



**Yellowfin on Docker & Kubernetes**

Table of Contents

[1. Introduction 3](#_Toc40099556)

[1.1 Yellowfin’s Docker History 3](#_Toc40099557)

[2. Yellowfin Docker Image Types 4](#_Toc40099558)

[2.1 Yellowfin All-In-One Image 4](#_Toc40099559)

[2.2 Yellowfin All-Only Image 4](#_Toc40099560)

[3. Yellowfin on Docker 5](#_Toc40099561)

[3.1 Deploying Yellowfin on a standalone Docker Server (non-Swarm) 5](#_Toc40099562)

[3.1.1 Deploying a sandbox Yellowfin instance – The All-In-One image 5](#_Toc40099563)

[3.2.2 Deploying a single instance of Yellowfin – App-Only image 5](#_Toc40099564)

[3.2.3 Deploying multiple discreet instances of Yellowfin – App-Only image 7](#_Toc40099565)

[3.2.3 Deploying a Yellowfin Cluster – App-Only image 8](#_Toc40099566)

# Introduction

This document serves as a technical guide to provide the best practices for deploying Yellowfin in Docker – Docker and Docker Swarm - and Kubernetes environments.

In this white paper, I will detail different Yellowfin setups that can be deployed, their use cases and provide deployment files that can be used as a baseline for you to deploy Yellowfin into your Docker and/or Kubernetes environments.

# Yellowfin Docker Image Types

In the section, I will be covering the 2 different Docker images that Yellowfin will be publishing, as well as use cases for the Docker images. Please note that

## 2.1 Yellowfin All-In-One Image

The Yellowfin All-In-One image contains both the Yellowfin application as well as the Yellowfin repository database, which in this case is an embedded PostgreSQL database.

The recommended use cases for this image are:

* Short-term POCs and Demos.
* Testing Yellowfin functionality in a sandboxed environment.

This image will not persist data outside of the Docker container, and all content will be lost when the container is destroyed.

It is not recommended to use this image for any production deployments.

This Docker image also has the limitation that it cannot be used in a clustered environment.

## 2.2 Yellowfin All-Only Image

The Yellowfin App Only image contains only the Yellowfin application and must be connected to an existing repository database.

Yellowfin instances deployed using this image can be deployed as discreet instances or as part of a Yellowfin Cluster. This image is suitable for both production and non-production environments, as data is persisted on the external Yellowfin repository, so that no Yellowfin data is lost when containers are destroyed.

The recommended use cases for this image are:

* Long-term instances of Yellowfin, where data persistence is important.
* Clustered Yellowfin deployments.

# Yellowfin on Docker

With Yellowfin on Docker, we will be covering how to deploy Yellowfin on a single Docker server, using Docker Compose, and how to deploy Yellowfin in a Docker Swarm environment, using a Docker Stack file.

## 3.1 Deploying Yellowfin on a standalone Docker Server (non-Swarm)

For deploying Yellowfin on a single Docker server, we recommend using Docker Compose to run the included deployment files to deploy Yellowfin.

### Deploying a sandbox Yellowfin instance – The All-In-One image

The simplest Yellowfin deployment, to deploy a self-contained instance of Yellowfin, save the below YAML markup to a file. In our example, we will save the file to “*yellowfin-all-in-one.yml*”

version: '3'

services:

yellowfin-all-in-one:

ports:

- "8080:8080" # Maps Yellowfin running on port 8080 to the host's port 8080

environment:

- APP\_MEMORY=4096 # The amount of memory in megabytes to assign to the Yellowfin Application.

image: "<DOCKER\_HUB\_ACCOUNT\_GOES\_HERE>/yellowfin-all-in-one:<RELEASE\_VERSION\_GOES\_HERE>"

Then run the following command in a terminal to deploy Yellowfin and execute it in the background.

docker-compose up -d -f yellowfin-all-in-one.yml

The result of the above deployment file example will be a Yellowfin All-In-One instance running on your Docker host’s port 8080, with 4GB of RAM allocated to Yellowfin.

### 3.2.2 Deploying a single instance of Yellowfin – App-Only image

For this deployment, you will need to have a Yellowfin Repository database already created and synced with same version of Yellowfin as the one that will be used in the Yellowfin container.

Save the following YAML markup to a file, which in this example we will be saving to the file “yellowfin-single-instance.yml”

With the above example, you will need to substitute in your database connection settings, which can be found in your existing Yellowfin installation, inside the Yellowfin web.xml file.

version: '3'

services:

yellowfin-standalone-single-instance:

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\_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.

