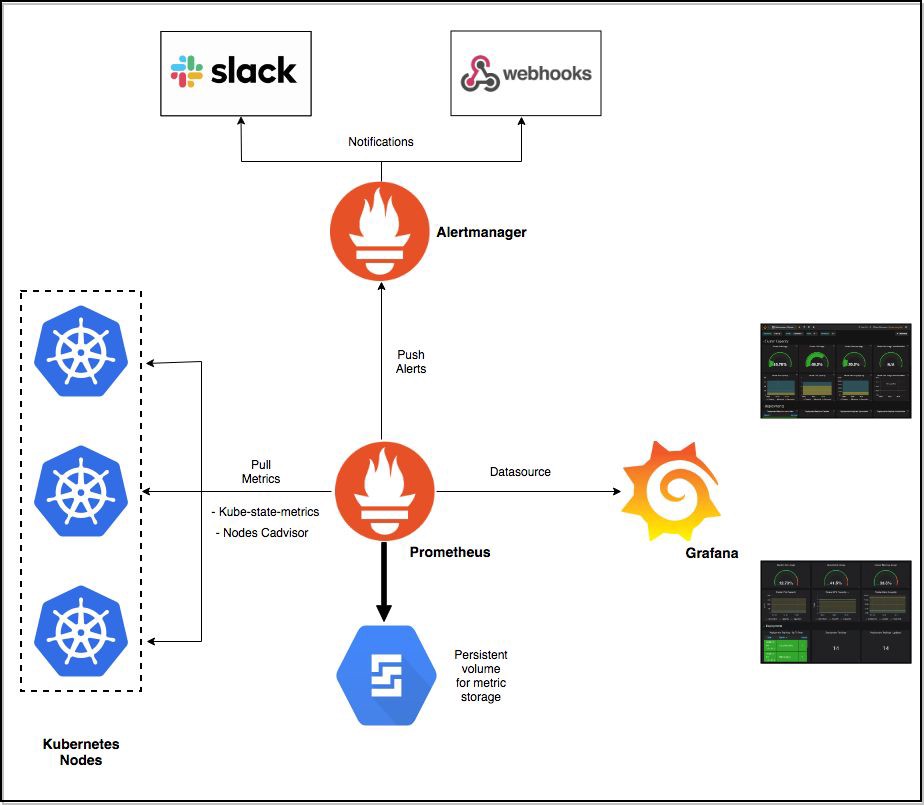
<https://h1.spark.org:30743>

Token: …

<https://xuri.me/2019/01/23/deploy-the-kubernetes-web-ui-dashboard.html>

## Monitor performance của k8s và alert bằng Prometheus & Grafana

### Mô hình:



### Deploy:

git clone <https://github.com/Thakurvaibhav/k8s.git>

cd k8s/monitoring

kubectl create -f kube-state-metrics/

kubectl create -f node-exporter/

vi alertmanager/alertmanager-service.yaml

spec:

selector:

app: alertmanager

type: NodePort

// config smtp để gởi mail alert

vi alertmanager/00-alertmanager-configmap.yaml

kubectl apply -f alertmanager/

// config alert rules

vi prometheus/02-prometheus-rules.yaml

// config storage cho prometheus tsdb

vi prometheus/03-prometheus-storage.yaml

vi prometheus/prometheus-service.yaml

type: NodePort

kubectl apply -f prometheus/

vi grafana/grafana-service.yaml

type: NodePort

kubectl apply -f grafana/

// vào web grafana, add data source và import các dashboard

kubectl -n monitoring get svc

<http://h3.spark.org:32495>

dashboards/

├── Kubernetes\ App\ Metrics.json

├── Kubernetes\ cluster\ monitoring\ (via\ Prometheus).json

├── Kubernetes\ Deployment\ metrics.json

├── Kubernetes\ Nodes\ -\ 01\ (Node\ Exporter).json

├── Kubernetes\ Nodes\ -\ 02(Node\ Exporter).json

└── Kubernetes\ Pod\ Metrics.json

<https://medium.com/faun/production-grade-kubernetes-monitoring-using-prometheus-78144b835b60>

Chi tiết hơn: <https://sysdig.com/blog/kubernetes-monitoring-prometheus/>

## PoC prometheus, xem dashboard trên grafana, alert bằng alertmanager, gmail

### Start prometheus (port 9090)

./prometheus --config.file=prometheus.yml

vi prometheus.yml

# my global config

global:

scrape\_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.

evaluation\_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.

