Multiple Discrete Instances with App-Only Image - no Load Balancer

Overview

In our steps for setting up a multiple discrete instance of Yellowfin, we'll create two Yellowfin instances using the **LoadBalancer** service type in Kubernetes, so that each Yellowfin instance can be accessed on port 8080 of the provisioned load balancer.

The **Production** instance of Yellowfin will be allocated 6GB of RAM, and the **Development** instance will be allocated 4GB of RAM. In a Kubernetes environment where a load balancer cannot be provisioned during the deployment (eg, a basic on-premises environment), the services specification can be modified to switch the ServiceType to NodePort.

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

```
---
### Yellowfin Production Instance - Service ###
apiVersion: v1
kind: Service
metadata:
  name: yellowfin-multi-instance-prod
spec:
  ports:
  - name: "web"
    port: 8080
    targetPort: 8080
  selector:
    app: yellowfin-multi-instance-prod
  type: LoadBalancer
status:
  loadBalancer: {}
---
### Yellowfin Development Instance - Service ###
apiVersion: v1
kind: Service
metadata:
  name: yellowfin-multi-instance-dev
spec:
  ports:
  - name: "web"
    port: 8080
    targetPort: 8080
  selector:
    app: yellowfin-multi-instance-dev
  type: LoadBalancer
status:
  loadBalancer: {}
---
### Yellowfin Production Instance - Deployment ###
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: default
  labels:
    app: yellowfin-multi-instance-prod
  name: yellowfin-multi-instance-prod
spec:
```

```yaml
    replicas: 1
    selector:
      matchLabels:
        app: yellowfin-multi-instance-prod
    template:
      metadata:
        labels:
          app: yellowfin-multi-instance-prod
      spec:
        containers:
          - env:
            - name: APP_MEMORY
              value: "6144"
            - name: JDBC_CLASS_NAME
              value: INSERT_DATABASE_TYPE_1_HERE
            - name: JDBC_CONN_ENCRYPTED
              value: "true"
            - name: JDBC_CONN_PASS
              value: INSERT_JDBC_PASSWORD_1_HERE
            - name: JDBC_CONN_URL
              value: jdbc:INSERT_JDBC_CONNECTION_STRING_1_HERE
            - name: JDBC_CONN_USER
              value: INSERT_DATABASE_USER_1_HERE
            name: yellowfin-multi-instance-prod
            image: yellowfinbi/yellowfin-app-only:<RELEASE_VERSION_GOES_HERE>
            ports:
              - name: web
                containerPort: 8080
---
### Yellowfin Development Instance - Deployment ###
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: default
  labels:
    app: yellowfin-multi-instance-dev
  name: yellowfin-multi-instance-dev

spec:
  replicas: 1
  selector:
    matchLabels:
      app: yellowfin-multi-instance-dev
  template:
    metadata:
      labels:
        app: yellowfin-multi-instance-dev
    spec:
      containers:
        - env:
          - name: APP_MEMORY
            value: "4096"
          - name: JDBC_CLASS_NAME
            value: INSERT_DATABASE_TYPE_2_HERE
          - name: JDBC_CONN_ENCRYPTED
            value: "true"
          - name: JDBC_CONN_PASS
            value: INSERT_JDBC_PASSWORD_2_HERE
          - name: JDBC_CONN_URL
            value: jdbc: INSERT_JDBC_CONNECTION_2_STRING_HERE
          - name: JDBC_CONN_USER
            value: INSERT_DATABASE_USER_2_HERE
          name: yellowfin-multi-instance-prod
          image: yellowfinbi/yellowfin-app-only:<RELEASE_VERSION_GOES_HERE>
          ports:
            - name: web
              containerPort: 8080
```

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. Save the text to a YAML file called **yellowfin-multiple-instances.yml**
7. Run the following command in a terminal to deploy Yellowfin:
   ```
   Kubectl apply –f yellowfin-multiple-instances.yml
   ```
8. Start Yellowfin by typing your host URLon port 8080
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

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 on Premises

Install in the Cloud

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