ibi™ WebFOCUS® - Container Edition

Installation and Deployment Guide

Version 9.3.5 | July 2025

Contents

Introduction

ibi™ WebFOCUS® - Container Edition provides a scalable, microservices-based analytical platform designed for container-based deployments. This new offering packages WebFOCUS® software components in a container image, including all required OS components and libraries, which can be deployed and run in a Kubernetes cluster.

The following are several benefits and advantages you can use in WebFOCUS® - Container Edition:

- **Automatic deployments**. Identify which (and how many) WebFOCUS® components (for example, ibi™ WebFOCUS® Reporting Server) need to be deployed and where (for example, specific clusters).
- **Reduce operational and infrastructure costs**. Add or remove services and resources to the Kubernetes cluster as needed. Data sources can exist in the cloud (private, public), or on-premises.
- **Increase adoption and usage**. Develop specific BI applications for targeted users on-premises and deploy to the cloud as needed. When you install WebFOCUS on Kubernetes, development and testing can occur within internal clusters or external clusters.
- **Portability**. Fully portable between cloud and on-premises environments.
- **Dynamic scaling**. Rapidly adjust to increased user and data demands.

Existing Kubernetes services that are provided by cloud providers can also be used (for example, Amazon® Elastic Kubernetes Service (EKS)).

Deploying WebFOCUS - Container Edition on Kubernetes provides a variety of deployment options in the private cloud, which can be centrally managed and administered based on dynamic workloads.

Docker images enable key components and functionality of the WebFOCUS platform to be isolated, packaged into specific containers and run on Kubernetes. As a result, each WebFOCUS component can be scaled independently to accommodate users, data, and data sources on demand.

Kubernetes Overview

Kubernetes is an open-source system for running, managing, and orchestrating containerized applications in a cluster of servers (known as a Kubernetes cluster). Kubernetes clusters can run in any cloud environment (for example, private, public, hybrid) or on-premises.

A Kubernetes cluster consists of a set of worker machines, called nodes that run containerized applications. Every cluster has at least one worker node.

The size of a Kubernetes cluster (determined by the number of nodes) depends on the application workloads that are running. For example, each node can represent an 8 core / 64 GB RAM system. Pods are the basic objects in a Kubernetes cluster, which consists of one or more software containers. The worker nodes host the Pods that are the components of the application workload. Usually, only one container runs in a Pod. However, multiple containers can be run in a Pod if needed (depending on specific environment requirements). If one Pod fails, Kubernetes can automatically replace that Pod with a new instance.

Key benefits of Kubernetes include automatic scalability, efficient load balancing, high availability, failover/fault tolerance, and deploying updates across your environment without any disruptions to users.

Requirements and Prerequisites

This section describes the requirements and prerequisites for deploying WebFOCUS - Container Edition in a Kubernetes cluster.

Following is the list of requirements and prerequisites for deployment.

- Docker Version 19.x upto 28.x
- Kubernetes Cluster Version 1.27.x upto 1.32.x
- Containerd Runtime
- kubectl
- PersistentVolume (PV) Storage Class Provisioner
- ReadWriteMany Access Mode
- Kubernetes Context and Kubeconfig
- Helm Version 3.5.x upto 3.16.x
- Helmfile Version 0.170.0 upto 1.1.2

Considerations for New and Existing ibi WebFOCUS Customers

Most enterprises/organizations adhere to strict and unique guidelines on a wide range of topics such as high availability, redundancy, failover. It is advised that WebFOCUS - Container Edition customers review their existing policies carefully and determine the best way to implement them when deploying WebFOCUS - Container Edition in a Kubernetes cluster.

The WebFOCUS - Container Edition installation provides infrastructure components such as the WebFOCUS Repository, search engine (Apache Solr), and so on, which should be used only for test and demo purposes (non-production environments).

ibi WebFOCUS - Container Edition Operating System and Base Images

You should utilize a Linux environment to construct your Docker images. From 1.3.0 onwards, we are shipping WebFOCUS - Container Edition images and rhel/ubi9 is served as base images of all the components.

Docker Version 19.x upto 28.x

If Docker is not installed, use the links below and follow the steps to install Docker on your machine:

https://docs.docker.com/engine/install

To verify the Docker version currently installed, enter the following command in your terminal window:

```
$> docker version
```

Kubernetes Cluster Version 1.27.x upto 1.32.x

Users can consult with their Kubernetes administrator to provision the best available Kubernetes cluster on their environment (such as HA, non-HA control-plane etc.). A Kubernetes cluster running on the Amazon® Elastic Kubernetes Service (EKS) is recommended.

- The Kubernetes cluster must use CNI that supports UDP transport.
- Kubernetes must have access to the registry where WebFOCUS - Container Edition component images are available.
- The Kubernetes cluster must support dynamic provision of volumes (pv).
- The common volume should be 30 to 40 GB.
- If you are using a back-end data source/database that is running outside (external) of the Kubernetes cluster, then ensure it is running somewhere where the cluster has access to it.
- Each node in the Kubernetes cluster must be sufficiently powerful.

  > **Note:** These specifications are recommended for production only, for development/poc purposes you can deploy WebFOCUS - Container Edition on regular laptop/desktops.

  - 32 GB RAM
  - 8 CPU 3.x Ghz
  - 50 GB Available Disk Space

> **Note:** The individual who is responsible for installing WebFOCUS - Container Edition must have admin privileges over the Kubernetes namespace where WebFOCUS - Container Edition is installed. To create a Kubernetes with restricted permissions, see Creating Kubernetes User with Limited Permissions.

To verify the Kubernetes version currently installed, enter the following command in your terminal window:

```
$> kubectl version
```

Containerd Runtime

Starting with Kubernetes version 1.25, `containerd` replacing Docker as the default runtime. With this change, Kubernetes architecture should be more streamlined and more compatible with other container runtimes, increasing its flexibility and efficiency.

```
$> kubectl get nodes -o wide
```

NAME	STATUS	ROLES	AGE	VERSION	OS-IMAGE	KERNAL-VERSION	CONTAINER-RUNTIME
mynode	Ready	control-plane, master	22h	v1.25.7+k3s1	Ubuntu 22.04 LTS	5.15.0-27-generic	containerd://1.6.15-k3s1

> **Note:** If your images are stored in the registry, then no need to run the `docker save` command. It is only required when you are running on a single node cluster, building and using images locally. For the multi-node cluster, your images are stored in some Docker registry.

You must use the docker save command after building WebFOCUS - Container Edition images from the build_images.sh script.

The following are the commands for available images on containered.

```
$> docker save ibi2020/webfocus:wfc-9.3.5 | sudo  ctr -n k8s.io images
import -

$> docker save ibi2020/webfocus:cm-9.3.5 | sudo  ctr -n k8s.io images
import -

$> docker save ibi2020/webfocus:wfs-etc-9.3.5 | sudo  ctr -n k8s.io
images import -

$> docker save ibi2020/webfocus:wfs-9.3.5 | sudo  ctr -n k8s.io images
import -
```

kubectl

The Kubernetes command-line tool, kubectl, allows you to run commands against Kubernetes clusters. You can use kubectl to deploy applications, inspect and manage cluster resources, and view logs.

If kubectl is not installed, use the following link to download and install kubectl (Linux, macOS):

https://kubernetes.io/docs/tasks/tools

PersistentVolume (PV) Storage Class Provisioner

PersistentVolume (PV) storage class for cloud environments such as Amazon EKS (Elastic Kubernetes Service) provides a broad and reliable solution for applications running on Kubernetes clusters. This storage class is designed to integrate with cloud services such as Amazon EFS (Elastic File System) and NFS (Network File System), providing convenient options for storing and accessing data.

Amazon EFS provides a storage management system that can be configured as a normal volume in a Kubernetes cluster. It offers scalable, elastic storage that can grow and shrink as needed, making it suitable for many applications.

NFS (Network File System) is a file sharing system often used to share files between servers. In Kubernetes, NFS can be used as a backup storage option for PersistentVolumes, allowing applications to access shared volumes on multiple nodes in a cluster.

Leveraging cloud-based storage solutions of the PV storage class, Kubernetes applications can benefit from high availability, durability, and scalability when managing the network. Adapt to the condition of the container.

On Google Cloud Platform (GCP), the equivalent service to Amazon EFS is Google Cloud Filestore. Google Cloud Filestore makes all data storage accessible via the NFSv3 protocol. It offers scalable performance and capacity, making it suitable for a variety of workloads built on Kubernetes clusters.

The equivalent service to Amazon EFS for Azure is Azure Files. Azure Files is an integrated file management system in Azure that provides the ability to create shared files that can be installed from Azure as a network drive or as a traditional service. Supports SMB (Server Message Block) protocol for Windows-based applications and NFS (Network File System) protocol for Linux-based applications; this makes it a versatile option for Kubernetes clusters deployed in Azure.

Ensure that a storage provisioner is available in the Kubernetes cluster. To verify that a storage class exists, enter the following command in your terminal window:

```
$> kubectl get sc
```

Note: If you are trying on non production setup and you don't have any storage class, follow the below steps to create for your local environment.

If a default storage class is not available, and if you want to deploy on non-production setup, then use the link below and follow the steps to enable local storage for your environment:

https://github.com/rancher/local-path-provisioner

To enable the default storage class, enter the following command in your terminal window:

```
$> kubectl apply -f https://raw.githubusercontent.com/rancher/local-
path-provisioner/v0.0.26/deploy/local-path-storage.yaml
```

Note: For the latest version, see the https://github.com/rancher/local-path-provisioner.

You can enter the following command in your terminal window to make local-path-storage (referenced in the above link) a default storage class:

```
$> kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

During testing, the Amazon EFS Container Storage Interface (CSI) driver and NFS storage class were used. To deploy with different storage class (like efs or nfs) , you must modify the following components in your configuration:

- **storage.storageClass**: Update this value with the correct storage class being used.
- **storage.accessMode**: Must be set to ReadWriteMany.

For more information, refer the following link:

Amazon EFS CSI driver: https://docs.aws.amazon.com/eks/latest/userguide/efs-csi.html

NFS CSI driver: https://github.com/kubernetes-csi/csi-driver-nfs?tab=readme-ov-file#readme

If you want to deploy the WebFOCUS - Container Edition with pre-created PVC, see Deploying ibi WebFOCUS - Container Edition with Existing PVC

ReadWriteMany Access Mode

For deployment scaling and multi-node Kubernetes clusters, ensure your PersistentVolume (PV) is set to the ReadWriteMany access mode.

Kubernetes Context and Kubeconfig

Ensure that your Kubernetes context is set. To display a list of contexts currently being used, enter the following command in your terminal window:

```
$> kubectl config get-contexts
```

If required, enter the following command in your terminal window to set the context:

```
$> kubectl config use-context <my-cluster-name>
```

Note:

- You can pass context directly with a `helmfile sync/destroy` command using a `--kube-context` parameter, it sets `kubectl` context otherwise it uses the current context by default.
- If you are using `webfocusce` tool, then pass the `--context` parameter.

Helm Version 3.5.x upto 3.16.x

Helm is the “package manager” for Kubernetes, which is used to install charts (“packages”) into Kubernetes. You can consider charts as packaged applications. Charts are your versioned, pre-configured application resources, which can be deployed as one unit. A Helm chart (.tgz file) contains templates that define Kubernetes resources to install.

Install helm based on your Operating System from the given link https://helm.sh/docs/intro/install.

To verify the Helm version is installed, enter the following command in your terminal window:

