A Cluster with App-Only Image

Overview

In our steps for setting up a cluster deployment of Yellowfin, each node of Yellowfin has 4GB of allocated RAM, with ports starting on the Docker host at 8080, then 8081, 8082 and so on.

As each Yellowfin instance is running on a separate port, we recommend placing a load balancer or reverse proxy with sticky session support in front of the instances so that users are directed to one of Yellowfin instances for the duration of their session.

Before deploying a Yellowfin cluster, 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.

In the steps below, we'll show you how to deploy a Yellowfin cluster with three nodes.

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 Docker is running
4. Copy the following text and paste it into your preferred text editor:

```
version: '3'
services:
  yellowfin-cluster-node-1:
   ports:
      - "8080:8080" # Maps Yellowfin running on port 8080 to the host's port 8080
      #- "7801:7800" # Maps the Yellowfin cluster port to an external port on the host (Optional)
environment:
      # Required environment variables
      - JDBC_CLASS_NAME=INSERT_DATABASE_TYPE_HERE # Database driver class name
      - JDBC_CONN_URL=jdbc:INSERT_JDBC_CONNECTION_STRING_HERE # Database connection string
      - JDBC_CONN_USER=INSERT_DATABASE_USER_HERE # Username to use when accessing the database
      - JDBC_CONN_PASS=INSERT_JDBC_PASSWORD_HERE # Password for the database user
      - JDBC_CONN_ENCRYPTED=true # Flag for indicating if the database user's password supplied is
encrypted or not.
      - APP_MEMORY=4096 # The amount of memory in megabytes to assign to the Yellowfin Application.
      - CLUSTER_ADDRESS=yellowfin-cluster-node-1 # Address to use for clustering – recommended to use
Docker networking to connect the containers
      - CLUSTER_PORT=7800 # TCP Port to use for cluster networking
      - NODE_BACKGROUND_TASKS=REPORT_BROADCAST_BROADCASTTASK,REPORT_BROADCAST_MIREPORTTASK,FILTER_CACHE,
SOURCE_FILTER_REFRESH,SOURCE_FILTER_UPDATE_REMINDER,THIRD_PARTY_AUTORUN,ORGREF_CODE_REFRESH,
ETL_PROCESS_TASK,SIGNALS_DCR_TASK,SIGNALS_ANALYSIS_TASK,SIGNALS_CLEANUP_TASK,COMPOSITE_VIEW_REFRESH,
SIGNALS_CORRELATION_TASK # Comma separated list of which background Task Types can be run on this node.
      - NODE_PARALLEL_TASKS=4,4,4,4,4,4,4,4,4,4,4,4,4 # Comma separated list of the number of concurrent
tasks for each Task Type that can be run on
   image: "yellowfinbi/yellowfin-app-only:<RELEASE_VERSION_GOES_HERE>" # Path to the app-only image of
Yellowfin

  yellowfin-cluster-node-2:
   ports:
      - "8081:8080" # Maps Yellowfin running on port 8081 to the host's port 8080
      #- "7802:7800" # Maps the Yellowfin cluster port to an external port on the host (Optional)
   environment:
      # Required environment variables
      - JDBC_CLASS_NAME=INSERT_DATABASE_TYPE_HERE # Database driver class name
      - JDBC_CONN_URL=jdbc:INSERT_JDBC_CONNECTION_STRING_HERE # Database connection string
      - JDBC_CONN_USER=INSERT_DATABASE_USER_HERE # Username to use when accessing the database
      - JDBC_CONN_PASS=INSERT_JDBC_PASSWORD_HERE # Password for the database user
      - JDBC_CONN_ENCRYPTED=true # Flag for indicating if the database user's password supplied is
encrypted or not.
      - APP_MEMORY=4096 # The amount of memory in megabytes to assign to the Yellowfin Application.
      - CLUSTER_ADDRESS=yellowfin-cluster-node-2 # Address to use for clustering – recommended to use
Docker networking to connect the containers
      - CLUSTER_PORT=7800 # TCP Port to use for cluster networking
   image: "yellowfinbi/yellowfin-app-only:<RELEASE_VERSION_GOES_HERE>" # Path to the app-only image of
Yellowfin
```

```
      - NODE_BACKGROUND_TASKS=REPORT_BROADCAST_BROADCASTTASK,REPORT_BROADCAST_MIREPORTTASK,FILTER_CACHE,
SOURCE_FILTER_REFRESH,SOURCE_FILTER_UPDATE_REMINDER,THIRD_PARTY_AUTORUN,ORGREF_CODE_REFRESH,
ETL_PROCESS_TASK,SIGNALS_DCR_TASK,SIGNALS_ANALYSIS_TASK,SIGNALS_CLEANUP_TASK,COMPOSITE_VIEW_REFRESH,
SIGNALS_CORRELATION_TASK # Comma separated list of which background Task Types can be run on this node.
      - NODE_PARALLEL_TASKS=4,4,4,4,4,4,4,4,4,4,4,4,4,4 # Comma separated list of the number of concurrent
tasks for each Task Type that can be run on

  yellowfin-cluster-node-3:
   ports:
      - "8082:8080" # Maps Yellowfin running on port 8082 to the host's port 8080
      #- "7803:7800" # Maps the Yellowfin cluster port to an external port on the host (Optional)
   environment:
      # Required environment variables
      - JDBC_CLASS_NAME=INSERT_DATABASE_TYPE_HERE # Database driver class name
      - JDBC_CONN_URL=jdbc:INSERT_JDBC_CONNECTION_STRING_HERE # Database connection string
      - JDBC_CONN_USER=INSERT_DATABASE_USER_HERE # Username to use when accessing the database
      - JDBC_CONN_PASS=INSERT_JDBC_PASSWORD_HERE # Password for the database user
      - JDBC_CONN_ENCRYPTED=true # Flag for indicating if the database user's password supplied is
encrypted or not.
      - APP_MEMORY=4096 # The amount of memory in megabytes to assign to the Yellowfin Application.
      - CLUSTER_ADDRESS=yellowfin-cluster-node-3 # Address to use for clustering – recommended to use
Docker networking to connect the containers
      - CLUSTER_PORT=7800 # TCP Port to use for cluster networking
   image: "yellowfinbi/yellowfin-app-only:<RELEASE_VERSION_GOES_HERE>" # Path to the app-only image of
Yellowfin
```