# scrape\_timeout is set to the global default (10s).

external\_labels:

monitor: 'test-system'

# Alertmanager configuration

alerting:

alertmanagers:

- static\_configs:

- targets:

- localhost:9093

# Load rules once and periodically evaluate them according to the global 'evaluation\_interval'.

rule\_files:

# - "first\_rules.yml"

# - "second\_rules.yml"

- 'prometheus.rules.yml'

# A scrape configuration containing exactly one endpoint to scrape:

# Here it's Prometheus itself.

scrape\_configs:

# The job name is added as a label `job=<job\_name>` to any timeseries scraped from this config.

- job\_name: 'prometheus'

# metrics\_path defaults to '/metrics'

# scheme defaults to 'http'.

#

scrape\_interval: 5s

static\_configs:

- targets: ['localhost:9090']

- job\_name: 'node\_exporter'

scrape\_interval: 5s

static\_configs:

- targets: ['localhost:9100', 'h2.spark.org:9100']

vi prometheus.rules.yml

groups:

- name: example

rules:

# - record: job\_service:rpc\_durations\_seconds\_count:avg\_rate5m

# expr: avg(rate(rpc\_durations\_seconds\_count[5m])) by (job, service)

- alert: InstanceDownLabels

expr: up == 0

for: 5m

labels:

severity: critical

annotations:

summary: "Instance {{ $labels.instance }} down"

description: "{{ $labels.instance }} of job {{ $labels.job }} has been down for more than 5 minutes."

Reload config: (optional)

reload : kill -HUP <pid>

curl -X POST http://localhost:9090/-/reload

### Start các node exporter (port 9100)

./ node\_exporter

### Start grafana (port 3000)

./ bin/grafana-server

### Start alertmanager (port 9093)

./alertmanager --config.file=alertmanager.yml

vi alertmanager.yml

global:

resolve\_timeout: 5m

route:

group\_by: ['alertname']

group\_wait: 10s

group\_interval: 10s

repeat\_interval: 1h

#receiver: 'web.hook'

receiver: email-me

receivers:

- name: email-me

email\_configs:

- to: XXXXX@gmail.com

from: XXXXX@gmail.com

smarthost: smtp.gmail.com:587

auth\_username: XXXXX@gmail.com

auth\_identity: XXXXX@gmail.com

auth\_password: \*\*\*\*\*

- name: 'web.hook'

webhook\_configs:

- url: 'http://127.0.0.1:5001/'

inhibit\_rules:

- source\_match:

severity: 'critical'

target\_match:

severity: 'warning'

equal: ['alertname', 'dev', 'instance']

Reference:

tổng quan: [https://github.com/otomato-gh/otom8-rprometheus/blob/master/Monitoring%20With%20Prometheus.pdf](https://github.com/otomato-gh/otom8-prometheus/blob/master/Monitoring%20With%20Prometheus.pdf)

alert rules có sẵn: <https://awesome-prometheus-alerts.grep.to/>

<https://www.replex.io/blog/kubernetes-in-production-the-ultimate-guide-to-monitoring-resource-metrics>

dashboard có sẵn: <https://blog.lwolf.org/post/going-open-source-in-monitoring-part-iii-10-most-useful-grafana-dashboards-to-monitor-kubernetes-and-services/>

secure prometheus bằng nginx reverse proxy: <https://www.robustperception.io/adding-basic-auth-to-prometheus-with-nginx>

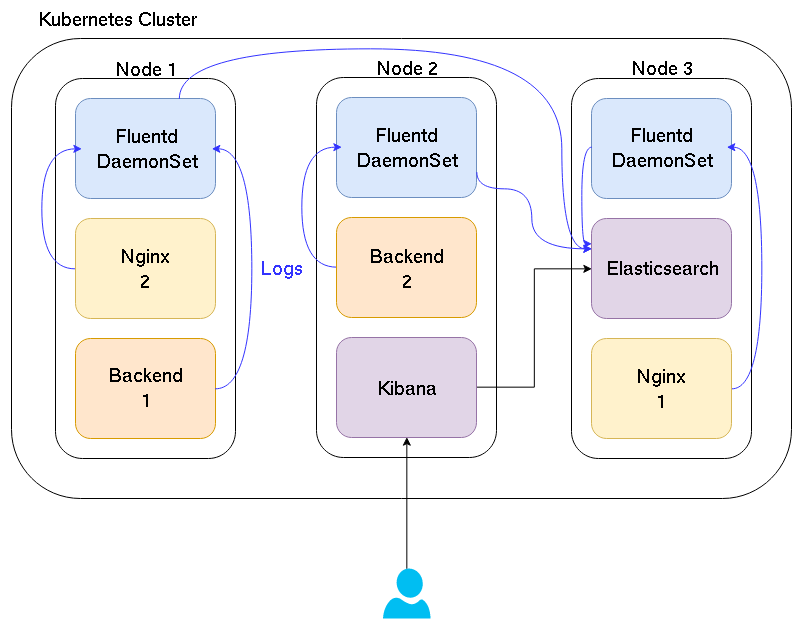
video: <https://www.youtube.com/watch?v=4WWW2ZLEg74>

<https://www.youtube.com/watch?v=rBunRQKIEdc>

<https://www.robustperception.io/sending-email-with-the-alertmanager-via-gmail>

## Centralize POD logs, k8s events bằng EFK stack

### Mô hình



* fluent-bit daemonset monitor /var/log/containers/\*.log push log vào Elasticsearch
* Kibana connect đến Elasticsearch để show log cho user