```
$> helm version
```

Helmfile Version 0.170.0 upto 1.1.2

Helmfile adds functionality to Helm by wrapping it in a declarative specification that allows you to compose several charts together to create a comprehensive deployment artifact, from a single application to your entire infrastructure stack.

For more information, reference the following link:

https://github.com/roboll/helmfile#installation

To download the Helmfile, reference the following link:

https://github.com/roboll/helmfile/releases

- macOS (using homebrew): `brew install helmfile`

To verify the Helmfile version currently installed, enter the following command in your terminal window:

```
$> helmfile version
```

Note: If you're using Helmfile and your version is less than 0.170.0, then the command syntax requires the -f flag (or --file) when specifying a file path to your helmfile.yaml.gotmpl.
For example:

```
helmfile -f path/to/your/helmfile.yaml.gotmpl sync
```

Component Version Reference

The following table lists versions of all the ancillary components that were tested with WebFOCUS - Container Edition for Release 9.3.5.

Component	Version
Kubernetes NGINX Ingress controller	0.40.2 (Helm chart: 3.4.1)
Swego	1.16.0
Octant	0.13.1
PostgreSQL	11.12.0 (Helm chart: 10.4.5)
Apache Solr	9.8.1 (Helm chart: 9.6.5)
Etcd	3.4.14 (Helm chart: 5.3.0)
Node.js	18.x
Prometheus Server (prometheus)	2.43.0 (Helm chart: 15.0.0)
Prometheus Adapter (prometheus-adapter)	0.10.0 (Helm chart: 3.0.0)
ibi™ WebFOCUS® Client (appserver)	9.3.5
WebFOCUS® Reporting Server (edaserver)	9.3.5
Cluster Manager (clm)	9.3.5
Cache Manager (cachemanager)	9.3.5

The images that are shipped or build by you with WebFOCUS - Container Edition consists of third-party libraries. Below is the list of libraries along with the information in which the components are used.

Library	Version	Component
Java	11	Appserver and Edaserver
Node.js	18	Cachemanager
Jetty	9.4.57	Appserver and Reportcaster

Key Components/Systems

Before deploying WebFOCUS - Container Edition, ensure that the following systems are available in your environment with the corresponding components installed accordingly.

Note:

- If required, you can also run everything from a single machine (from building the Docker images to deploying WebFOCUS - Container Edition in a Kubernetes cluster).
- If you are running on low resources and in a cloud environment, you can remove the `environments/resource-limit.yaml.gotmpl` line in the cloud environment, which is located in the `/scripts/helmfile/helmfile.yaml` file. You can also update the default values for the allotted resources as required. For more information, see Resource (CPU and Memory) Parameters.

1. **Build Server**: After obtaining the WebFOCUS - Container Edition `.tar` file, this machine is used to build the required Docker images for WebFOCUS - Container Edition on the image registry.

   Recommended Specifications:

   - 16 GB RAM
   - 4 CPU 3.x Ghz
   - 50 GB Available Disk Space

   Installed Software

   - Docker

2. **Deployment Server**: Using the Helm charts that are provided with each Docker image (as is or modified as per end-user requirements), this machine is used to deploy WebFOCUS - Container Edition components on the Kubernetes cluster.

   Recommended Specifications:

   - 16 GB RAM

- 4 CPU 3.x Ghz
- 20 GB Available Disk Space

Installed Software

- Helm and Helmfile
- Kubectl

3. **Kubernetes Cluster**: Ensure that your Kubernetes cluster is ready/available and each node in your cluster where WebFOCUS - Container Edition is being deployed supports the following recommended specifications:
    - 32 GB RAM
    - 8 CPU 3.x Ghz
    - 50 GB Available Disk Space

Creating or Using Docker Images

Pre-built WebFOCUS - Container Edition images are included in the download package. You can download ready-to-use images `.tar` file from e-delivery. It helps provide a consistent environment across development, testing, and production environments. You can also create your own images using the `build-images.sh` script.

Following are the ways to build docker images:

- Downloading Shipped Images to Load in Docker
- Creating your own Images for Deployment

Downloading Shipped Images to Load in Docker

With the use of pre-built container images, deployment becomes a straightforward process, eliminating the need to build and configure images from scratch, saving significant time and effort.

Procedure

1. Download the `IBI_wfce_images_9.3.5.tar` image file.

   Note: The minimum disk space requirement for shipped images is 40 GB.

2. Load images to Docker using the following command:

```
$> docker load -i IBI_wfce_images_9.3.5.tar

$> docker images
```

   Note: Images that are shipped have `label maintainer = ibi` but images that are build from `./build_image.sh` script does not have that label.

Result

You can see 4 images loaded to your docker.

Creating your own Images for Deployment

You can create all Docker images for WebFOCUS - Container Edition using the following command:

```
$> cd <webfocus-ce>/scripts
$> ./build-images.sh
```

It creates 4 images to your docker.

Docker Images Used by ibi WebFOCUS - Container Edition

The following table provides a reference for the Docker images that are used (pulled) by WebFOCUS - Container Edition components.

Service Pod Name	Base Image	Docker Image
appserver	redhat/ubi9	wfc-9.3.5
cachemanager	redhat/ubi9-minimal	cm-9.3.5
clm	redhat/ubi9	wfs-9.3.5
edaserver	redhat/ubi9	wfs-9.3.5
reportcaster	redhat/ubi9	wfc-9.3.5

The following table provides a reference for the Docker images that are used (pulled) by third-party components.

Service Pod Name	Docker Image
etcd	3.5.12-debian-12-r7
postgresql	redhat/ubi9

Service Pod Name	Docker Image
	postgresql:11.12.0-debian-10-r1
prometheus-server	k8s.gcr.io/prometheus-adapter/prometheus-adapter:v0.10.0 quay.io/prometheus/prometheus:v2.43.0 jimmidyson/configmap-reload:v0.5.0
prometheus-adapter	k8s.gcr.io/prometheus-adapter/prometheus-adapter:v0.10.0
solr	bitnami/solr:9.8.1-debian-12-r9
zookeeper	bitnami/zookeeper:3.9.3-debian-12-r15
swego	ishswar/swego:1.2.3 ployst/nginx-ssl-proxy xmartlabs/htpasswd
metrics-server	metrics-server:v0.5.2

Building and Loading ibi WebFOCUS - Container Edition Images using Podman

Note: You can either build the WebFOCUS Container edition images using Podman or download the IBI_wfce_images_9.3.5.tar file and load it with Podman, as outlined in the steps below.

Procedure

1. To build WebFOCUS - Container Edition images using Podman, navigate to the `/scripts` directory, set the environment variable TOOL=podman before building image and run the `./build-images.sh` command.

```
$> cd /IBI_wfce_9.3.5/scripts
```

```
$> export TOOL=podman
$> ./build-images.sh
```

Following is the sample output:

```
export TOOL=podman
~/IBI_wfce_9.3.5/scripts$ ./build-images.sh
SCRIPTS_DIR = /home/ibi/IBI_wfce_9.3.5/scripts
Variables exported:
PLATFORM_NAME=webfocus
INFRA_NS=webfocus
WF_TAG=wfc-9.3.5
RS_TAG=wfs-9.3.5
ETC_TAG=wfs-etc-9.3.5
CM_TAG=cm-9.3.5
podman daemon is running.
***********************************************
Building Reporting server Docker image
***********************************************
...
```

Note: If you cannot set the TOOL then it defaults to Docker, and images builds using Docker.

Following is the sample output:

```
~/IBI_wfce_9.3.5/scripts$ ./build-images.sh
SCRIPTS_DIR = /home/ibi/IBI_wfce_9.3.5/scripts
Variables exported:
PLATFORM_NAME=webfocus
INFRA_NS=webfocus
WF_TAG=wfc-9.3.5
RS_TAG=wfs-9.3.5
ETC_TAG=wfs-etc-9.3.5
CM_TAG=cm-9.3.5
TOOL is not set. Defaulting to Docker.
```

2. To load WebFOCUS - Container Edition images using Podman, run the `podman load` command.

    a. Download the `IBI_wfce_images_9.3.5.tar` image file.

b. You can load all WebFOCUS - Container Edition images using the following command:

```
podman load -i IBI_wfce_images_9.3.5.tar
```

3. Verify images after building or loading images using the following command:

```
podman images
```

Pushing and Pulling Docker Images from Docker Registry

Push Docker Images in Docker Registry

You need to push build images to your registry. Following are the steps to push docker images in docker registry.

Procedure

1. Login to the docker account using the following command:

   ```
   $> docker login -u <docker_username>
   ```

2. After you logged in, tag the docker images and push it to the registry, use the following commands.

   Tagging and pushing WebFOCUS Client image to docker hub.

   ```
   $> docker tag ibi2020/webfocus:wfc-9.3.5 <docker_username>/<docker_
   repo_name>:wfc-9.3.5

   $> docker push <docker_username>/<docker_repo_name>:wfc-9.3.5
   ```

   Tagging and pushing WebFOCUS Reporting Server image to docker hub.

   ```
   $> docker tag ibi2020/webfocus:wfs-9.3.5 <docker_username>/<docker_
   repo_name>:wfs-9.3.5

   $> docker push <docker_username>/<docker_repo_name>:wfs-9.3.5
   ```

   Tagging and pushing Cachemanager image to docker hub.

```
$> docker tag ibi2020/webfocus:cm-9.3.5 <docker_username>/<docker_
repo_name>:cm-9.3.5

$> docker push <docker_username>/<docker_repo_name>:cm-9.3.5
```

Tagging and pushing WebFOCUS Reporting Server etc image to docker hub.

```
$> docker tag ibi2020/webfocus:wfs-etc-9.3.5 <docker_
username>/<docker_repo_name>:wfs-etc-9.3.5

$> docker push <docker_username>/<docker_repo_name>:wfs-etc-9.3.5
```

Replace `<docker_username>` with your username and `<docker_repo_name>` with your repository name.

Pulling an Image from Private Docker Registry

You need to pull build images from your registry. Following are the steps to pull docker images from the private docker registry.

Procedure

1. When you pull the image from the private docker registry, then you need to provide the valid `dockerconfigjson` in `global.dockerconfigjson`.
2. Create `dockerconfigjson` for pulling image during WebFOCUS - Container Edition deployment, use the following command:

```
$> kubectl create secret docker-registry --dry-run=true ibi-docker-
hub --docker-server=https://index.docker.io/v1/ --docker-
username=<docker_username> --docker-password=dckr_pat_dummytest --
docker-email=<email-address> --namespace=<namespace> -o yaml
```

3. Edit the `<webfocus_ce>/scripts/helmfile/environments/wf.integ.yaml.gotmpl` and replace the existing value with your generated secret in `dockerconfigjson`.

> **Note:** While WebFOCUS - Container Edition deployment, it creates `ibi-docker-hub` secret on the base in `dockerconfigjson` and uses for pulling the image from private docker registry.

Image Customization Using Docker Scripts

Image customization using Docker scripts allows customers to avoid manually placing library files inside containers. Instead, WebFOCUS - Container Edition provides a default lib location where users can place their JARs or driver file, which then is picked up automatically by the Docker build scripts.

1. Image customization before building images:
    a. Copy any file, for example drivers or jar file inside the <webfocus-ce>/wfs/iserver/lib folder.
    b. Build a reporting server and other required docker images and deploy WebFOCUS - Container Edition.
    For more information, see Creating or Using Docker Images.
    c. Access the "edaserver-0" POD and check the location /opt/ibi/srv/lib. You see your required jar.
2. Image customization using pre-build docker images.
    a. Copy any file, for example drivers or jar file inside <webfocus-ce>/samples/lib folder.
    b. Re-build reporting server image using steps mentioned in the readme.txt located in the <webfocus-ce>/samples folder.
    c. Access the "edaserver-0" POD and check the location /opt/ibi/srv/lib. You see your required jar.

Sample Output:

