Kubernetes Hands-On

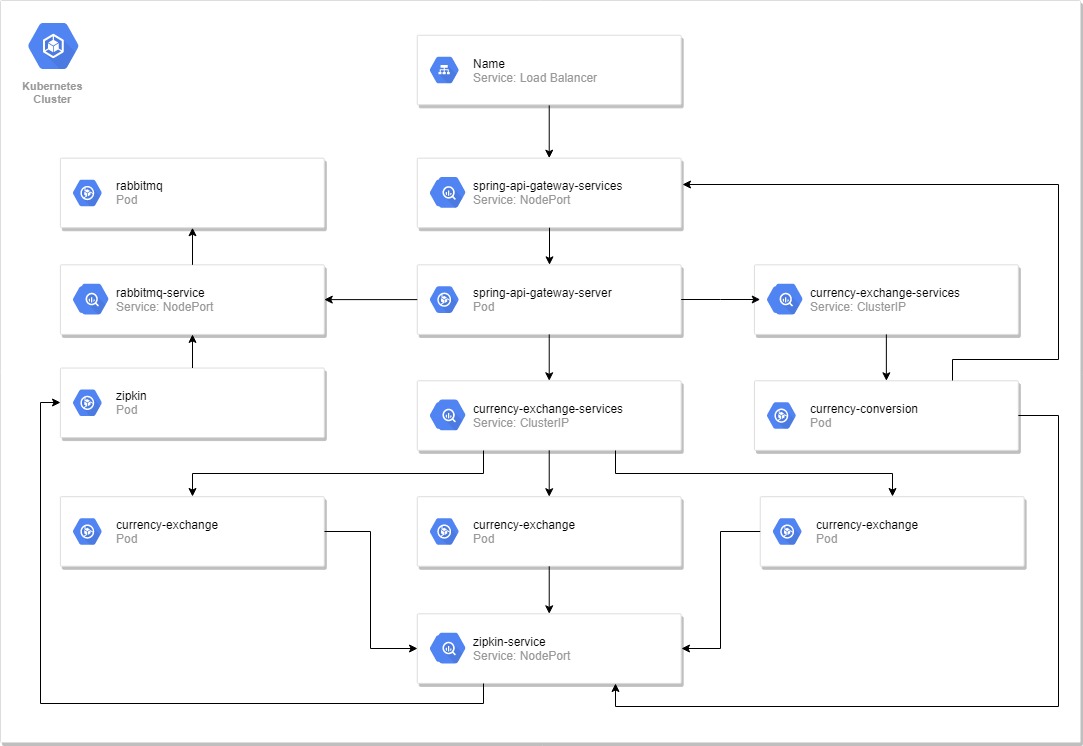
Service Discovery, Distributed Tracing, and Deployment

In this tutorial, we will have practical labs related to Service Discovery, Distributed Tracing, and Deployment in Kubernetes. The technology we will use are :

1. Spring Boot & Cloud
2. Feign
3. Sleuth
4. Zipkin
5. RabbitMQ
6. Zuul API Gateway

# Architecture

The architecture we will use is describe below:



# Connecting to Cluster

1. Using Powershell

We can use Windows Powershell to connecting to our kubernetes cluster at 172.20.103.122, and before that, you need to install Docker for Windows first for building images later. Docker for Windows installer located here: \\cx-id2-file01\Temp\Docker

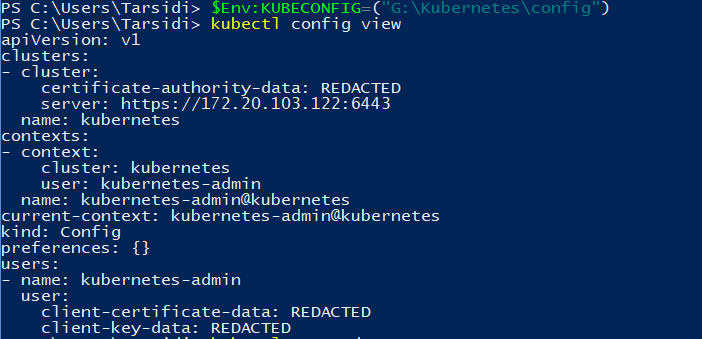
You can get config file from <https://raw.githubusercontent.com/sidie88/service-discovery-in-kubernetes/master/kube-config/config>.