### Deploy:

// Cấu hình dockerd lưu log ra json file:

sudo vi /etc/docker/daemon.json

"log-driver": "json-file"

sudo systemctl restart docker.service

// Tạo local storage, persistent volume (chỉ để test, optional)

<https://vocon-it.com/2018/12/20/kubernetes-local-persistent-volumes/>

// Tạo ElasticSearch cluster để nhận và lưu log: gồm 3 master, 2 ingest, 2 data, dùng **emptyDir** volume (chỉ để test)

|  |  |
| --- | --- |
| ns.yaml | apiVersion: v1  kind: Namespace  metadata:  name: logging |
| es-discovery-svc.yaml | apiVersion: v1  kind: Service  metadata:  name: elasticsearch-discovery  namespace: logging  labels:  component: elasticsearch  role: master  spec:  selector:  component: elasticsearch  role: master  ports:  - name: transport  port: 9300  protocol: TCP  clusterIP: None |
| es-svc.yaml | apiVersion: v1  kind: Service  metadata:  name: elasticsearch  namespace: logging  labels:  component: elasticsearch  role: data  spec:  selector:  component: elasticsearch  role: data  ports:  - name: http  port: 9200  #type: LoadBalancer |
| es-master.yaml | apiVersion: apps/v1beta1  kind: Deployment  metadata:  name: es-master  namespace: logging  labels:  component: elasticsearch  role: master  spec:  replicas: 3  template:  metadata:  labels:  component: elasticsearch  role: master  spec:  initContainers:  - name: init-sysctl  image: busybox:1.27.2  command:  - sysctl  - -w  - vm.max\_map\_count=262144  securityContext:  privileged: true  containers:  - name: es-master  image: quay.io/pires/docker-elasticsearch-kubernetes:6.3.2  env:  - name: NAMESPACE  valueFrom:  fieldRef:  fieldPath: metadata.namespace  - name: NODE\_NAME  valueFrom:  fieldRef:  fieldPath: metadata.name  - name: CLUSTER\_NAME  value: myesdb  - name: NUMBER\_OF\_MASTERS  value: "2"  - name: NODE\_MASTER  value: "true"  - name: NODE\_INGEST  value: "false"  - name: NODE\_DATA  value: "false"  - name: HTTP\_ENABLE  value: "false"  - name: ES\_JAVA\_OPTS  value: -Xms256m -Xmx256m  - name: PROCESSORS  valueFrom:  resourceFieldRef:  resource: limits.cpu  resources:  requests:  cpu: 0.25  limits:  cpu: 1  ports:  - containerPort: 9300  name: transport  livenessProbe:  tcpSocket:  port: transport  initialDelaySeconds: 20  periodSeconds: 10  volumeMounts:  - name: storage  mountPath: /data  volumes:  - emptyDir:  medium: ""  name: "storage" |
| es-ingest-svc.yaml | apiVersion: v1  kind: Service  metadata:  name: elasticsearch-ingest  namespace: logging  labels:  component: elasticsearch  role: ingest  spec:  selector:  component: elasticsearch  role: ingest  ports:  - name: http  port: 9200  #type: LoadBalancer |
| es-ingest.yaml | apiVersion: apps/v1beta1  kind: Deployment  metadata:  name: es-ingest  namespace: logging  labels:  component: elasticsearch  role: ingest  spec:  replicas: 2  template:  metadata:  labels:  component: elasticsearch  role: ingest  spec:  initContainers:  - name: init-sysctl  image: busybox:1.27.2  command:  - sysctl  - -w  - vm.max\_map\_count=262144  securityContext:  privileged: true  containers:  - name: es-ingest  image: quay.io/pires/docker-elasticsearch-kubernetes:6.3.2  env:  - name: NAMESPACE  valueFrom:  fieldRef:  fieldPath: metadata.namespace  - name: NODE\_NAME  valueFrom:  fieldRef:  fieldPath: metadata.name  - name: CLUSTER\_NAME  value: myesdb  - name: NODE\_MASTER  value: "false"  - name: NODE\_DATA  value: "false"  - name: HTTP\_ENABLE  value: "true"  - name: ES\_JAVA\_OPTS  value: -Xms256m -Xmx256m  - name: NETWORK\_HOST  value: \_site\_,\_lo\_  - name: PROCESSORS  valueFrom:  resourceFieldRef:  resource: limits.cpu  resources:  requests:  cpu: 0.25  limits:  cpu: 1  ports:  - containerPort: 9200  name: http  - containerPort: 9300  name: transport  livenessProbe:  tcpSocket:  port: transport  initialDelaySeconds: 20  periodSeconds: 10  readinessProbe:  httpGet:  path: /\_cluster/health  port: http  initialDelaySeconds: 20  timeoutSeconds: 5  volumeMounts:  - name: storage  mountPath: /data  volumes:  - emptyDir:  medium: ""  name: storage |
| es-data.yaml | apiVersion: apps/v1beta1  kind: Deployment  metadata:  name: es-data  namespace: logging  labels:  component: elasticsearch  role: data  spec:  replicas: 2  template:  metadata:  labels:  component: elasticsearch  role: data  spec:  initContainers:  - name: init-sysctl  image: busybox:1.27.2  command:  - sysctl  - -w  - vm.max\_map\_count=262144  securityContext:  privileged: true  containers:  - name: es-data  image: quay.io/pires/docker-elasticsearch-kubernetes:6.3.2  env:  - name: NAMESPACE  valueFrom:  fieldRef:  fieldPath: metadata.namespace  - name: NODE\_NAME  valueFrom:  fieldRef:  fieldPath: metadata.name  - name: CLUSTER\_NAME  value: myesdb  - name: NODE\_MASTER  value: "false"  - name: NODE\_INGEST  value: "false"  - name: HTTP\_ENABLE  value: "true"  - name: ES\_JAVA\_OPTS  value: -Xms256m -Xmx256m  - name: PROCESSORS  valueFrom:  resourceFieldRef:  resource: limits.cpu  resources:  requests:  cpu: 0.25  limits:  cpu: 1  ports:  - containerPort: 9200  name: http  - containerPort: 9300  name: transport  livenessProbe:  tcpSocket:  port: transport  initialDelaySeconds: 20  periodSeconds: 10  readinessProbe:  httpGet:  path: /\_cluster/health  port: http  initialDelaySeconds: 20  timeoutSeconds: 5  volumeMounts:  - name: storage  mountPath: /data  volumes:  - emptyDir:  medium: ""  name: storage |