```
$ kubectl exec -it edaserver-0 -n webfocus -- /bin/sh
$ cd /opt/ibi/srv/lib
$ ls
mssql-jdbc-7.2.2.jre11.jar  ojdbc11-23.3.0.23.09.jar  readme.txt
```

Deploying ibi WebFOCUS - Container Edition

Following are the list of WebFOCUS - Container Edition deployment.

- Deploying ibi WebFOCUS - Container Edition on Kubernetes Cluster
- Deploying ibi WebFOCUS - Container Edition Using Docker Compose
- Deploying ibi WebFOCUS - Container Edition with Podman and Podman Compose On RHEL 9
- Deploying ibi WebFOCUS - Container Edition with Red Hat OpenShift Cluster
- Deploying ibi WebFOCUS - Container Edition Appserver with MS-SQL server or Oracle as a Repository Database
- Deploying ibi WebFOCUS - Container Edition using Docker Compose with External MS-SQL or Oracle Database
- Deploying ibi WebFOCUS - Container Edition on Cloud Environment
- Deploying ibi WebFOCUS - Container Edition with External Solr Server
- Deploying ibi WebFOCUS - Container Edition for Prometheus in Different Namespace and Same Cluster
- Deploying ibi WebFOCUS - Container Edition with Existing PVC

Deploying ibi WebFOCUS - Container Edition on Kubernetes Cluster

WebFOCUS - Container Edition is deployed in two phases:

- **Infra Components:** It includes Apache solr, postgresql, prometheus, and prometheus-adapter. All these components are by default installed with infra deployment but you can skip the installation of any component based on your requirements.

  > **Note:** Infra components are third-party components and not maintained by us. It is recommended to use your own managed infra components for production deployment.

- **WebFOCUS Components:** It includes clm, etcd, appserver, edaserver, cachemanager, reportcaster. All these components are core components for the deployment of WebFOCUS - Container Edition.

Use the following steps to deploy WebFOCUS - Container Edition using Helm charts.

> **Note:**
> - It is mandatory to set `global.ACCEPT_EUA=Y` and `global.WF_CUSTOMERID=<customerid>` in `<webfocusce>/scripts/helmfile/environments/wf.integ.yaml.gotmpl` before proceeding to WebFOCUS - Container Edition deployment. If you do not know your customer ID, then visit our product Support website.
> - You have to replace the active `license.txt` file to the following location `<webfocus-ce>/scripts/license` or update the file path of `global.WF_LICENSE` with your license file location. If you do not know about your license file, then visit our product Support website.

Procedure

1. Navigate to `<webfocus-ce>/scripts/helmfile` folder, and export the required

environment variable using export-default shell scripts, use the following command:

```
$> cd <webfocus-ce>/scripts/helmfile/

$> source export-defaults.sh
```

It exports all necessary variables to run WebFOCUS - Container Edition. You can overwrite the default values.

> **Note:** If you want to deploy WebFOCUS - Container Edition on different namespace, then change the `PLATFORM_NAME` name and export the variables. For more information, see Export Environment Variable Parameters.

2. To deploy infra components, navigate to `<webfocus-ce>/scripts/helmfile/infra` folder and use the following command:

```
$> cd <webfocus-ce>/scripts/helmfile/infra

$> helmfile sync
```

3. To deploy WebFOCUS - Container Edition components, navigate to `<webfocus-ce>/scripts/helmfile` folder, while running the helmfile command pass environment variable (`dev` or `cloud`), use the following command:

```
$> cd <webfocus-ce>/scripts/helmfile/

$> helmfile -e <env-name> sync
```

For more information about `dev` and `cloud` environments, see Helmfile Environments Reference.

> **Note:** If you are deploying on RHEL 9.x or Rocky Linux 9.x, then set edaserver.ULIMIT_NPROC=200000 and clm.ULIMIT_NPROC=200000 in `<webfocusce>/scripts/helmfile/environments/wf.integ.yaml.gotmpl` file. For more information, see Troubleshooting.

It deploys WebFOCUS - Container Edition with default configurations. helmfile `sync`

also supports other configuration parameters `--state-values-file`, and `--state-values-set` etc. For more information to set different parameters apart from the default value, see Appendix A: Configuration Parameters Reference.

> **Note:** If you are deploying WebFOCUS - Container Edition using the `dev` environment, which open ports for Cluster Manager (CLM) (Port: 31121) and WebFOCUS Reporting Server (Port: 31131).

4. Optionally, to list all deployed releases of WebFOCUS - Container Edition, enter the following command in your terminal window/console:

```
$> helm list -n <namespace>
```

5. Access WebFOCUS using the following URL:

```
http://<hostname>:31080/webfocus
```

The default login credentials are:

- Username: secret
- Password: terces

When deployed using the `-e dev` environment option, the URLs for CLM and WebFOCUS Reporting Server URLs are available as follows:

- CLM: `http://<hostname>:31121`
- WebFOCUS Reporting Server: `http://<hostname>:31131`

Deploying ibi WebFOCUS - Container Edition Using Docker Compose

To deploy WebFOCUS - Container Edition using Docker Compose, perform the following steps.

Note: It is recommended only for non production deployment.

Note: If you are deploying on RHEL 9.x or Rocky Linux 9.x, then set ULIMIT_NPROC=200000 in <webfocus-ce>/scripts/docker-compose/.env file. For more information, see Troubleshooting.

Procedure

1. Download the WebFOCUS - Container Edition tar file and extract it.
2. Update the docker compose environment file, navigate to `<webfocus-ce>/scripts/docker-compose`.

   ```
   $> cd <webfocus-ce>/scripts/docker-compose
   ```

   Edit the `.env` file and add the Customer ID in `WF_CUSTOMERID` parameter.

   ```
   WF_CUSTOMERID= <customerid>
   ```

3. Build the Docker images, using the following commands.

   ```
   $> cd <webfocus-ce>/scripts/docker-compose

   $> docker compose build
   ```

4. Deploy it on the server only, using the following commands.

   ```
   $> cd <webfocus-ce>/scripts/docker-compose
   ```

```
$> docker compose --profile server up
```

> **Note:** If you want to pull images from the registry, and skip image build then run with `--no-build` argument, for example, `docker compose --profile server up --no-build`.

5. Deploy it on the server and client using the following command:

```
$> docker compose up
```

6. Check the running container using the following command:

```
$> docker compose ps
```

7. Run the smoke test using the following command:

```
$> docker compose --profile smoketest up
```

8. Undeploy the cluster using the following command:

```
$> docker compose down
```

Deploying ibi WebFOCUS - Container Edition with Podman and Podman Compose On RHEL 9

Podman is a lightweight, daemonless container engine compatible with Docker, offering a flexible and secure environment for running containers, including rootless operation. Podman Compose, a tool for defining and running multi-container Podman applications, simplifies the orchestration of ibi™ WebFOCUS® - Container Edition components. This deployment method is useful for non-production environments, local development, and streamlined container orchestration.

Note: It is recommended only for nonproduction deployment.

Installing Podman on RHEL 9

This section covers the installation and initial testing of Podman on RHEL 9.

Step 1: Update the System

Ensure that your system is up to date before proceeding.

```
sudo dnf update -y
```

Step 2: Install Podman

Install the Podman package using DNF.

```
sudo dnf install -y podman
```

Step 3: Verify Installation.

Verify that Podman is installed correctly by checking its version.

```
podman --version
```

Optional: Enable and Start the Podman Service

For socket activation or rootless containers, enable and start the Podman service.

```
systemctl --user enable --now podman.socket
```

Note: This step requires a non-root user and lingering to be enabled.

```
loginctl enable-linger $USER
```

Step 4: Test Podman

Run a test container to verify that Podman is functioning correctly.

```
podman run --rm -it alpine sh
```

Troubleshooting Podman Installation

This section addresses common errors encountered during Podman installation and testing.

Error: Copying System Image

If you encounter the following error:

```
Error: copying system image from manifest list: writing blob: adding
layer with blob
"sha256:fe07684b16b82247c3539ed86a65ff37a76138ec25d380bd80c869a1a4c73236
"/"/"/"sha256:fd2758d7a50e2b78d275ee7d1c218489f2439084449d895fa17eede6c6
1ab2c4": unpacking failed (error: exit status 1; output: potentially
insufficient UIDs or GIDs available in user namespace (requested 0:42
for /etc/shadow): Check /etc/subuid and /etc/subgid if configured
locally and run "podman system migrate": lchown /etc/shadow: invalid
argument)
```

This indicates an issue with UID/GID mapping. Follow the steps below to resolve it.

Configure Subuid and Subgid Properly

1. Check your username:

   ```
   whoami
   ```

   Let us assume your username is "John".

2. Edit /etc/subuid and /etc/subgid:

   Ensure that the following entries exist.

```
john:100000:65536
```

Add these entries if they are missing.

```
sudo sh -c 'echo "john:100000:65536" >> /etc/subuid'
sudo sh -c 'echo "john:100000:65536" >> /etc/subgid'
```

Note: Note: Ensure that there are no duplicate entries or overlapping ranges.

3. Run Podman System Migration:

   After fixing the UID/GID ranges, run the following command.

   ```
   podman system migrate
   ```

   This updates your user storage and configuration based on the new mappings.

Installing Podman Compose

This section guides you through the installation of Podman Compose on RHEL 9.

Steps to Enable EPEL on RHEL 9

Podman Compose is available in the Extra Packages for Enterprise Linux (EPEL) repository.

Install EPEL Repo Manually

Download and install the epel-release RPM.

```
sudo dnf install -y https://dl.fedoraproject.org/pub/epel/epel-release-
latest-9.noarch.rpm
```

Enable EPEL and Refresh the Cache

Enable the EPEL repository and refresh the DNF cache.

```
sudo dnf config-manager --enable epel
sudo dnf makecache
```

Install Podman Compose

Install Podman Compose using DNF.