# Optional environment variables

# - JDBC\_MAX\_COUNT=25 # The maximum amount of connections available to the connection pool connecting to the repository database

# - WELCOME\_PAGE=custom\_index.jsp # Option to change the login page

# - APP\_SERVER\_PORT=8080 # Controls the port that Yellowfin will listen on (HTTP port)

# - APP\_SHUTDOWN\_PORT=9093 # Controls the port that Yellowfin listens on for shutdown commands (TCP port)

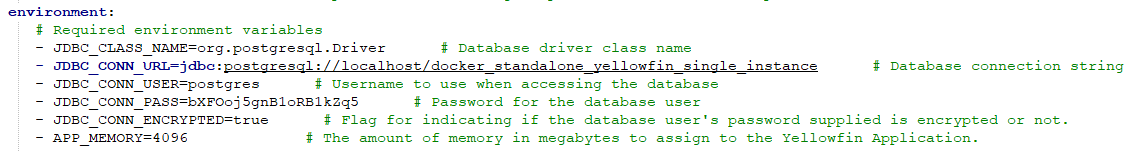
# - PROXY\_PORT=443 # External proxy configuration for Yellowfin - HTTP/S port

# - PROXY\_SCHEME=https # External proxy configuration for Yellowfin - Connection scheme: HTTP/S

# - PROXY\_HOST=reporting.example.com # External proxy configuration for Yellowfin - The URL for the external proxy

image: "<DOCKER\_HUB\_ACCOUNT\_GOES\_HERE>/yellowfin-app-only: <RELEASE\_VERSION\_GOES\_HERE>" # Path to your app-only image of Yellowfin

As an example, here is an example of those values setup to connect to a local PostgreSQL instance.



Then run the following command in a terminal to deploy Yellowfin and execute it in the background.

docker-compose up -d -f yellowfin-single-instance.yml

The result of the above deployment file example will be a Yellowfin App-Only instance running on your Docker host’s port 8080, with 4GB of RAM allocated to Yellowfin.

### 3.2.3 Deploying multiple discreet instances of Yellowfin – App-Only image

As with the single-instance approach, for each discreet instance of Yellowfin that you want to deploy, you will need to have initialized a separate Yellowfin Repository database, which needs to be synced with same version of Yellowfin as the one that will be used in the Yellowfin container that is connecting to it.

Save the following YAML markup to a file, which in this example we will be saving to the file “yellowfin-multiple-instances.yml”

|  |
| --- |
| 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\_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.  # Optional environment variables  # - JDBC\_MAX\_COUNT=25 # The maximum amount of connections available to the connection pool connecting to the repository database  # - WELCOME\_PAGE=custom\_index.jsp # Option to change the login page  # - APP\_SERVER\_PORT=8080 # Controls the port that Yellowfin will listen on (HTTP port)  # - APP\_SHUTDOWN\_PORT=9093 # Controls the port that Yellowfin listens on for shutdown commands (TCP port)  # - PROXY\_PORT=443 # External proxy configuration for Yellowfin - HTTP/S port  # - PROXY\_SCHEME=https # External proxy configuration for Yellowfin - Connection scheme: HTTP/S  # - PROXY\_HOST=reporting.example.com # External proxy configuration for Yellowfin - The URL for the external proxy  image: "<DOCKER\_HUB\_ACCOUNT\_GOES\_HERE>/yellowfin-app-only: <RELEASE\_VERSION\_GOES\_HERE>" # Path to your 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\_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.  # Optional environment variables  # - JDBC\_MAX\_COUNT=25 # The maximum amount of connections available to the connection pool connecting to the repository database  # - WELCOME\_PAGE=custom\_index.jsp # Option to change the login page  # - APP\_SERVER\_PORT=8080 # Controls the port that Yellowfin will listen on (HTTP port)  # - APP\_SHUTDOWN\_PORT=9093 # Controls the port that Yellowfin listens on for shutdown commands (TCP port)  # - PROXY\_PORT=443 # External proxy configuration for Yellowfin - HTTP/S port  # - PROXY\_SCHEME=https # External proxy configuration for Yellowfin - Connection scheme: HTTP/S  # - PROXY\_HOST=reporting.example.com # External proxy configuration for Yellowfin - The URL for the external proxy  image: "<DOCKER\_HUB\_ACCOUNT\_GOES\_HERE>/yellowfin-app-only: <RELEASE\_VERSION\_GOES\_HERE>" # Path to your app-only image of Yellowfin |

With the above example, as with the single Yellowfin instance example, you will need to substitute in your database connection settings, which can be found in your existing Yellowfin installations, inside the Yellowfin web.xml file.