// Triển khai kibana

|  |  |
| --- | --- |
| kibana-cm.yaml | apiVersion: v1  kind: ConfigMap  metadata:  name: kibana-config  namespace: logging  data:  kibana.yml: |  ---  server.name: kibana  server.host: "0"  elasticsearch.url: <http://elasticsearch:9200> |
| kibana-svc.yaml | apiVersion: v1  kind: Service  metadata:  name: kibana  namespace: logging  labels:  component: kibana  spec:  selector:  component: kibana  type: NodePort  ports:  - name: http  port: 5601 |
| kibana.yaml | apiVersion: apps/v1beta1  kind: Deployment  metadata:  name: kibana  namespace: logging  labels:  component: kibana  spec:  replicas: 1  selector:  matchLabels:  component: kibana  template:  metadata:  labels:  component: kibana  spec:  containers:  - name: kibana  image: docker.elastic.co/kibana/kibana-oss:6.3.2  env:  - name: CLUSTER\_NAME  value: myesdb  resources:  limits:  cpu: 1000m  requests:  cpu: 100m  ports:  - containerPort: 5601  name: http  volumeMounts:  - name: config  mountPath: /usr/share/kibana/config  readOnly: true  volumes:  - name: config  configMap:  name: kibana-config |

// Triển khai fluent-bit daemonset

|  |  |
| --- | --- |
| fluent-bit-service-account.yaml | apiVersion: v1  kind: ServiceAccount  metadata:  name: fluent-bit  namespace: logging |
| fluent-bit-service-account.yaml | apiVersion: v1  kind: ServiceAccount  metadata:  name: fluent-bit  namespace: logging |
| fluent-bit-role-binding.yaml | apiVersion: rbac.authorization.k8s.io/v1beta1  kind: ClusterRoleBinding  metadata:  name: fluent-bit-read  roleRef:  apiGroup: rbac.authorization.k8s.io  kind: ClusterRole  name: fluent-bit-read  subjects:  - kind: ServiceAccount  name: fluent-bit  namespace: logging |
| fluent-bit-configmap.yaml | apiVersion: v1  kind: ConfigMap  metadata:  name: fluent-bit-config  namespace: logging  labels:  k8s-app: fluent-bit  data:  # Configuration files: server, input, filters and output  # ======================================================  fluent-bit.conf: |  [SERVICE]  Flush 1  Log\_Level info  Daemon off  Parsers\_File parsers.conf  HTTP\_Server On  HTTP\_Listen 0.0.0.0  HTTP\_Port 2020  @INCLUDE input-kubernetes.conf  @INCLUDE filter-kubernetes.conf  @INCLUDE output-elasticsearch.conf  input-kubernetes.conf: |  [INPUT]  Name tail  Tag kube.\*  Path /var/log/containers/\*.log  Parser docker  DB /var/log/flb\_kube.db  Mem\_Buf\_Limit 5MB  Skip\_Long\_Lines On  Refresh\_Interval 10  filter-kubernetes.conf: |  [FILTER]  Name kubernetes  Match kube.\*  Kube\_URL https://kubernetes.default.svc:443  Kube\_CA\_File /var/run/secrets/kubernetes.io/serviceaccount/ca.crt  Kube\_Token\_File /var/run/secrets/kubernetes.io/serviceaccount/token  Merge\_Log On  K8S-Logging.Parser On  K8S-Logging.Exclude Off  output-elasticsearch.conf: |  [OUTPUT]  Name es  Match \*  Host ${FLUENT\_ELASTICSEARCH\_HOST}  Port ${FLUENT\_ELASTICSEARCH\_PORT}  Logstash\_Format On  Replace\_Dots On  Retry\_Limit False  parsers.conf: |  [PARSER]  Name apache  Format regex  Regex ^(?