```
sudo dnf install -y podman-compose
```

Deploying ibi WebFOCUS - Container Edition Using Podman Compose

To deploy WebFOCUS - Container Edition using podman compose, perform the following steps.

1. Download the WebFOCUS - Container Edition tar file and extract it.
2. Update the docker compose environment file, navigate to <webfocusce>/scripts/docker-compose.

   ```
   $> cd <webfocus-ce>/scripts/docker-compose
   ```

3. Edit the .env file and add the Customer ID in `WF_CUSTOMERID` parameter.

   ```
   WF_CUSTOMERID= <customerid>
   ```

4. Build WebFOCUS - Container Edition images, refer Creating or Using Docker Imagesor Building and Loading ibi WebFOCUS - Container Edition Images using Podman.
5. Deploy it on the server and client using the following command:

   ```
   podman compose -f podman-compose.yaml up -d
   ```

6. Undeploy the cluster using the following command:

   ```
   podman compose -f podman-compose.yaml down
   ```

Deploying ibi WebFOCUS - Container Edition with Red Hat OpenShift Cluster

Red Hat OpenShift, a hybrid cloud application platform powered by Kubernetes, brings together tested and trusted services to reduce the friction of developing, modernizing, deploying, running, and managing applications. Red Hat OpenShift delivers a consistent experience across public cloud, on-premises, hybrid cloud, and edge architecture.

Prerequisites

- Helm
- Helmfile
- Docker/Podman

System Requirements

- Memory: At least 16 GB of RAM.
- CPU: At least 8 CPU cores.
- Disk space: Approximately 80 GB of free disk space.

Supported Platforms

A Linux distribution (RHEL, Fedora, Ubuntu, CentOS, or similar) with a 64-bit architecture.

> ⚠ **Caution:** Code snippets in the PDF format might have undesired line breaks due to space constraints. You must verify code snippets before directly copying and pasting them in your program.

Setting up CRC

Procedure

1. Select the operating system from the dropdown list and download the Red Hat OpenShift local installer from the following link:

https://console.redhat.com/openshift/create/local

2. Extract the downloaded CRC tar file to your specified location and add it to the path using the following command:

```
$ tar xvf crc-linux-amd64.tar.xz
crc-linux-2.16.0-amd64/
crc-linux-2.16.0-amd64/LICENSE
crc-linux-2.16.0-amd64/crc

$ mkdir -p ~/local/bin

$ mv crc-linux-*-amd64/crc ~/local/bin/

$ export PATH=$HOME/local/bin:$PATH

$ crc version
CRC version: 2.16.0+05b62a75
OpenShift version: 4.12.9
Podman version: 4.4.1

$ echo 'export PATH=$HOME/local/bin:$PATH' >> ~/.bashrc
```

3. Navigate to the extracted folder and set up the Red Hat OpenShift cluster using the following command:

```
$> crc setup
```

4. Check the default configuration of CPU, memory, and disk space using the following command:

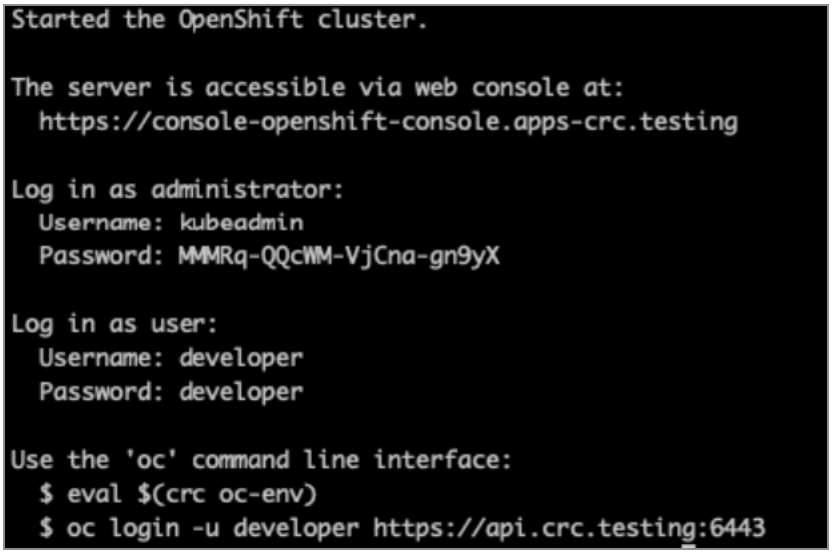
```
$> crc config view
```

Update the Red Hat OpenShift cluster default configuration to the recommended configuration for CPU, memory, and disk space using the following command:

```
$> crc config set memory 16000
$> crc config set cpus 6
$> crc config set disk-size 80
```

5. You need a pull secret to start the CRC, which you can download or copy from https://console.redhat.com/openshift/create/local, and run the following command:

```
$> crc start -p <path-to-your-pull-secret-file>
```



Internal Registry for Getting Images

Use the following steps to deploy the WebFOCUS - Container Edition using internal registry images.

Procedure

1. Push the images using Docker or Podman:

    a. Update the Docker configuration to allow push images to the unsecure Red Hat OpenShift local registry using the following command:

    ```
    $> vi ~/etc/docker/daemon.json
    ```

    After opening the `daemon.json`, add the following entry:

    ```
    {
        "insecure-registries" : [ "default-route-openshift-image-
    registry.apps-crc.testing:443" ]
    }
    ```

Now, restart the docker using the following command:

```
$> sudo systemctl daemon-reload
$> sudo systemctl restart docker
```

b. To push the images using Podman, use the following command:

```
vi /etc/containers/registries.conf

[registries.insecure]
registries = ['default-route-openshift-image-registry.apps-
crc.testing:443']
```

2. Log in to the Red Hat OpenShift cluster as a developer user using the following command:

```
$> eval $(crc oc-env)
$> oc login -u developer https://api.crc.testing:6443
```

3. Create the new Red Hat OpenShift project using the following command:

```
$> oc new-project webfocus
```

4. Download the `IBI_wfce_images_9.3.5.tar` and `IBI_wfce_9.3.5.tar` from Tibco eDelivery.

   Load the Docker images using the following command:

```
$> docker load -i IBI_wfce_images_9.3.5.tar
```

   Verify the images using the following command:

```
$> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ibi2020/webfocus	wfc-9.3.5	349845691ab7	5 days ago	1.76GB

ibi2020/webfocus	cm-9.3.5	c724570f7420	5 days ago	259MB
ibi2020/webfocus	wfs-etc-9.3.5	aae38cc4ac10	5 days ago	3.06GB
ibi2020/webfocus	wfs-9.3.5	200787c8d252	5 days ago	952MB

5. Tag the images and push it to the CRC internal registry using the following command:

```
$> docker login -u `oc whoami` -p `oc whoami --show-token` default-
route-openshift-image-registry.apps-crc.testing:443

$> docker image tag ibi2020/webfocus:wfs-9.3.5 default-route-
openshift-image-registry.apps-crc.testing:443/webfocus/repo:wfs-
9.3.5

$> docker image push default-route-openshift-image-registry.apps-
crc.testing:443/webfocus/repo:wfs-9.3.5

$> docker image tag ibi2020/webfocus:wfs-etc-9.3.5  default-route-
openshift-image-registry.apps-crc.testing:443/webfocus/repo:wfs-
etc-9.3.5

$> docker image push default-route-openshift-image-registry.apps-
crc.testing:443/webfocus/repo:wfs-etc-9.3.5

$> docker image tag ibi2020/webfocus:wfc-9.3.5  default-route-
openshift-image-registry.apps-crc.testing:443/webfocus/repo:wfc-
9.3.5

$> docker image push default-route-openshift-image-registry.apps-
crc.testing:443/webfocus/repo:wfc-9.3.5

$> docker image tag ibi2020/webfocus:cm-9.3.5   default-route-
openshift-image-registry.apps-crc.testing:443/webfocus/repo:cm-
9.3.5

$> docker image push default-route-openshift-image-registry.apps-
crc.testing:443/webfocus/repo:cm-9.3.5
```

Verify the completion using the following command:

```
$> oc get imagestreams repo -n webfocus  -o jsonpath='{range
```

```
.status.tags[*]}{.tag}{"\n"}'
```

6. Extract the downloaded `IBI_wfce_9.3.5.tar` file to get the directory of all the scripts using the following command:

```
$> tar -xvf IBI_wfce_9.3.5.tar
```

7. Deploy `infra` components using helmfile or WebFOCUS - Container Edition Health Check Tool:

   a. Deploy the `infra` components using the helmfile command:

```
$> cd IBI_wfce_9.3.5/scripts/helmfile/infra

$> source ../export-defaults.sh

$> helmfile --state-values-file=<webfocus-
ce>/helmfile/environments/oc-default.yaml.gotmpl sync
```

   OR

   b. Deploy the `infra` components using WebFOCUS - Container Edition Health Check Tool:

```
$> cd IBI_wfce_9.3.5/scripts

$> source helmfile/export-defaults.sh

$> ./webfocusce run sync --only-run infra  --state-values-file
<webfocus-ce>/helmfile/environments/oc-default.yaml.gotmpl --
ignore-healthcheck-errors
```

   Verify that the status of `infra` pods are running using the following command:

```
$> oc get pods -n webfocus
```

8. Edit `scripts/helmfile/environments/dev-wf.integ.yaml.gotmpl` and remove or comment all the instances of `imagePullSecret`.

```
imageInfo:
```

```
  imagePullSecrets:
  - name: ibi-docker-hub
```

The following is the reference file.

```
appserver:
#  imagePullSecrets:
#  - name: ibi-docker-hub
  autoscaling:
    enabled: true
#edaetc:
#  imagePullSecrets:
#  - name: ibi-docker-hub
edaserver:
#  imagePullSecrets:
#  - name: ibi-docker-hub
  service:
    type: NodePort
    #sessionAffinity: ClientIP
  autoscaling:
    enabled: true
clm:
#  imagePullSecrets:
#  - name: ibi-docker-hub
  service:
    annotations: {}
    type: NodePort
  autoscaling:
    enabled: false
cachemanager:
#  imagePullSecrets:
#  - name: ibi-docker-hub
  service:
    type: NodePort
  autoscaling:
    enabled: true
#etcd:
#  image:
#    pullSecrets:
#      name: ibi-docker-hub
swego:
  enabled: true
storage:
```

```
  accessMode: ReadWriteOnce

global:
#  imageInfo:
#    imagePullSecrets:
#    - name: ibi-docker-hub
  prometheusAdapter:
    metricsServer: false
  prometheus:
    server:
      service:
        type: NodePort
```

9. Set the `global.ACCEPT_EUA=Y` and `global.WF_CUSTOMERID=<customerid>` in `<webfocusce>/scripts/helmfile/environments/wf.integ.yaml.gotmpl` and replace the active `license.txt` file to the following location `<webfocus-ce>/scripts/license` or update the file path of `global.WF_LICENSE` with your license file location. If you do not know about your license file, then visit our product Support website.
10. Deploy the WebFOCUS - Container Edition using helmfile or WebFOCUS - Container Edition Health Check Tool:
    a. Navigate to `scripts/helmfile` and deploy the WebFOCUS - Container Edition using the following helmfile command:

    ```
    $> helmfile -e dev --state-values-file=environments/oc-
    default.yaml.gotmpl --state-values-set
    global.imageInfo.image.repository=default-route-openshift-
    image-registry.apps-crc.testing/webfocus/repo sync
    ```

    OR

    b. Navigate to `scripts` and deploy the WebFOCUS - Container Edition using WebFOCUS - Container Edition Health Check Tool:

    ```
    $> ./webfocusce run sync -e dev --only-run webfocus --state-
    values-file <webfocus-ce>/helmfile/environments/oc-
    default.yaml.gotmpl --state-values-set
    global.imageInfo.image.repository=default-route-openshift-
    image-registry.apps-crc.testing/webfocus/repo --ignore-
    healthcheck-errors
    ```

Verify the pod status using the following command:

```
$> oc get pods -n webfocus
```

11. Create a route to access the appserver using the following command:

    ```
    $> oc expose svc appserver
    ```

    View the route using the following command:

    ```
    $> oc get route -n webfocus
    ```

    For more information on the route, see Route Configuration.

External Registry for Getting Images

Use the following steps to deploy the WebFOCUS - Container Edition using external registry images.