5. Read through the above text and, for each container, replace the environment variable placeholders with your own configuration details (these are located in the web.xml file of the Yellowfin installation); here’s an example to connect to a PostgreSQL instance:

```
      # Required environment variables
      - JDBC_CLASS_NAME=org.postgresql.Driver # Database driver class name
      - JDBC_CONN_URL=jdbc:postgresql://192.168.1.50/docker_yellowfin _cluster # Database connection
string
      - JDBC_CONN_USER=postgres # Username to use when accessing the database
      - JDBC_CONN_PASS=bXF0oj5gnB1oRB1kZq5 # Password for the database user
      - JDBC_CONN_ENCRYPTED=true # Flag for indicating if the database user's password supplied is
encrypted or not.
      - APP_MEMORY=4096 # The amount of memory in megabytes to assign to the Yellowfin Application.
      - CLUSTER_ADDRESS=yellowfin-cluster-node-1 # Address to use for clustering – recommended to use
Docker networking to connect the containers
      - CLUSTER_PORT=7800 # TCP Port to use for cluster networking
   image: "yellowfinbi/yellowfin-app-only:9.6.0" # Path to the app-only image of Yellowfin
```

6. Save the text to a YAML file called **yellowfin-cluster.yml**
7. Run the following command in a terminal to deploy Yellowfin and execute it in the background:
   `docker-compose up -d -f yellowfin-cluster.yml`
8. Start Yellowfin by typing your host URL on port 8080 (or any other port you’ve set)
9. Ensure that Yellowfin is running from your container and that you can login (this confirms that your login credentials are correct, so you can safely delete the workstation instance of Yellowfin)
10. Delete the workstation instance of Yellowfin by removing the folder

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

- 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

- 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
- Kubernetes
- Upgrading Yellowfin Container Deployment

Deploy Yellowfin

Deploy Yellowfin

- Logs and Logging
- Yellowfin Directory Structure
- User Welcome

Advanced Deployments

Advanced Deployments

- Clustering Guide
- Yellowfin Server Specification
- Automate Yellowfin Deployment on Linux
- SAML Bridge
- Standalone Configuration Tools

top