<host>[^ ]\*) [^ ]\* (?<user>[^ ]\*) \[(?<time>[^\]]\*)\] "(?<method>\S+)(?: +(?<path>[^\"]\*?)(?: +\S\*)?)?" (?<code>[^ ]\*) (?<size>[^ ]\*)(?: "(?<referer>[^\"]\*)" "(?<agent>[^\"]\*)")?$  Time\_Key time  Time\_Format %d/%b/%Y:%H:%M:%S %z  [PARSER]  Name apache2  Format regex  Regex ^(?<host>[^ ]\*) [^ ]\* (?<user>[^ ]\*) \[(?<time>[^\]]\*)\] "(?<method>\S+)(?: +(?<path>[^ ]\*) +\S\*)?" (?<code>[^ ]\*) (?<size>[^ ]\*)(?: "(?<referer>[^\"]\*)" "(?<agent>[^\"]\*)")?$  Time\_Key time  Time\_Format %d/%b/%Y:%H:%M:%S %z  [PARSER]  Name apache\_error  Format regex  Regex ^\[[^ ]\* (?<time>[^\]]\*)\] \[(?<level>[^\]]\*)\](?: \[pid (?<pid>[^\]]\*)\])?( \[client (?<client>[^\]]\*)\])? (?<message>.\*)$  [PARSER]  Name nginx  Format regex  Regex ^(?<remote>[^ ]\*) (?<host>[^ ]\*) (?<user>[^ ]\*) \[(?<time>[^\]]\*)\] "(?<method>\S+)(?: +(?<path>[^\"]\*?)(?: +\S\*)?)?" (?<code>[^ ]\*) (?<size>[^ ]\*)(?: "(?<referer>[^\"]\*)" "(?<agent>[^\"]\*)")?$  Time\_Key time  Time\_Format %d/%b/%Y:%H:%M:%S %z  [PARSER]  Name json  Format json  Time\_Key time  Time\_Format %d/%b/%Y:%H:%M:%S %z  [PARSER]  Name docker  Format json  Time\_Key time  Time\_Format %Y-%m-%dT%H:%M:%S.%L  Time\_Keep On  # Command | Decoder | Field | Optional Action  # =============|==================|=================  Decode\_Field\_As escaped log  [PARSER]  Name syslog  Format regex  Regex ^\<(?<pri>[0-9]+)\>(?<time>[^ ]\* {1,2}[^ ]\* [^ ]\*) (?<host>[^ ]\*) (?<ident>[a-zA-Z0-9\_\/\.\-]\*)(?:\[(?<pid>[0-9]+)\])?(?:[^\:]\*\:)? \*(?<message>.\*)$  Time\_Key time  Time\_Format %b %d %H:%M:%S |
| fluent-bit-ds.yaml | apiVersion: apps/v1  kind: DaemonSet  metadata:  name: fluent-bit  namespace: logging  labels:  component: fluent-bit-logging  version: v1  kubernetes.io/cluster-service: "true"  spec:  selector:  matchLabels:  component: fluent-bit-logging  template:  metadata:  labels:  component: fluent-bit-logging  version: v1  kubernetes.io/cluster-service: "true"  spec:  containers:  - name: fluent-bit  image: fluent/fluent-bit:0.12.17  env:  - name: FLUENT\_ELASTICSEARCH\_HOST  value: "elasticsearch" # the name of the previous es-svc.yml  - name: FLUENT\_ELASTICSEARCH\_PORT  value: "9200" # the port of the previous es-svc.yml  volumeMounts:  - name: varlog  mountPath: /var/log  - name: varlibdockercontainers  mountPath: /var/lib/docker/containers  readOnly: true  - name: fluent-bit-config  mountPath: /fluent-bit/etc/  terminationGracePeriodSeconds: 10  volumes:  - name: varlog  hostPath:  path: /var/log  - name: varlibdockercontainers  hostPath:  path: /var/lib/docker/containers  - name: fluent-bit-config  configMap:  name: fluent-bit-config # name of the previously created ConfigMap  serviceAccountName: fluent-bit  tolerations:  - key: node-role.kubernetes.io/master  operator: Exists  effect: NoSchedule |