1. Set kubernetes config

$Env:KUBECONFIG=("<YOUR\_CONFIG\_FILE\_LOCATION>\config")

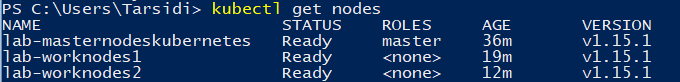
1. Check kubernetes config

kubectl config view

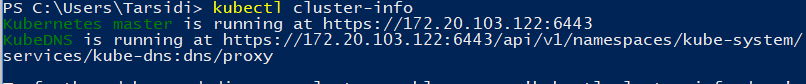


1. Check connection to kubernetes cluster

kubectl get nodes



kubectl cluster-info



1. Using Kubernetes Control Plane

Login into control plane using your credential :

1. Create kubernetes config cluster connection

mkdir .kube

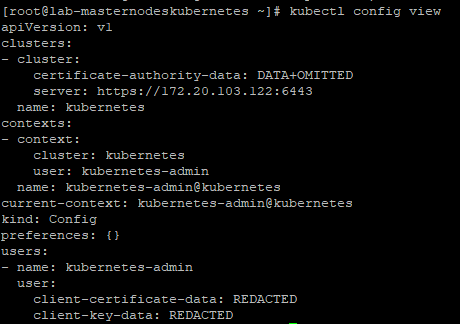
cd .kube/

curl -O <https://raw.githubusercontent.com/sidie88/service-discovery-in-kubernetes/master/kube-config/config>

export KUBECONFIG=$KUBECONFIG:$HOME/.kube/config

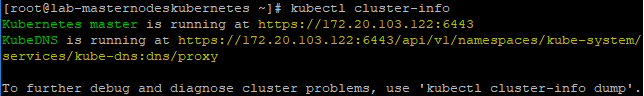
1. Checking your config file

kubectl config view



1. Checking your connection to cluster

kubectl cluster-info



# Clone source code from Github

You need to clone source code by following this command below:

git clone <https://github.com/sidie88/service-discovery-in-kubernetes.git>

cd service-discovery-in-kubernetes

# Zipkin Distributed Server

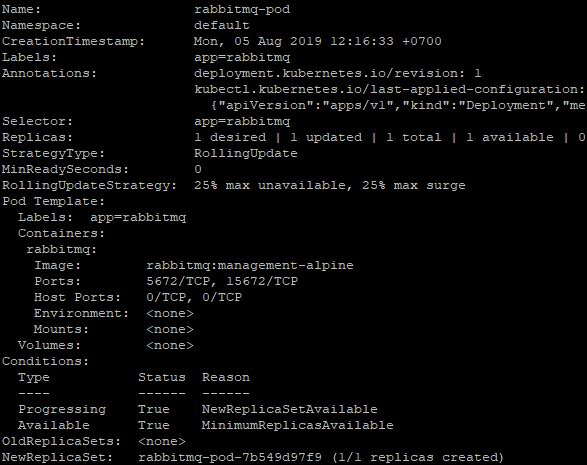
Before deploy zipkin, we need to deploy RabbitMQ first. Zipkin using RabbitMQ to tracing request from other services.