Procedure

1. Log in to the Red Hat OpenShift cluster as a developer user using the following command:

   ```
   $> eval $(crc oc-env)
   $> oc login -u developer https://api.crc.testing:6443
   ```

2. Create the new Red Hat OpenShift project using the following command:

   ```
   $> oc new-project webfocus
   ```

3. Download the `IBI_wfce_images_9.3.5.tar` and `IBI_wfce_9.3.5.tar` from Tibco eDelivery.

   Load the Docker images using the following command:

   ```
   $> docker load -i IBI_wfce_images_9.3.5.tar
   ```

   Verify the images using the following command:

```
$> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ibi2020/webfocus	wfc-9.3.5	349845691ab7	5 days ago	1.76GB
ibi2020/webfocus	cm-9.3.5	c724570f7420	5 days ago	259MB
ibi2020/webfocus	wfs-etc-9.3.5	aae38cc4ac10	5 days ago	3.06GB
ibi2020/webfocus	wfs-9.3.5	200787c8d252	5 days ago	952MB

4. Tag the images and push it to the external registry. For more information, see Pushing and Pulling Docker Images from Docker Registry.
5. Updating the repository path and tags in the `local.yaml.gotmpl` file. A sample file (local.yaml.gotmpl) is provided in the environments directory:

   ```
   <webfocus-ce>/scripts/helmfile/environments/local.yaml.gotmpl
   ```

   Note: You can overwrite this sample file or create a new copy and update the file based on your environment settings. Optionally, you can place this file in any path (it is not required to be located only in the environments folder).

6. Extract the `IBI_wfce_9.3.5.tar` to get the directory of all the scripts using the following command:

   ```
   $> tar -xvf IBI_wfce_9.3.5.tar
   ```

7. Deploy `infra` components using helmfile or WebFOCUS - Container Edition Health Check Tool:

   a. Deploy the `infra` components using the following helmfile command:

```
$> cd IBI_wfce_9.3.5/scripts/helmfile/infra

$> source ../export-defaults.sh

$> helmfile --state-values-file=<webfocus-
ce>/helmfile/environments/oc-default.yaml.gotmpl --state-
values-set global.imageInfo.image.repository=docker_
username/docker_repo_name sync
```

OR

b. Deploy the `infra` components using WebFOCUS - Container Edition Health Check Tool:

```
$> cd IBI_wfce_9.3.5/scripts

$> source helmfile/export-defaults.sh

$> ./webfocusce run sync --only-run infra  --state-values-file
<webfocus-ce>/helmfile/environments/oc-default.yaml.gotmpl --
state-values-set global.imageInfo.image.repository=docker_
username/docker_repo_name --ignore-healthcheck-errors
```

Verify that the status of `infra` pods are running using the following command:

```
$> oc get pods -n webfocus
```

8. Now, set the `global.ACCEPT_EUA=Y` and `global.WF_CUSTOMERID=<customerid>` in `<webfocusce>/scripts/helmfile/environments/wf.integ.yaml.gotmpl` and replace the active `license.txt` file to the following location `<webfocus-ce>/scripts/license` or update the file path of `global.WF_LICENSE` with your license file location. If you do not know about your license file, then visit our product Support website.
9. Deploy the WebFOCUS - Container Edition using helmfile or WebFOCUS - Container Edition Health Check Tool:
   a. Navigate to `scripts/helmfile` and deploy the WebFOCUS - Container Edition using helmfile command:

```
$> helmfile -e dev --state-values-file=<webfocus-
```

```
ce>/scripts/helmfile/environments/oc-default.yaml.gotmpl --
state-values-set global.imageInfo.image.repository=docker_
username/docker_repo_name sync
```

OR

b. Navigate to `scripts` and deploy the WebFOCUS - Container Edition using WebFOCUS - Container Edition Health Check Tool:

```
$> ./webfocusce run sync -e dev --only-run webfocus --state-
values-file <webfocus-ce>/scripts/helmfile/environments/oc-
default.yaml.gotmpl --state-values-set
global.imageInfo.image.repository=docker_username/docker_repo_
name --ignore-healthcheck-errors
```

Verify the pod status using the following command:

```
$> oc get pods -n webfocus
```

10. Create a route to access the appserver using the following command:

```
$> oc expose svc appserver
```

View the route using the following command:

```
oc get route -n webfocus
```

For more information on the route, see Route Configuration.

11. You can increase the timeout limit of a route by using the following command:

```
$> oc annotate route appserver --overwrite
haproxy.router.openshift.io/timeout=5m
```

Note: The annotation `haproxy.router.openshift.io/timeout` is specific to Red Hat OpenShift's HAProxy-based router. It configures the HTTP timeout for the route. This is the maximum time that the router waits for a response from the backend service before terminating the connection.

Deploying Custom Metrics Horizontal Pod Autoscaling in the Red Hat OpenShift Cluster

Edaserver uses custom metrics for Horizontal pod autoscaling and it is not working with the Red Hat OpenShift cluster.

Prerequisite

If you are using CRC OpenShift, then cluster monitoring should be enable on your cluster. To enable it, use the following command:

```
$> crc config set enable-cluster-monitoring true
$> crc stop
$> crc start
```

To deploy custom metrics Horizontal Pod Autoscaling on the Red Hat OpenShift cluster, perform the following steps:

> ⚠ **Caution:** Code snippets in the PDF format might have undesired line breaks due to space constraints. You must verify code snippets before directly copying and pasting them in your program.

Procedure

1. Log in to the Red Hat OpenShift console as one of the users with a `cluster-admin` privilege.
2. Install KedaController with default values, following is the path: **Operators -> OperatorHub -> Search for "Custom Metrics Autoscaler" -> Select "Custom Metrics Autoscaler" -> Install**.
3. To complete the installation, you need to create `CustomResource KedaController` in `openshift-keda` namespace. Navigate to the installed operator `Custom Metrics Autoscaler` and create an instance of `KedaController` in `openshift-keda` namespace.
4. Check the pod status for KedaController using the command:

```
$> oc get deployment -n openshift-keda
```

Output:

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
custom-metrics-autoscaler-operator	1/1	1	1	3m11s
keda-metrics-apiserver	1/1	1	1	4m36s
keda-operator	1/1	1	1	4m36s

5. Change to the project with the object you want to scale:

```
$> oc project webfocus
```

6. Create a `configmap` named `cluster-monitoring-config` under the `openshift-monitoring` namespace using the following command:

Note: All sample scripts for Red Hat OpenShift are stored in the `<webfocus-ce>/samples/openshift` location.

Note: You need proper access to the user to create `configmap`.

```
$> oc apply -f <webfocus-ce/samples/openshift/config.yaml>
```

7. Check that the `prometheus-operator`, `prometheus-user-workload` and `thanos-ruler-user-workload` pods are running in the `openshift-user-workload-monitoring` project. Use the following command:

```
$> oc -n openshift-user-workload-monitoring get pod
```

It might take a short while for the pods to start.

Output:

NAME	READY	STATUS	RESTARTS	AGE
prometheus-operator-6f7b748d5b-t7nbg	2/2	Running	0	3h
prometheus-user-workload-0	4/4	Running	1	3h
prometheus-user-workload-1	4/4	Running	1	3h
thanos-ruler-user-workload-0	3/3	Running	0	3h
thanos-ruler-user-workload-1	3/3	Running	0	3h

Note:
For the most accurate and up-to-date instructions on auto scaling pods using the Custom Metrics Autoscaler Operator, refer to the official OpenShift documentation:
https://docs.redhat.com/en/documentation/openshift_container_platform/4.19/html/nodes/automatically-scaling-pods-with-the-custom-metrics-autoscaler-operator#nodes-pods-autoscaling-custom-rn-2151-6_nodes-cma-autoscaling-custom-rn

8.

Configure authentication on Red Hat OpenShift monitoring. Create a service account as `thanos` using the following command:

```
$> oc create serviceaccount thanos -n webfocus
```

9. If you are using Red Hat OpenShift 4.14 or later, then create a `secret` yaml to generate a service account token using the following command:

```
$> oc create -f <webfocus-ce>/samples/openshift/secret.yaml> -n
webfocus
```

For more information, see https://docs.openshift.com/container-

platform/4.17/nodes/cma/nodes-cma-autoscaling-custom-trigger.html

10. Get the token (`Tokens`) key value (`thanos-token-XXXX`) from the output using the following command:

```
$> oc describe serviceaccount thanos -n webfocus
```

11. Create a `TriggerAuthentication` custom resource to refer credentials, secrets, or any sensitive information in `ScaledObject`. Copy the token from the above output (name: `thanos-token-XXXX`) and paste into the name section in `triggerauthentication.yaml`.

    Now, create a new `TriggerAuthentication` resource using the following command:

```
$> oc apply -f <webfocus-
ce/samples/openshift/triggerauthentication.yaml> -n webfocus
```

12. Create a new role in the webfocus namespace using `role.yaml`. Use the following command:

```
$> oc apply -f <webfocus-ce/samples/openshift/role.yaml> -n
webfocus
```

13. Bind the role with a service account `thanos` using the following command:

```
$> oc adm policy add-role-to-user thanos-metrics-reader -z thanos -
-role-namespace=webfocus
```

14. Delete the existing edaserver Horizontal Pod Autoscaling using the following command:

```
$> oc delete hpa edaserver -n webfocus
```

15. Create `podMonitor` using the following command:

```
$> oc apply -f <webfocus-ce/samples/openshift/edaserver-
podmonitor.yaml>
```

> **Note:** You need proper access to create `podMonitor`.

16. Deploy `ScaledObject` to enable application autoscaling using the following command:

```
$> oc apply -f <webfocus-ce/samples/openshift/edaserver-
scaledobject.yaml> -n webfocus
```

17. Check the deployment using the following command:

> **Caution:** Code snippets in the PDF could have undesired line breaks due to space constraints and should be verified before directly copying and running them in your program

```
$> oc get scaledobject prometheus-scaledobject -o 'jsonpath=
{..status.conditions[?(@.type=="Ready")].status}'
```

Output should be `True`.

18. Check the Horizontal Pod Autoscaling status using the following command:

```
$> oc get hpa -n webfocus
```

Output:

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
keda-hpa-prometheus-scaledobject	StatefulSet/edaserver	60m/300m (avg)	1	2	1	14h

For more information, see Custom Metrics Autoscaler on OpenShift.

Deploying ibi WebFOCUS - Container Edition Appserver with MS-SQL server or Oracle as a Repository Database

Procedure

1. Download the WebFOCUS - Container Edition `.tar` file and extract it.
2. From the extracted WebFOCUS - Container Edition `.tar` file, navigate to the `<webfocus-ce>/wfc`.
3. Edit the Dockerfile and locate the "Download OpenSource drivers for repository" section.
4. In the "Download OpenSource drivers for the repository" section, update the driver url of MS-SQL server or Oracle.

   ```
   RUN curl -q https://jdbc.postgresql.org/download/postgresql-
   42.3.7.jar \
   -o $JETTY_BASE/lib/ext/pgclient.jar \
   ```

   Replace the previous code with the following code:

   - If you are using MS SQL, then use the following code.

     ```
     RUN curl
     https://repo1.maven.org/maven2/com/microsoft/sqlserver/mssql-
     jdbc/7.2.2.jre11/mssql-jdbc-7.2.2.jre11.jar \
     -o $JETTY_BASE/lib/ext/mssql-jdbc-7.2.2.jre11.jar \
     ```

   - If you are using Oracle, then use the following code.

     ```
     RUN curl https://download.oracle.com/otn-pub/otn_
     software/jdbc/232-DeveloperRel/ojdbc11.jar \
     -o $JETTY_BASE/lib/ext/ojdbc11.jar \
     ```

   - Save the updated Dockerfile.
5. Run the `./build-images.sh` from `<webfocus-ce>/scripts` directory. This creates a

modified WebFOCUS Client (wfc) docker image.

6. You need pre-created db user and database. Use the following commands to grant the appropriate permissions:

```
CREATE USER orcluser IDENTIFIED BY oraclpassword;
GRANT CREATE VIEW TO orcluser;
ALTER SESSION SET CURRENT_SCHEMA=orcluser;
GRANT CREATE SEQUENCE TO orcluser;
GRANT CREATE TABLE TO orcluser;
GRANT CREATE TRIGGER TO orcluser;
GRANT CREATE PROCEDURE TO orcluser;
GRANT CREATE SYNONYM TO orcluser;
GRANT CREATE TYPE TO orcluser;
GRANT CONNECT, RESOURCE TO orcluser;
GRANT CREATE SESSION TO orcluser;
GRANT UNLIMITED TABLESPACE TO orcluser;
```

7. Navigate to the Helmfile `<webfocus-ce>/scripts/helmfile/environment/` directory.
8. Edit the `<webfocus-ce>/scripts/helmfile/environments/wf.integ.yaml.gotmpl` file, navigate to `appserver`, and edit the following sections:
   a. Edit the appserver parameter, and change the `usePgSQL` flag to "false".
   b. Update the `dbuser` and `dbpassword` of your Oracle database.
   c. Update the `dbUrl` and place the `URL` of your Oracle database.