// k8s event-exporter

|  |  |
| --- | --- |
| k8s-events-printer.yaml | apiVersion: v1  kind: ServiceAccount  metadata:  name: event-exporter-sa  ---  apiVersion: rbac.authorization.k8s.io/v1beta1  kind: ClusterRoleBinding  metadata:  name: event-exporter-rb-example  roleRef:  apiGroup: rbac.authorization.k8s.io  kind: ClusterRole  name: view  subjects:  - kind: ServiceAccount  name: event-exporter-sa  namespace: default  ---  apiVersion: extensions/v1beta1  kind: Deployment  metadata:  name: kubernetes-events-printer  labels:  component: fluentd  role: kubernetes-events-printer  spec:  strategy:  type: Recreate  replicas: 1  template:  metadata:  labels:  component: fluentd  role: kubernetes-events-printer  spec:  serviceAccountName: event-exporter-sa  containers:  - name: kubernetes-events-printer  image: kayrus/kubernetes-events-printer:latest  imagePullPolicy: Always |

# Vào kibana web, add index: logstash-\*, ts: @timestamp

Tham khảo: [**https://blog.ptrk.io/how-to-deploy-an-efk-stack-to-kubernetes/**](https://blog.ptrk.io/how-to-deploy-an-efk-stack-to-kubernetes/) **\***

<https://mherman.org/blog/logging-in-kubernetes-with-elasticsearch-Kibana-fluentd/>

# Quản lý user trên ELK

1) Change password của "elastic" user

vi config/elasticsearch.yaml

xpack.security.enabled: true

bin/elasticsearch-keystore create

echo "\*\*\*\*\*\*" | bin/elasticsearch-keystore add -x 'bootstrap.password'

(docker: env "ELASTIC\_PASSWORD": \*\*\*\*\*\*)

bin/elasticsearch

2) Update config cho kibana

vi config/kibana.yml

server.host: "0.0.0.0"

bin/kibana-keystore create

echo "elastic" | bin/kibana-keystore add elasticsearch.username

echo "\*\*\*\*\*\*" | bin/kibana-keystore add elasticsearch.password

bin/kibana

3) Login vào kibana web bằng user elastic, password: \*\*\*\*\*\*, section Management / Security / Users, Roles

4) Config out\_elasticsearch fluentd

user fluent

password mysecret

# Security

<https://dev.to/petermbenjamin/kubernetes-security-best-practices-hlk>

## Benchmark security

<https://github.com/aquasecurity/kube-bench>

// chạy trên master node  
docker run --pid=host -v /etc:/etc:ro -v /var:/var:ro -t aquasec/kube-bench:latest master --version 1.13

// chạy trên các worker nodes

docker run --pid=host -v /etc:/etc:ro -v /var:/var:ro -t aquasec/kube-bench:latest node --version 1.13

## Update authentication

### Disable anonymous login

sudo vi /etc/kubernetes/manifests/kube-apiserver.yaml

- --anonymous-auth=false

### Authen user bằng X509 cert

Ví dụ: tạo user “kube-ops” cho cluster “kubernetes” (api server: <https://192.168.2.101:6443)>

# tạo client cert và sign cert

openssl req -newkey rsa:2048 -nodes -keyout kube-ops.key -out kube-ops.csr -subj "/CN=kube-ops"

sudo openssl x509 -req -in kube-ops.csr -CA /etc/kubernetes/pki/ca.crt -CAkey /etc/kubernetes/pki/ca.key -CAcreateserial -out kube-ops.crt -days 1000

# tạo config file

vi config

apiVersion: v1

clusters:

- cluster:

certificate-authority-data: <CA-DATA>

server: https://192.168.2.101:6443

name: <CLUSTER-NAME>

contexts:

- context:

cluster: <CLUSTER-NAME>

user: <USER> # e.g. kube-ops

name: <USER>@<CLUSTER-NAME>

kind: Config

users:

- name: <USER> # e.g. kube-ops

user:

client-certificate-data: <CLIENT-CRT-DATA>

client-key-data: <CLIENT-KEY-DATA>

# Generate the <CA-DATA>

sudo cat /etc/kubernetes/pki/ca.crt | base64 | tr -d '\n'

# Generate the <CLIENT-CRT-DATA>

cat kube-ops.crt | base64 | tr -d '\n'

# Generate the <CLIENT-KEY-DATA>

cat kube-ops.key | base64 | tr -d '\n'

# Gán role cluster-admin cho user kube-ops

kubectl create clusterrolebinding kube-ops --clusterrole=cluster-admin --user=kube-ops

# Test login bằng user kube-ops

kubectl --kubeconfig=./config --context=kube-ops@kubernetes get clusterrolebindings kube-ops

<https://codefarm.me/2019/02/01/access-kubernetes-api-with-client-certificates/>

### Disable auto-mout “default” service account khi tạo pod

kubectl patch serviceaccount default -p "automountServiceAccountToken: false"

attack PoC: <https://medium.com/@pjbgf/aks-security-challenge-2-reaching-out-to-fellow-pods-eb8a99968a73>