1. Deploy RabbitMQ Server
2. Navigate to distributed-tracing folder
3. Deploy RabbitMQ to Kubernetes cluster

kubectl apply –f rabbitmq.yaml

1. Check deployment status

kubectl describe deployments rabbitmq-pod

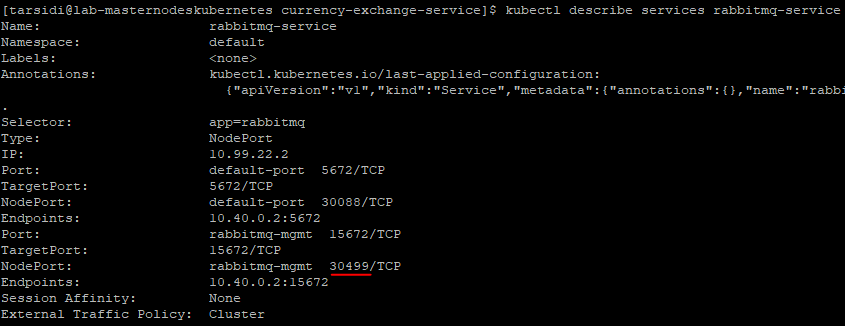


1. Check service endpoint

kubectl get services



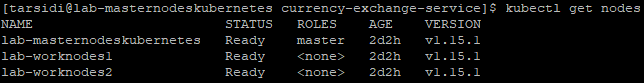
kubectl describe services rabbitmq-service



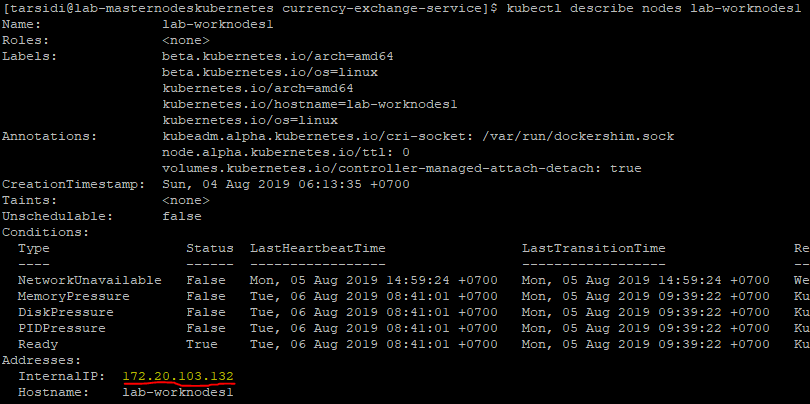
1. Check RabbitMQ Management site

You can access it using IP address from one of worker nodes and NodePort highlighted at previous step (Port: 30499). You can check worker node IP address using this command:

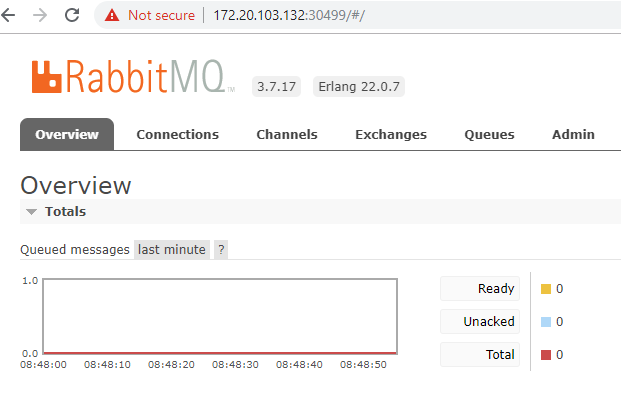
kubectl get nodes



kubectl describe nodes lab-workernodes1



After you get the IP address and port, you can open it using browser and use username “guest” and password “guest” to login.

