A Cluster with App-Only Image and Load Balancer

Overview

In our steps, we'll create a two-node Yellowfin cluster, with each node allocated 6GB of RAM, with Traefik forwarding requests from port 80 (standard HTTP port) of your DNS hostname or IP address to Yellowfin instances.

When deploying Yellowfin with Traefik fronting it, the Yellowfin Kubernetes service will default to the "ClusterIP" service type in Kubernetes, so it will not expose any ports to the external interface of the Kubernetes cluster.

Before deploying a cluster with these defaults, make sure you have already created a repository database and synced it with the same version of Yellowfin that will be used in the Yellowfin container. To do this, download the full application installer for Yellowfin, and install it on your workstation. This will create a Yellowfin repo DB as well as an instance of Yellowfin in a folder which can be deleted after configuring the containers.

For a list of supported database types, see the database information on Install And Deploy Yellowfin.

To deploy a Yellowfin cluster, follow the steps below.

- 1. Install the full application installer version of Yellowfin on your workstation (this is temporary to ensure the repo DB is available for the containers to use)
- 2. Copy the web.xml file from this installation and save it as a backup to your preferred location (this acts as a reference for the Yellowfin credentials required to connect to your Yellowfin repo DB)
- 3. Ensure Kubernetes is running and that Traefik has been installed
- 4. Copy the following text and paste it into your preferred text editor:

```
### Yellowfin Cluster Service ###
apiVersion: v1
kind: Service
metadata:
 name: yellowfin-cluster
 ports:
    - protocol: TCP
     name: web
     port: 8080
  selector:
   app: yellowfin-cluster
### Yellowfin Cluster Deployment ###
kind: Deployment
apiVersion: apps/v1
metadata:
 namespace: default
 name: vellowfin-cluster
   app: yellowfin-cluster
spec:
  replicas: 2
  selector:
   matchLabels:
     app: yellowfin-cluster
  template:
   metadata:
      labels:
       app: yellowfin-cluster
    spec:
      containers:
          - name: APP_MEMORY
           value: "6144"
          - name: CLUSTER_PORT
           value: "7800"
          - name: JDBC CLASS NAME
           value: INSERT_DATABASE_TYPE_HERE
          - name: JDBC_CONN_ENCRYPTED
           value: "true"
          - name: JDBC_CONN_PASS
```

```
value: INSERT_JDBC_PASSWORD_HERE
          - name: JDBC_CONN_URL
            value: jdbc:INSERT_JDBC_CONNECTION_STRING_HERE
          - name: JDBC_CONN_USER
            value: INSERT_DATABASE_USER_HERE
          - name: NODE BACKGROUND TASKS
            value: REPORT_BROADCAST_BROADCASTTASK, REPORT_BROADCAST_MIREPORTTASK, FILTER_CACHE,
\verb|SOURCE_FILTER_REFRESH, \verb|SOURCE_FILTER_UPDATE_REMINDER, THIRD_PARTY_AUTORUM, ORGREF_CODE_REFRESH, \\
ETL_PROCESS_TASK,SIGNALS_DCR_TASK,SIGNALS_ANALYSIS_TASK,SIGNALS_CLEANUP_TASK,COMPOSITE_VIEW_REFRESH,
SIGNALS CORRELATION TASK
          - name: NODE_PARALLEL_TASKS
           value: 4,4,4,4,4,4,4,4,4,4,4,4,4,4
          name: yellowfin-cluster
          image: yellowfinbi/yellowfin-app-only:<RELEASE_VERSION_GOES_HERE>
          ports:
            - name: web
              containerPort: 8080
### Yellowfin Cluster Ingress ###
apiVersion: traefik.containo.us/vlalphal
kind: IngressRoute
metadata:
 name: yellowfiningressroute
 namespace: default
spec:
 entryPoints:
    - webs
 routes:
  - match: Host(`INSERT_DNS_HOSTNAME`)
   kind: Rule
   services:
    - name: yellowfin-cluster
      port: 8080
      sticky:
        cookie:
         httpOnly: true
          name: stickyCookie
```

- 5. Read through the above text and replace the database connection settings with your own configuration details (these are located in the web.xml file of the Yellowfin installation)
- 6. In the text above, replace the fext INSERT_DNS_HOSTNAME with your own DNS name (or IP address) for Traefik to listen to, for routing requests to the Yellowfin instance. In the example below, we've added an example DNS name:

```
### Yellowfin Cluster Ingress ###
apiVersion: traefik.containo.us/vlalphal
kind: IngressRoute
metadata:
 name: yellowfiningressroute
 namespace: default
spec:
 entryPoints:
   - webs
  routes:
  - match: Host(`yellowfin.example.com`)
   kind: Rule
   services:
    - name: yellowfin-cluster
     port: 8080
     stickv:
       cookie:
          httpOnly: true
          name: stickyCookie
```

- 7. Save the text to a YAML file called **yellowfin-cluster.yml**
- 8. Run the following command in a terminal to deploy Yellowfin and execute it in the background: Kubectl apply -f yellowfin-cluster.yml
- 9. Start Yellowfin by typing your host URL

- 10. Ensure that Yellowfin is running from your cluster and that you can login (this confirms that your login credentials are correct, so you can safely delete the workstation instance of Yellowfin)
- 11. Delete the workstation instance of Yellowfin by removing the folder



If you're using AWS EKS, or having issues clustering your Yellowfin instances, merge the following environment variable **CLUSTER_INTERFACE** and value into the **containers** section of your Kubernetes deployment file:

containers:
-env
name: CLUSTER_INTERFACE
value: "match-interface:ethl"

top

Section navigation Current topic - Install in a Container

The page is part of the Install in a Container topic contains the following pages, split by Docker and Kubernetes:

A Cluster with App-Only Image and Load Balancer

- Deploy to Docker without Swarm
 - Sandbox Instance with All-In-One Image
 - Single Instance with App-Only Image
 - Multiple Discrete Instances with App-Only Image
 - A Cluster with App-Only Image
- Deploy to Docker with Swarm
 - Sandbox instance with All-In-One Image Swarm
 - Single Instance with App-Only Image Swarm
 - Multiple Discrete Instances with App-Only Image Swarm
 - A Cluster with App-Only Image Swarm

Kubernetes

- Deploy to Kubernetes without load balancing
 - Sandbox Instance with All-In-One Image no Load Balancer
 - Multiple Discrete Instances with App-Only Image no Load Balancer
- Deploy to Kubernetes with Load Balancing
 - Single Instance with App-Only Image and Load Balancer
 - A Cluster with App-Only Image and Load Balancer

This page is part of the Install And Deploy Yellowfin section of the wiki, which has these topics:

Install on Premises

A Cluster with App-Only Image and Load Balancer

Installation Steps

Install in the Cloud

Install in the Cloud

- Yellowfin for AWS
- Yellowfin for Azure
- Yellowfin for Google Cloud Platform

Install in a container

Install in a Container

- Docker
- KubernetesUpgrading Yellowfin Container Deployment

Deploy Yellowfin

Deploy Yellowfin

- Logs and LoggingYellowfin Directory
- Structure
- User Welcome

Advanced Deployments

Advanced Deployments

- Clustering GuideYellowfin Server
- Specification

 Automate Yellowfin
 Deployment on Linux

 SAML Bridge

- Standalone Configuration Tools