## Setup Pod Policy

### Enable PodSecurityPolicy

sudo vi /etc/kubernetes/manifests/kube-apiserver.yaml

- --enable-admission-plugins=NodeRestriction,PodSecurityPolicy

### Nguyên tắc set policy

* Tất cả pod default dùng policy “psp-restrict” (container chạy bằng non-root user, không access đến [process, network, memory] của host, optional: không thay đổi được file system: readOnlyRootFilesystem: true)
* Một số pod cần chạy ở quyền root sẽ set serviceAccountName: privileged-sa để dùng policy “psp-privileged”

Demo: <https://banzaicloud.com/blog/pod-security-policy/>

(giải thích các option trong PodSecurityPolicy: <https://www.slideshare.net/Paris_Container_Day/security-in-a-containerized-world-jessie-frazelle>

)

vi psp-restrict.yaml

apiVersion: policy/v1beta1

kind: PodSecurityPolicy

metadata:

name: psp.restricted

annotations:

seccomp.security.alpha.kubernetes.io/defaultProfileName: 'docker/default'

seccomp.security.alpha.kubernetes.io/allowedProfileNames: 'docker/default'

spec:

privileged: false

# Required to prevent escalations to root.

allowPrivilegeEscalation: false

# This is redundant with non-root + disallow privilege escalation,

# but we can provide it for defense in depth.

requiredDropCapabilities:

- ALL

# Allow core volume types.

volumes:

- configMap

- emptyDir

- secret

- persistentVolumeClaim

hostNetwork: false

hostIPC: false

hostPID: false

runAsUser:

# Require the container to run without root privileges.

rule: 'MustRunAsNonRoot'

seLinux:

# This policy assumes the nodes are using AppArmor rather than SELinux.

rule: 'RunAsAny'

supplementalGroups:

rule: 'MustRunAs'

ranges:

# Forbid adding the root group.

- min: 1

max: 65535

fsGroup:

rule: 'MustRunAs'

ranges:

# Forbid adding the root group.

- min: 1

max: 65535

readOnlyRootFilesystem: false

vi psp-privileged.yaml

apiVersion: policy/v1beta1

kind: PodSecurityPolicy

metadata:

creationTimestamp: null

name: psp.privileged

spec:

readOnlyRootFilesystem: false

privileged: true

allowPrivilegeEscalation: true

runAsUser:

rule: 'RunAsAny'

supplementalGroups:

rule: 'RunAsAny'

fsGroup:

rule: 'RunAsAny'

seLinux:

rule: 'RunAsAny'

volumes:

- configMap

- emptyDir

- secret

Tạo Pod

## Setup Network Policy

### Nguyên tắc set policy

Option1: High security : block tất cả các kết nối vào ra

Option 2: Low security: chỉ cho phép traffic giữa các pod trong cùng 1 namespace

### Demo

namespace=secondary chạy 2 pod web port 80 và db port 80. Bên ngoài được phép access vào pod web. Các pod chỉ được ra ngoài đến kube-dns và 8.8.8.8

high\_security

├── allow-in-web-app-from-any.yaml

├── allow-out-to-k8s-dns.yml

├── allow-out-to-others.yaml

├── default-deny-in.yaml

├── default-deny-out.yaml

├── *internal-allow-in.yaml*

└── *internal-allow-out.yaml*

low\_security

├── allow-in-web-app-from-any.yaml

├── allow-out-to-k8s-dns.yml

├── allow-out-to-others.yaml

├── default-deny-in.yaml

├── default-deny-out.yaml

└── *internal-allow.yaml*

|  |  |
| --- | --- |
| File | Content |
| default-deny-in.yaml | kind: NetworkPolicy  apiVersion: networking.k8s.io/v1  metadata:  namespace: secondary  name: default-deny-in  spec:  podSelector:  matchLabels:  ingress: [] |
| default-deny-out.yaml | apiVersion: networking.k8s.io/v1  kind: NetworkPolicy  metadata:  name: default-deny-out  namespace: secondary  spec:  podSelector:  matchLabels:  egress: [] |
| internal-allow-in.yaml | kind: NetworkPolicy  apiVersion: networking.k8s.io/v1  metadata:  namespace: secondary  name: internal-allow-in  spec:  podSelector:  matchLabels:  app: web  #matchExpressions:  # - {key: app, operator: NotIn, values: [db]}  ingress:  - from:  - podSelector: {} |
| internal-allow-out.yaml | apiVersion: networking.k8s.io/v1  kind: NetworkPolicy  metadata:  name: internal-allow-out  namespace: secondary  spec:  podSelector:  matchLabels:  egress:  - to:  - podSelector:  matchLabels:  app: web |
| allow-out-to-k8s-dns.yml | apiVersion: networking.k8s.io/v1  kind: NetworkPolicy  metadata:  name: allow-out-to-k8s-dns  namespace: secondary  spec:  podSelector:  matchLabels:  egress:  - ports:  - port: 53  protocol: TCP  - port: 53  protocol: UDP  - to:  - namespaceSelector:  matchLabels:  namespace: kube-system  podSelector:  matchLabels:  k8s-app: kube-dns |
| allow-out-to-others.yaml | apiVersion: networking.k8s.io/v1  kind: NetworkPolicy  metadata:  name: allow-out-to-others  namespace: secondary  spec:  podSelector:  matchLabels:  egress:  - to:  - ipBlock:  cidr: 8.8.8.8/32 |
| allow-in-web-app-from-any.yaml | kind: NetworkPolicy  apiVersion: networking.k8s.io/v1  metadata:  namespace: secondary  name: allow-in-web-app-from-any  spec:  podSelector:  matchLabels:  app: web  #matchExpressions:  # - {key: app, operator: NotIn, values: [db]}  ingress:  - ports:  - port: 80  protocol: TCP  from: [] |
| internal-allow.yaml | kind: NetworkPolicy  apiVersion: networking.k8s.io/v1  metadata:  namespace: secondary  name: internal-allow  spec:  podSelector:  matchLabels:  #matchExpressions:  # - {key: app, operator: NotIn, values: [db]}  policyTypes:  - Ingress  - Egress  ingress:  - from:  - podSelector: {}  egress:  - to:  - podSelector: {} |

