Multiple Discrete Instances with App-Only Image

Overview

In our steps for setting up a multiple discrete instance deployment of Yellowfin, each instance of Yellowfin has 4GB of allocated RAM, with ports starting at 8080, then 8090, and so on.

Before deploying your instances 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 — for each 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 two discrete instances of Yellowfin.

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-multi-instance-prod:
   ports:
      - "8080:8080" # Maps Yellowfin running on port 8080 to the host's port 8080 environment:
      # Required environment variables
      - JDBC_CLASS_NAME=INSERT_DATABASE_TYPE_1_HERE # Database driver class name
      - JDBC_CONN_URL=jdbc:INSERT_JDBC_CONNECTION_STRING_1_HERE # Database connection string
      - JDBC_CONN_USER=INSERT_DATABASE_USER_1_HERE # Username to use when accessing the database
      - JDBC_CONN_PASS=INSERT_JDBC_PASSWORD_1_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=8192 # The amount of memory in megabytes to assign to the Yellowfin Application.
   image: "yellowfinbi/yellowfin-app-only:<RELEASE_VERSION_GOES_HERE>" # Path to the app-only image of
Yellowfin

  yellowfin-multi-instance-dev:
   ports:
      - "8090:8080" # Maps Yellowfin running on port 8090 to the host's port 8080
   environment:
      # Required environment variables
      - JDBC_CLASS_NAME=INSERT_DATABASE_TYPE_2_HERE # Database driver class name
      - JDBC_CONN_URL=jdbc:INSERT_JDBC_CONNECTION_STRING_2_HERE # Database connection string
      - JDBC_CONN_USER=INSERT_DATABASE_USER_2_HERE # Username to use when accessing the database
      - JDBC_CONN_PASS=INSERT_JDBC_PASSWORD_2_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.
   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_multiple_yellowfin _instances # 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.
   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-multiple-instances.yml**
7. Run the following command in a terminal to deploy Yellowfin and execute it in the background:
`docker-compose up -d -f yellowfin-multiple-instances.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 containers 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:

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

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