Then 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

The result of the above deployment file example will be 2 Yellowfin App-Only instances running on your Docker host, one on port 8080, which will be allocated 8GB of RAM, and the other on port 8090, which will be allocated 4GB of RAM.

### 3.2.4 Deploying a Yellowfin Cluster – App-Only image

For this deployment, you will need to have a Yellowfin Repository database already created and synced with same version of Yellowfin as the one that will be used in the Yellowfin containers.

Save the following YAML markup to a file, which in this example we will be saving to the file “yellowfin-cluster.yml”

|  |
| --- |
| 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  # Optional environment variables  # - JDBC\_MAX\_COUNT=25 # The maximum amount of connections available to the connection pool connecting to the repository database  # - WELCOME\_PAGE=custom\_index.jsp # Option to change the the login page  # - APP\_SERVER\_PORT=8080 # Controls the port that Yellowfin will listen on (HTTP port)  # - APP\_SHUTDOWN\_PORT=9093 # Controls the port that Yellowfin listens on for shutdown commands (TCP port)  # - PROXY\_PORT=443 # External proxy configuration for Yellowfin - HTTP/S port  # - PROXY\_SCHEME=https # External proxy configuration for Yellowfin - Connection scheme: HTTP/S  # - PROXY\_HOST=reporting.example.com # External proxy configuration for Yellowfin - The URL for the external proxy  image: "<DOCKER\_HUB\_ACCOUNT\_GOES\_HERE>/yellowfin-app-only: <RELEASE\_VERSION\_GOES\_HERE>" # Path to your 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  # Optional environment variables  # - JDBC\_MAX\_COUNT=25 # The maximum amount of connections available to the connection pool connecting to the repository database  # - WELCOME\_PAGE=custom\_index.jsp # Option to change the the login page  # - APP\_SERVER\_PORT=8080 # Controls the port that Yellowfin will listen on (HTTP port)  # - APP\_SHUTDOWN\_PORT=9093 # Controls the port that Yellowfin listens on for shutdown commands (TCP port)  # - PROXY\_PORT=443 # External proxy configuration for Yellowfin - HTTP/S port  # - PROXY\_SCHEME=https # External proxy configuration for Yellowfin - Connection scheme: HTTP/S  # - PROXY\_HOST=reporting.example.com # External proxy configuration for Yellowfin - The URL for the external proxy  image: "<DOCKER\_HUB\_ACCOUNT\_GOES\_HERE>/yellowfin-app-only: <RELEASE\_VERSION\_GOES\_HERE>" # Path to your app-only image of Yellowfin  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  # Optional environment variables  # - JDBC\_MAX\_COUNT=25 # The maximum amount of connections available to the connection pool connecting to the repository database  # - WELCOME\_PAGE=custom\_index.jsp # Option to change the the login page  # - APP\_SERVER\_PORT=8080 # Controls the port that Yellowfin will listen on (HTTP port)  # - APP\_SHUTDOWN\_PORT=9093 # Controls the port that Yellowfin listens on for shutdown commands (TCP port)  # - PROXY\_PORT=443 # External proxy configuration for Yellowfin - HTTP/S port  # - PROXY\_SCHEME=https # External proxy configuration for Yellowfin - Connection scheme: HTTP/S  # - PROXY\_HOST=reporting.example.com # External proxy configuration for Yellowfin - The URL for the external proxy  image: "<DOCKER\_HUB\_ACCOUNT\_GOES\_HERE>/yellowfin-app-only: <RELEASE\_VERSION\_GOES\_HERE>" # Path to your app-only image of Yellowfin |

With the above example, as with the other deployment examples, you will need to substitute in your database connection settings, which can be found in your existing Yellowfin installations, inside the Yellowfin web.xml file.

Then run the following command in a terminal to deploy Yellowfin and execute it in the background.

docker-compose up -d -f yellowfin-cluster.yml

The result of the above deployment file example will be a 3 node Yellowfin Cluster, each allocated 4GB of RAM and running on ports 8080, 8081 and 8082 on the Docker host.