```
appserver:
     usePgSQL: false
     dbUser: <db_username>
     dbPassword: <db_password>
     dbUrl: jdbc:oracle:thin:@<host_name>:1521:ORCL;
```

   If you are using MS SQL, then replace the `jdbc:oracle:thin:@<host_name>:1521:ORCL` with `jdbc:sqlserver://<host_name>:1433;databaseName=<db_name>`.

9. Edit the `config/install_cfg` section and add the new parameter `IBI_REPOS_DB_DRIVER` with repository class value of respective db:

```
appserver:
   config:
```

```
      install_cfg:
          IBI_REPOS_DB_DRIVER: "oracle.jdbc.driver.OracleDriver"
```

If you are using MS SQL, then replace the `oracle.jdbc.driver.OracleDriver` with `com.microsoft.sqlserver.jdbc.SQLServerDriver`.

10. In the global section, update the repository class value as per your respective db:

```
global:
  config:
   REPOSITORY_CLASS: oracle.jdbc.driver.OracleDriver
   JDBC_PATH: "${JETTY_BASE}/lib/ext/*"
```

If you are using MS SQL, then replace the `oracle.jdbc.driver.OracleDriver` with `com.microsoft.sqlserver.jdbc.SQLServerDriver`.

11. Lastly, deploy WebFOCUS - Container Edition to the cluster.

Deploying ibi WebFOCUS - Container Edition using Docker Compose with External MS-SQL or Oracle Database

Procedure

1. Download the WebFOCUS - Container Edition `.tar` file and extract it.
2. From the extracted WebFOCUS - Container Edition `.tar` file, navigate to the `<webfocus-ce>/wfc`.
3. Create a backup of Dockerfile, use the following command:

   ```
   $> cp Dockerfile Dockerfile_bkp
   ```

4. Edit the Dockerfile and locate the "Download OpenSource drivers for repository" section.
5. In the "Download OpenSource drivers for the repository" section, update the driver url of MS-SQL server or Oracle.

   ```
   RUN curl -q https://jdbc.postgresql.org/download/postgresql-
   42.3.7.jar \
   -o $JETTY_BASE/lib/ext/pgclient.jar \
   ```

   Replace the previous code with the following code:

   - If you are using MS SQL, then use the following code.

     ```
     && curl -q -f
     https://repo1.maven.org/maven2/com/microsoft/sqlserver/mssql-
     jdbc/7.2.2.jre11/mssql-jdbc-7.2.2.jre11.jar \
     -o $JETTY_BASE/lib/ext/mssql-jdbc-7.2.2.jre11..jar \
     ```

   - If you are using Oracle, then use the following code.

     ```
     && curl -q -f
     ```

```
https://repo1.maven.org/maven2/com/oracle/database/jdbc/ojdbc1
1/23.3.0.23.09/ojdbc11-23.3.0.23.09.jar
-o $JETTY_BASE/lib/ext/ojdbc11.jar \
```

6. Navigate to the <webfocus-ce>/scripts/docker-compose and edit the .env file.
7. In the .env file, update the Appserver section, comment-out the default Postgres DB-URL and Repository-Class. Update the REPOS_DB_USER and REPOS_DB_PASS. Uncomment and update the REPOS_DB_URLand REPOSITORY_CLASS parameter for MySQL or Oracle database.

```
#Appserver
REPOS_DB_USER= <db-user>
REPOS_DB_PASS= <db-password>
#mssql
#REPOS_DB_URL=jdbc:sqlserver://<hostname>:1433;databaseName= <db-
name>
#REPOSITORY_CLASS=com.microsoft.sqlserver.jdbc.SQLServerDriver
#oracle
#REPOS_DB_URL=jdbc:oracle:thin:@<hostname>:1521:ORCL
#REPOSITORY_CLASS=oracle.jdbc.driver.OracleDriver
```

8. Navigate to the <webfocus-ce>/scripts/docker-compose and update the docker-compose.yaml file.
9. In the docker-compose.yaml file, navigate to services section and comment-out the entire postgres and init_db section.
10. Navigate to Appserver_init_config section, remove the dependency of init_db by commenting the depends_on section and save the docker-compose.yaml file.

```
#Appserver starts
  appserver_init_config:
    #depends_on:
     # init_db:
      #  condition: service_completed_successfully
```

11. Lastly, deploy WebFOCUS - Container Edition with docker compose, For the deployment steps, see the Deploying ibi WebFOCUS - Container Edition Using Docker Compose.

Deploying ibi WebFOCUS - Container Edition on Cloud Environment

Following is the list of supported cloud providers.

- Amazon Web Services (EKS)
- Microsoft Azure (AKS)
- Google Cloud Provider (GKE)

To deploy in cloud environment, you must modify the following components in your configuration:

- **storage.storageClass** - Update this value with the correct storage class being used.
- **storage.accessMode** - Must be set to ReadWriteMany.

Deploying on AWS (EKS)

When tested with the Amazon EFS Container Storage Interface (CSI) driver. The driver provides a CSI interface that allows Kubernetes clusters running on AWS to manage the lifecycle of Amazon EFS file systems.

For more information, reference the following link:

https://docs.aws.amazon.com/eks/latest/userguide/efs-csi.html

> **Note:** Although there are other drivers available, the Amazon EFS CSI driver is recommended since it was tested with WebFOCUS - Container Edition. If you use any other storage driver, configuration settings may vary.

Deploying ibi WebFOCUS - Container Edition with External Solr Server

Users can plug in their existing customized Solr server, and tailor the search experience based on their specific needs. Users can set up search relevance, custom analyzers, schema fields, and integrate with external systems as needed.

Perform the following steps to use external solr search:

Procedure

1. Download the WebFOCUS - Container Edition `.tar` file and extract it.
2. Edit the `<webfocus-ce>/scripts/helmfile/environments/wf.integ.yaml.gotmpl` and set the false value for external solr use.

   ```
   solr:
      install: false
   ```

3. Update the hostname and port of an external solr in the following section.

   ```
   global:
      config:
          IBI_INFOSEARCH_SOLR_URL: "http://<host_name>:<solr_
   port>/solr"
   ```

   Update the value for external solr adminUser, adminPassword, host, and port.

   ```
   solr:
       adminUser: admin
       adminPassword: solr123
       host: <host_name>
       port: <solr_port>
   ```

   Update the hostname and port of the external solr in the below section.

   ```
   appserver:
       config:
   ```

```
       install_cfg:
       IBI_INFOSEARCH_SOLR_URL: "http://<host_name>:<solr_
port>/solr"
```

Update the SOLR_ADMIN_USER and SOLR_ADMIN_PASSWORD of the external solr.

```
initHookSolr:
  SOLR_ADMIN_USER: admin
  SOLR_ADMIN_PASSWORD: solr123
```

4. After changing, deploy the infra and webfocus component

Deploying ibi WebFOCUS - Container Edition for Prometheus in Different Namespace and Same Cluster

Perform the following steps to deploy WebFOCUS - Container Edition for Prometheus in different Namespace and same Cluster:

Procedure

1. Download the WebFOCUS - Container Edition tar file and extract it.
2. Build the images using the `./build-images.sh` from the scripts directory.
3. From the extracted WebFOCUS - Container Edition tar file, navigate to `<webfocus-ce>/scripts/helmfile/environments` and update the `<webfocus-ce>/scripts/helmfile/environments/wf.integ.yaml.gotmpl` file.
4. In the global section of the `<webfocus-ce>/scripts/helmfile/environments/wf.integ.yaml.gotmpl` file, disable the default Prometheus and set the enabled as "false".
5. Update the `url` and `port` in the `prometheusAdapter` section that is to be deployed in the different namespace, see the following section.

```
global:
  prometheus:
    enabled: false

  prometheusAdapter:
    prometheus:
      url: http://prometheus-server
      port: 80
```

6. Deploy Prometheus externally using a different namespace instead of the default namespace.
7. Deploy the infra (excluding Prometheus) and webfocus using the default namespace.
8. Open the Prometheus server UI and check the target metrics for CLM and Eda-server.

Deploying ibi WebFOCUS - Container Edition with Existing PVC

You can deploy WebFOCUS - Container Edition using your existing PVC, providing greater flexibility in managing your storage resources. You can easily adjust the capacity and other characteristics of existing PVCs to meet the changing needs of your applications, eliminating the need for complex provisioning or migration processes.

Perform the following steps to create PVC and PV:

Procedure

1. Navigate to `scripts/samples`, and update the correct nfs mount path and nfs server IP in the `pvc.yaml` file and run the following command:

   ```
   $> kubectl apply -f "pvc.yaml"
   ```

   It creates new PVC and pv under the `webfocus` namespace.

   Optional, If pvc is not bound with the nfs storage class, then you can set the nfs storage class as default and recreate pvc.

2. Create a sub-directory for the volume under the nfs mount path.

3. Navigate to `scripts/samples`, and update the correct nfs mount path and nfs server IP in the "volume_subdir.yaml" file and run the following command:

   ```
   $> kubectl apply -f "volume_subdir.yaml"
   ```

   It creates a subdirectory for WebFOCUS - Container Edition resources volume mount - under the specified nfs path.

4. Set the value to "true" for `global.useExistingPvc` in `<webfocus-ce>/scripts/helmfile/environments/wf.integ.yaml.gotmpl` file to skip pvc creation.

> **Note:** If `useExistingPvc` is set to `true`, then it uses the existing pvc and it does not create new pvc. If the user set the `useExistingPvc` to `false` then it creates new pvc for deployment.

```
global:
   useExistingPvc: true
```

5. Deploy the WebFOCUS - Container Edition with existing pvc and run the following command from the `scripts/helmfile` path.

> **Note:** While deploying WebFOCUS - Container Edition, it refers `pvc.yaml.gotmpl` file for pvc. So before deployment, ensure that existing pvc names are correctly defined for WebFOCUS - Container Edition resources in the `<webfocusce>/scripts/helmfile/environments/pvc.yaml.gotmpl` file.