# tạo pod web (pod db làm tương tự)

apiVersion: v1

kind: Namespace

metadata:

name: secondary

---

apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2

kind: Deployment

metadata:

name: web

namespace: secondary

spec:

selector:

matchLabels:

app: web

replicas: 1 # tells deployment to run 1 pods matching the template

template: # create pods using pod definition in this template

metadata:

labels:

app: web

spec:

containers:

- name: nginx

image: nginx

ports:

- containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

name: web

namespace: secondary

labels:

app: nginx

spec:

ports:

- name: http

port: 80

protocol: TCP

targetPort: 80

selector:

app: web

type: NodePort

# apply network policy

kubectl apply -f ./high\_security

(kubectl apply -f ./low\_security)

# test network policy

kubectl run test-$RANDOM --namespace=secondary --rm -i -t --image=alpine -- sh

wget -qO- --timeout=2 <http://web>

wget -qO- --timeout=2 <http://db>

ping 8.8.8.8

kubectl run test-$RANDOM --namespace=default --rm -i -t --image=alpine -- sh

wget -qO- --timeout=2 <http://web>

wget -qO- --timeout=2 <http://db>

# delete network policy

kubectl delete -f ./high\_security

(kubectl delete -f ./low\_security)

Tham khảo:

<https://github.com/ahmetb/kubernetes-network-policy-recipes>

<https://medium.com/@reuvenharrison/an-introduction-to-kubernetes-network-policies-for-security-people-ba92dd4c809d>

<https://schoolofdevops.github.io/ultimate-kubernetes-bootcamp/network_policies/>

## Enable audit logging cho kube-apiserver

<https://www.noqcks.io/notes/2018/03/31/kubernetes-audit-logging-tutorial/>

## Phát hiện các security event bất thường bằng Falco

### Giải thích:

falco dùng kernel module log các system call event, và alert bằng rule based

## Setup:

// TODO: cần build lại docker hoặc dùng initContainer của k8s để set đúng timezone cho falco

.

├── falco-account.yaml

├── falco-config

│   ├── application\_rules.yaml

│   ├── falco\_rules.local.yaml

│   ├── falco\_rules.yaml

│   ├── falco.yaml

│   └── k8s\_audit\_rules.yaml

├── falco-daemonset-configmap.yaml

├── falco-event-generator-deployment.yaml

└── falco-service.yaml

<https://github.com/falcosecurity/falco/tree/dev/integrations/k8s-using-daemonset>

// TODO: <https://github.com/falcosecurity/falco/tree/dev/examples/k8s_audit_config>

// TODO: update rules và cách gởi alert của Falco (mặc định falco alert được write vào /var/log/messages của node servers)

vi falco-config/falco.yaml, falco\_rules.yaml, …

kubectl create -f falco-account.yaml

kubectl create -f falco-service.yaml

kubectl create configmap falco-config --from-file=./falco-config

kubectl create -f falco-daemonset-configmap.yaml

### Test:

kubectl get pods

kubectl exec -it falco-74htl bash

# exit

kubectl logs falco-74htl

{"output":"17:48:58.590038385: Notice A shell was spawned in a container with an attached terminal (user=root k8s.pod=falco-74htl container=a98c2aa8e670 shell=bash parent=<NA> cmdline=bash terminal=34816)","priority":"Notice","rule":"Terminal shell in container","time":"2017-12-20T17:48:58.590038385Z", "output\_fields": {"container.id":"a98c2aa8e670","evt.time":1513792138590038385,"k8s.pod.name":"falco-74htl","proc.cmdline":"bash ","proc.name":"bash","proc.pname":null,"proc.tty":34816,"user.name":"root"}}

kubectl create -f falco-event-generator-deployment.yaml

Disaster recovery