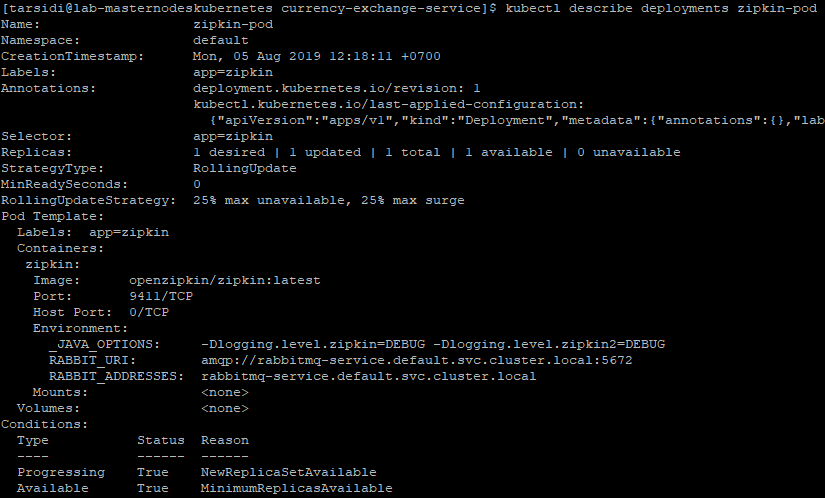


1. Deploy Zipkin Distributed Tracing Server
2. Navigate to distributed-tracing folder
3. Deploy zipkin using yaml file

kubectl apply -f zipkin.yaml

1. Check zipkin deployment

kubectl describe deployments zipkin-pod

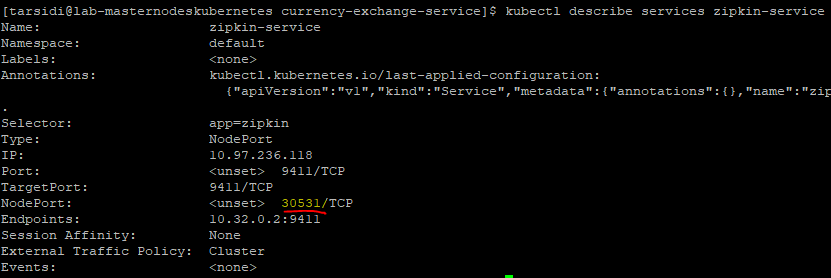


1. Check zipkin service endpoint

kubectl get services

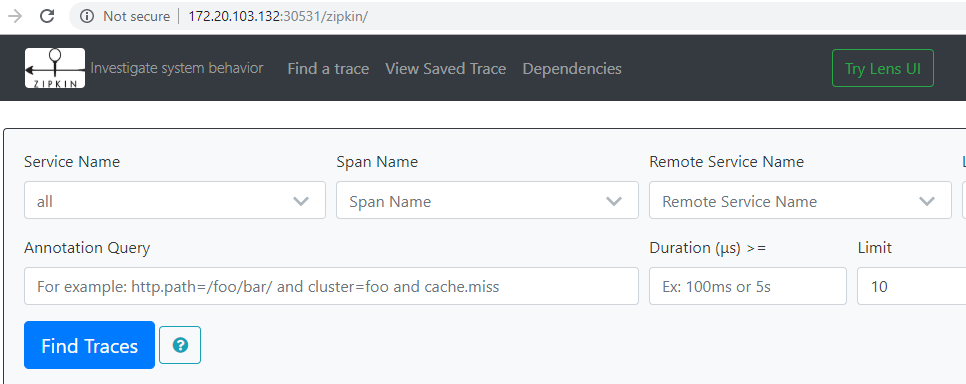


kubectl describe services zipkin-service



1. Access zipking web tracing

Open your browser and use worker node IP and NodePort described at previous step.