```
$> helmfile -e cloud --state-values-
file="/scripts/helmfile/environments/pvc.yaml.gotmpl" sync
```

Securing ibi WebFOCUS - Container Edition with Service Mesh

This outlines security measures for network traffic within a WebFOCUS - Container Edition deployment on Kubernetes. It focuses on securing both incoming traffic to the WebFOCUS environment and internal traffic between components.

For more information, see Securing WebFOCUS Container Edition with Service Mesh (Linkerd).

Updating Expired License

If your `edaserver` and `clm` pod failed and gave the `Server's license has expired` error, then you need to update the license.

To update the expired license file, perform the following steps:

Procedure

1. You need the updated license file. If you do not have it, then visit our product Support website.
2. Replace the new license key in the `<webfocus-ce>/scripts/license/license.txt` file.
3. Replace the `license.txt` in `eda-license-secret`. `clm` pod, redeployment updates the license key in `eda-license-secret`, using the following command:

```
$> cd scripts/helmfile/
$> source ./export-defaults.sh
$> helmfile -e <env-name> -l name=clm sync
```

4. Ensure the new license key is correctly replaced, using the following command:
   a. Get the license key from `eda-license-secret`, using the following command:

```
$> kubectl get secret eda-license-secret -n webfocus -o json |
grep license.txt
```

   b. Copy `<license key>` from the `eda-license-secret` to decode the license key and update using the following command:

```
$> echo <license_key> | base64 --decode
```

   c. Compare the output with your new license key, it should be same.
5. Restart the `clm` and `edaserver` pod.

Installing Python 3.9 in ibi WebFOCUS Reporting Server Docker Image

Procedure

1. From the extracted WebFOCUS - Container Edition .tar file, navigate to the `<webfocus-ce>/wfs/iserver_etc`.
2. Edit the `Dockerfile`, uncomment the section of the python installation part and save the `Dockerfile`.
3. Build the images using `./build-images.sh` command from `<webfocus-ce>/scripts` directory.
4. Deploy the WebFOCUS - Container Edition. For more information about deployment, see Deploying ibi WebFOCUS - Container Edition on Kubernetes Cluster.
5. Once the deployment is done, log in to the WebFOCUS - Container Edition GUI, navigate to the **Management center > Administration console > Configuration > Reporting server > Cluster management > EDSERVE** and right-click on the WebFOCUS Reporting Server console.
6. In the WebFOCUS Reporting Server console, click the **Tools icon**  , navigate to the **Workspaces > Configuration > Workspace - edaserve.cfg**.
7. Edit the `[Workspace]` section in the `workspace-edaserve.cfg` file and add the following.

```
pyth_home = /opt/ibi/srv/etc/python
pyth_access = y
pyth_rel = 39
```

8. Click on the **Save icon**  to save the changes.
9. For verifying that your configuration is correct, navigate to the WebFOCUS Reporting Server homepage and click the **Data icon**  , then click the Procedure. Use the following command and click the **Run icon**  .

```
EX _edahome/catalog/adaptivp PYTHON,ECHO=ALL
```

Using the Swego Utility

Swego is a utility provided for WebFOCUS - Container Edition development (dev) environments that enable you to traverse through files in a container from a web browser. It is useful for administrators who may not have access to containers that are running, but perform configuration modifications, and need to view resulting changes in the actual file system.

By default, the file system provides read-only access for users. Users can only view files from a web browser, but cannot make any changes directly.

You can change the default behavior from read-only (true) to false by modifying the Helm charts.

> **Note:** It is not recommended to enable Swego for any other environments, especially production (prod), as it exposes configuration files to users. Swego is intended for testing purposes only.

Swego can be accessed using the following URL:

`http://<hostname>:31081`

> **Note:** To disable swego, you need to update the configuration in the `<webfocus-ce>/scripts/helmfile/environments/wf.integ.yaml.gotmpl` file.

```
swego:

  enabled: false
```

Configuring an Air Gap Installation Using Images From a Single Repository

An air gap installation places all required images and charts in a single repository, and only references this repository during deployment. It avoids downloading any component during runtime or from multiple repositories.

Since all required charts are packaged in the WebFOCUS - Container Edition `.tar` file, no downloading occurs during runtime or from the respective repositories.

This section provides the required steps and commands to configure an air gap installation, which is summarized as follows:

Procedure

1. Build/pull the required images.
2. Tag and push the images to your repository.
3. Update the repository path and tags in the local.yaml.gotmpl file.
4. Run the `sync` command with the `state-values-file` parameter.

For demonstration purposes, AWS ECR is used as an example repository where all images are stored.

Step 1: Building/Pulling the Required Images

The images referenced in the table below are required to run WebFOCUS - Container Edition. Four images are built from the `build-images.sh` script (appserver, edaserver, clm, and cachemanager). For more information, see Creating or Using Docker Images.
Any other (non-ibi) images that are required must be pulled from the respective repositories. The following table also includes the pull commands for non-ibi images.

Service Pod Name	Description	Default Tag
appserver and reportcaster	Image is built from the `build_image` script. The base image used is “redhat/ubi9”. Once the image is created, tag the image and push the image to your repository. Scan the Docker image before issuing a push operation to the repository to check if the image has any vulnerabilities.	WF_TAG = wfc-9.3.5
clm and edaserver	Image is built from the `build_image` script. The base image used is “redhat/ubi9”. Once the image is created, tag the image and push the image to your repository. Scan the Docker image before issuing a push operation to the repository to check if the image has any vulnerabilities.	RS_TAG = wfs-9.3.5
eda-etc(Static stored content of edaserver)	Image is built from the `build_image` script. The base image used is “redhat/ubi9”. Once the image is created, tag the image and push the image to your repository. Scan the Docker image before issuing a push operation to the repository to check if the image has any vulnerabilities.	ETC_TAG = wfs-etc-9.3.5 (From export-defaults.sh)
cachemanager	Same as above, additionally uses the node:18 image as well.	cachemanager, however you can overwrite before building the image or tag the image with a different name before issuing a push operation.
etcd	You must pull/download the `bitnami/etcd:3.5.12-debian-12-r7` image from the Docker hub, tag the image, and push	3.5.12-debian-12-r7

Service Pod Name	Description	Default Tag
	the image to your repository. Pull command: `docker pull bitnami/etcd:3.5.12-debian-12-r7`	
postgresql	You must pull the postgresql:11.12.0-debian-10-r1 image from the Docker hub, tag the image, and push the image to your repository. Pull command: `docker pull bitnami/postgresql:11.12.0-debian-10-r1`	postgresql:11.12.0-debian-10-r1

prometheus-server	Pull the `quay.io/prometheus/prometheus:v2.43.0` image from the Docker hub.
	Pull commands:
	`docker pull quay.io/prometheus/prometheus:v2.43.0`
	`docker pull jimmidyson/configmap-reload:v0.5.0`	

prometheus-adapter	Pull the `k8s.gcr.io/prometheus-adapter/prometheus-adapter:v0.10.0` image from Docker hub. Pull command: `docker pull k8s.gcr.io/prometheus-adapter/prometheus-adapter:v0.10.0`	v0.10.0
solr	Pull the `bitnami/solr:9.8.1-debian-12-r9` image from the Docker hub, tag the image, and push the image to your repository. Pull command: `docker pull bitnami/solr:9.8.1-debian-12-r9`	9.8.1-debian-12-r9
zookeeper	Pull the bitnami/zookeeper:3.9.3-debian-12-r15 image from the Docker hub, tag the image, and push the image to your repository.	3.9.3-debian-12-r15

Service Pod Name	Description	Default Tag
	Pull command: `docker pull bitnami/zookeeper:3.9.3-debian-12-r15`	
metrics-server	metrics-server:v0.5.2 `docker pull rainbond/metrics-server:v0.5.2`	v0.5.2

swego
(Optional if you enabled Swego)
	Pull commands:
	`docker pull ishswar/swego:1.0.0`
	`docker pull ployst/nginx-ssl-proxy`
	`docker pull xmartlabs/htpasswd`	

Step 2: Tagging and Pushing the Images to Your Repository

Once you pull and build all the images, you must tag and push these images to your repository.

The following are examples of pull, tag, and push operations to an AWS ECR repository:

```
$> docker pull bitnami/etcd:3.5.12-debian-12-r7

$> docker tag bitnami/etcd:3.5.12-debian-12-r7 0123xxxxxx.dkr.ecr.us-west-2.amazonaws.com/my_repo:tag_name

$> docker push 0123xxxxxx.dkr.ecr.us-west-2.amazonaws.com/my_repo:tag_name
```

Note: Ensure to scan the Docker image before issuing a push operation to the repository to check if the image has any vulnerabilities.

Step 3: Updating the Repository Path and Tags in the local.yaml.gotmpl File

A sample file (local.yaml.gotmpl) is provided in the environments directory:

```
<webfocus-ce>/scripts/helmfile/environments/local.yaml.gotmpl
```

You can overwrite this sample file or create a new copy and update the file based on your environment settings. Optionally, you can place this file in any path (it is not required to be located only in the environments folder).

Step 4: Running the Sync Command With the State-Values-File parameter

This is an important step in the normal scenario where `helmfile sync` is run with only passing `-e <env-name>`. However, with the air gap installation/deployment, there is a change in how the sync command is run. In this case, you must pass the file with the `state-values-file` parameter using the `helmfile sync` command as follows:

```
$> cd <webfocus-ce>/scripts/helmfile/infra

$> source ../export-defaults.sh

$> helmfile --state-values-file=<file-path> sync

$> cd <webfocus-ce>/scripts/helmfile

$> helmfile -e <env-name> --state-values-file=<webfocus-
ce/scripts/helmfile/environments/local.yaml.gotmpl> sync
```

This command deploys the cluster using the values that were specified in the file.

You can also overwrite other configuration parameters in the same file. The following helmfile switches can be used to overwrite configuration parameters:

```
--state-values-set
--state-values-file
```

All customizations are in the `wf.integ.yaml.gotmpl` and `cloud-wf.integ.yaml.gotmpl` files.

For example, if you want to overwrite the value of the repository as shown in the following sample, you need to know the exact hierarchy.

```
global
  imageinfo:
     image:
    repository: ibi2020/webfocus
```

In this case, it is: `global.imageInfo.image.repository`

To provide a new value during helmfile execution, you can overwrite it as follows:

```
$> helmfile -e cloud --state-values-set
global.imageInfo.image.repository=123xxxxxxx.dkr.ecr.us-west-
2.amazonaws.com/airgaptest sync
```

If you have additional values to overwrite, you can provide them as comma-separated values. For example:

```
$> helmfile -e cloud --state-values-file=$ECR_REPO_IMAGES --state-
values-set global.imageInfo.image.repository=123xxxxxxx.dkr.ecr.us-west-
2.amazonaws.com/airgaptest,edaetc.securityContext.enabled=true,swego.ena
bled
=true sync
```

If you require more values to overwrite, you can create a customization file, and then provide it as input using the `--state-values-file` helmfile switch.

Using Your Own PostgreSQL Database

You can use your own PostgreSQL database instance with WebFOCUS - Container Edition instead of the default. To point to your PostgreSQL database instance, edit the `<webfocus-ce>/scripts/helmfile/environments/wf.integ.yaml.gotmpl` environment file and update the following postgres parameters with your host, username, and password.

> **Note:** Database changes are only applicable for new installations/deployments of WebFOCUS - Container Edition in a Kubernetes cluster.

If you do not want to use a script to create a user and database in PostgreSQL, and would like to use a pre-created PostgreSQL user and database with WebFOCUS - Container Edition, then use the following steps:

Procedure

1. Create a new user in PostgreSQL. For more information, see Sample PostgreSQL Commands to Create a New User and Databases.
2. Create a database and set the user created in step 1 as the database owner. The environment variable PLATFORM_NAME is the default name of the database. If you are required to use a different database name, then update the value of the key in the `<webfocus-ce>/scripts/helmfile/environments/wf.integ.yaml.gotmpl` file.
    a. Under the global.config section, update the following value for the WebFOCUS Client (AppServer) database:

```
global:
  config:
    postgresUser: <postgres-username>
    postgresPassword: <postgres-password>
    pgDBName: <db-name>
```

    b. If you are using your existing or new database, then set `useExistingDB: true`, and `install: false`. Update the `adminUsername`, `adminPassword`, `host`, and `port`.