# Currency Exchange Service

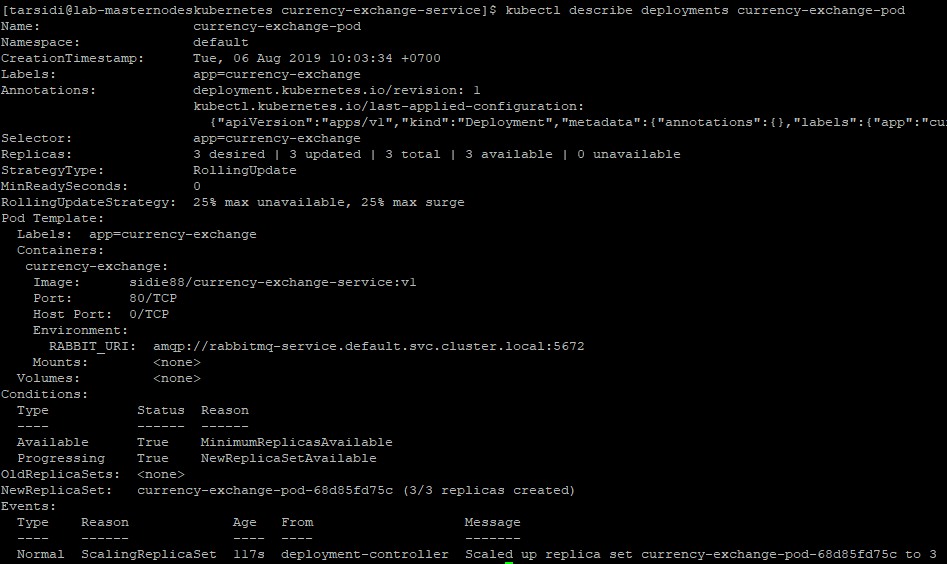
1. Navigate to currency-exchange-service folder
2. Deploy currency-exchange-service to cluster

kubectl apply -f currency-exchange-sevice.yaml



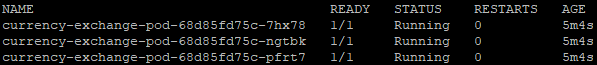
1. Check your deployment status

kubectl describe deployments currency-exchange-pod



1. Check pod status

kubectl get pods --selector=app=currency-exchange

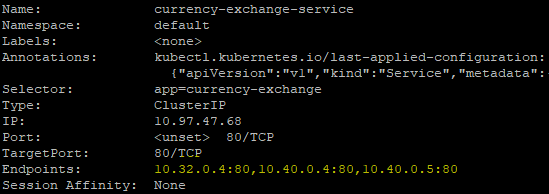


1. Check services endpoint

kubectl get services --field-selector metadata.name=currency-exchange-service



kubectl describe services currency-exchange-service



1. Access currency exchange service endpoint

We cannot access this endpoint directly because service type is ClusterIP, so we need to use port forwarding to access the endpoint using command below.

kubectl port-forward svc/currency-exchange-service 8000:80



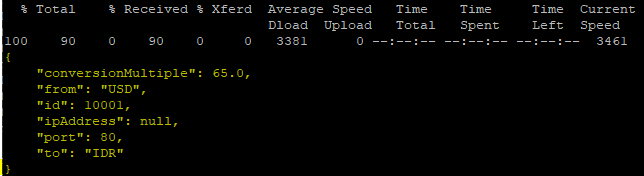
Open other terminal, and execute this command

curl localhost:8000/currency-exchange/from/USD/to/IDR

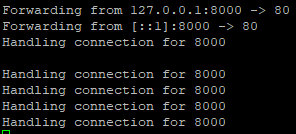


You can use pretty print to format json view

curl localhost:8000/currency-exchange/from/USD/to/IDR | python -m json.tool



At the previous terminal should be printed like this:



# API Gateway Server

For API Gateway server, we will use Netflix Zuul API Gateway Server. Every request should be through this server to centralize request and make tracing request easier.

1. Register all services to zuul routes in application.properties, line below is sample how to add service to zuul routes.

* Route path

Route path is URI used to call service through zuul api gateway. It’s format config is zuu.routes.<SERVICE\_NAME>.path=api path.

Example :

zuul.routes.currencyexchange.path=/currency-exchange-service/\*\*

Configuration above will set currency-exchange-service as URI at api gateway, so url will be: http://<ZUUL\_API\_GATEWAY\_SERVICE\_ENDPOINT>:8762/currency-exchange-service

* Route URL

Route url is the redirection from route path above.

Example:

zuul.routes.currencyexchange.url=http://currency-exchange-service.default.svc.cluster.local

Configuration above will redirect request from http://<ZUUL\_API\_GATEWAY\_SERVICE\_ENDPOINT>:8762/currency-exchange-service to <http://currency-exchange-service.default.svc.cluster.local>

1. Build docker image