```
global:
  postgres:
    useExistingDB: true
    install: false
    adminUsername: <postgres-admin-username>
    adminPassword: <postgres-admin-password>
    host: <hostname>
    port: "5432"
```

3. Lastly, deploy the WebFOCUS - Container Edition with existing database.

Important: The `helmfile destroy` command does not delete PostgreSQL database tables when `global.postgres.useExistingDB=true`.

Sample PostgreSQL Commands to Create a New User and Databases

Connect to the PostgreSQL console using the following command:

```
$> psql -h <hostname> -p <port> -d <database> -U <username>
```

Create a new user and databases using the following commands. Note, commands may change based on the specific version of PostgreSQL being used.

```
$> CREATE USER <username> WITH NOSUPERUSER NOCREATEDB NOINHERIT;

$> alter user <username> with encrypted password <PASS>;

$> grant <username> to postgres;

$> CREATE DATABASE <db-name> OWNER <username>;
```

Install NGINX with Helm

The NGINX Ingress controller can be installed via Helm using the chart from the project repository. To install the chart with the release name `ingress-nginx`, enter the following commands in your terminal window/console:

```
$> helm repo add ingress-nginx https://kubernetes.github.io/ingress-
nginx

$> helm repo update

$> helm install ingress-nginx ingress-nginx/ingress-nginx
```

To detect which version of the ingress controller is running, enter the following commands in your terminal window/console:

```
$> POD_NAME=$(kubectl get pods -l app.kubernetes.io/name=ingress-nginx -
o jsonpath='{.items[0].metadata.name}')

$> kubectl exec -it $POD_NAME -- /nginx-ingress-controller --version
```

Ingress Controller with Custom Context Root

Following are the steps for the Ingress Controller with Custom Context Root:

Procedure

1. Navigate to the `<webfocus-ce>/scripts/helmfile/environment/` directory and edit the `<webfocus-ce>/scripts/helmfile/environments/wf.integ.yaml.gotmpl` file.
2. Update the value for `servingDomain`.

   ```
   platform:
     servingDomain: <host_name>
   ```

3. Edit the `ingress` section and add the below parameter.

```
ingress:
    enabled: true
    annotations:
      nginx.ingress.kubernetes.io/configuration-snippet: |
        proxy_set_header    WF_proxyUrl /cldq01;
        proxy_pass_request_headers on;
        proxy_set_header Referer https://<host_name>/cldq01;
      nginx.ingress.kubernetes.io/rewrite-target: /webfocus/$2
      nginx.ingress.kubernetes.io/app-root: /cldq01/webfocus
      nginx.ingress.kubernetes.io/proxy-cookie-path: ~*^/.* /cldq01
    path: /cldq01(/|$)(.*)
```

Note: `cldq01` is the context root path, and you can replace `cldq01` with your context path value.

4. Access the appserver with Ingress custom context root, for example: `https://<host_name>/cldq01`.

Common Deployment Scenarios (Use Cases) for Kubernetes Clusters

This section describes common deployment scenarios (use cases) for WebFOCUS - Container Edition when working with specific Kubernetes cluster configurations.

Deploying to a Single-Node Kubernetes Cluster Running on the Same Machine

This scenario outlines how WebFOCUS - Container Edition can be deployed to a single-node Kubernetes cluster that is running on the same machine as all other ancillary components (for example, Helmfile, Docker, Kubernetes).

Procedure

1. Install Docker.
2. Install Kubeadm.
3. Install Kubectl, Helm, and Helmfile.
4. Create a Kubernetes cluster.

Usage Considerations

In a single-node Kubernetes cluster, you cannot scale the configuration (for example, adding a new node), unless you use images from a Docker registry and a storage provider that supports the ReadWriteMany access mode.

Deploying to a Remote Kubernetes Cluster

This scenario outlines how WebFOCUS - Container Edition can be deployed to a remote Kubernetes cluster, where you need to build Docker images and push these images to the registry.

In this scenario, access to the Kube config file (`~/.kube/config`) is required. Ensure that the ancillary components (specifically, Kubectl, Helm, and Helmfile) on your local machine have connectivity to the remote Kubernetes cluster.

If your Kube config file has many cluster-info (contexts), then ensure you switch to the correct cluster context (the one you intend to use).

Procedure

1. Create a Kubernetes cluster using Kubeadm or any of the supported cluster creation techniques.
2. On a local machine, install Docker, Kubectl, Helm, and Helmfile.
3. Build your Docker images and push your images to the Docker registry to which your Kubernetes cluster has access.

> **Note:** You need to update your pull Secret (`ibi-docker-hub`) and the repository name for your image registry (default is `ibi2020/webfocus`).

Usage Considerations

If your remote Kubernetes cluster is multi-node and you want to scale WebFOCUS - Container Edition, then you must use a storage provider that supports the ReadWriteMany access mode.

If you are planning to scale the WebFOCUS Client (AppServer), then you require an Ingress controller (for example, NGINX) so you can load balance incoming browser traffic requests between application servers. A sample Ingress controller YAML file is provided with the product.

You can generate a sample YAML file (Kubernetes manifest) for an Ingress controller and all other Kubernetes objects using the following command:

```
$> PLATFORM_NAME=dummy
$> RS_TAG=dummy
$> WF_TAG=dummy ETC_TAG=dummy
$> helmfile -e cloud template > templates.yaml
```

> **Note:** This command assumes you are in the `~/webfocus-ce/scripts/helmfile` folder.

When generated, open the `templates.yaml` file and search for the `kind: Ingress`.

The pull Secret for your remote image registry can also be found in the `templates.yaml`. Search for `ibi-docker-hub`.

Deploying to a Local or Remote Kubernetes Cluster Where Docker Images Exist

This scenario outlines how WebFOCUS - Container Edition can be deployed to a local or remote Kubernetes cluster, but you already have WebFOCUS - Container Edition images pushed to the image registry. This is a typical use case where WebFOCUS - Container Edition has been previously installed in an environment.

Procedure

1. Ensure that a remote Kubernetes cluster (single-node or multi-node) is available.
2. On a local machine, install Kubectl, Helm, and Helmfile. Docker is not required.
3. Update your pull Secret (`ibi-docker-hub`) and the repository name for your image registry (default is `ibi2020/webfocus`).

Usage Considerations

Just as described in the previous topic Deploying to a Remote Kubernetes Cluster, if you are planning to use a multi-node Kubernetes cluster, then you must use a storage provider that supports the ReadWriteMany access mode. You also require an Ingress controller (for example, NGINX) if you plan to scale the WebFOCUS Client (AppServer).

Testing Specifications

The deployment scenarios in the previous sections consider Amazon® Elastic Kubernetes Service (EKS) as remote Kubernetes clusters. If there are any distinctions for particular managed Kubernetes clusters, ensure to incorporate those as well. EKS has been tested as follows:

Procedure

1. EKS was created using `eksctl` with unmanaged nodes (version 1.20). Node EC2 instance types were `c5.2xlarge`.
2. Storage driver was Amazon EFS CSI, backed with an EFS drive in the same region/same VPC as the EKS cluster (unencrypted).
3. All the ancillary software, such as PostgreSQL for the WebFOCUS Client (AppServer), Apache Solr, and Zookeeper were running on the same cluster.
4. The database adapter for the WebFOCUS Reporting Server (Edaserver) was Amazon RDS for PostgreSQL.
5. Application server scaling was tested using the NGINX Ingress controller with AWS ELBv2 (network load balancer). Sticky session cookie support was provided by the NGINX Ingress controller.
6. While running Helmfile, the cloud environment file (including the product) was used, as per the following command:

   ```
   $> helmfile -e cloud sync
   ```

7. By default, the swego web server is only enabled in a `dev` environment. If you require swego in your configuration, then add the following switch to your sync command:

   ```
   --state-values-set swego.enabled=true
   ```

   For example:

   ```
   $> helmfile -e cloud --state-values-set swego.enabled=true sync
   ```

Helmfile Environments Reference

The following table provides a reference for the files used and Helmfile commands for each environment.

Environment Name	Files Used	Sample Command	Description
dev	All values come from `<webfocus-ce>/scripts/helmfile/environments/wf.integ.yaml.gotmpl` and `environments/dev-wf.integ.yaml.gotmpl`, which means the last loaded file wins (overwrites) previous values.	`helmfile -e dev sync`	dev environment is used for single node cluster and readwriteonce access mode. Assumes that Docker images are available locally or they are available to the Kubernetes cluster locally. No image pull is required and the default Kubernetes storage driver is used.
cloud	Just like `dev`, most values come from `<webfocus-ce>/scripts/helmfile/environments/wf.integ.yaml.gotmpl`, but others come from `environments/cloud-wf.integ.yaml.gotmpl`.	`helmfile -e cloud sync`	Assumes that Docker images are available on a remote image registry.

Environment Name	Files Used	Sample Command	Description
			The default image repository name is `ibi2020/webfocus` and the image pull Secret is `ibi-docker-hub`. In this case the Kubernetes storage driver must support the ReadWriteMany access mode, and an Ingress controller object for the WebFOCUS Client (AppServer) is created.
default	All values from `<webfocus-ce>/scripts/helmfile/environments/wf.integ.yaml.gotmpl`.	`helmfile sync`	Assumes that Docker images are available locally or they are available to the Kubernetes cluster locally. No image pull is required and the default Kubernetes

Environment Name	Files Used	Sample Command	Description
			storage driver is used.

Post-Deployment Configuration Options

After deploying WebFOCUS - Container Edition, you can configure WebFOCUS - Container Edition in your Kubernetes cluster.

Scaling in Kubernetes

To scale your deployment (for example, by adding more WebFOCUS Reporting Server), enter the following command in your terminal window/console:

```
$> kubectl scale statefulset.apps/edaserver --replicas=3 -n webfocus
```

Note: If required, you can also scale WebFOCUS Client (AppServer) and Cluster Manager (CLM) in your Kubernetes cluster.

Configuring Horizontal Pod Autoscaling

WebFOCUS - Container Edition supports Horizontal Pod Autoscaling in a Kubernetes cluster.

A `HorizontalPodAutoscaler` automatically updates a workload resource (such as a Deployment or StatefulSet), with the objective of automatically scaling the workload to match demand. In Kubernetes, horizontal scaling means that the response to increased load is to deploy more pods.

In WebFOCUS - Container Edition, the following components can be autoscaled:

- WebFOCUS Reporting Server (edaserver)
- WebFOCUS Client (appserver)
- Cache Manager (cachemanager)

For `appserver` and `cachemanager`, CPU-based autoscaling is implemented. In this scenario, if CPU usage is greater than the percentage set limit or threshold value for a specific period, then Kubernetes automatically scale up by increasing the number of available pods.

Similarly, if the CPU threshold is under a specified percentage set limit for a specific period, then Kubernetes automatically scale down by decreasing the number of available pods (to the minimum that is allowed).

For `edaserver`, custom metrics are implemented for autoscaling. This is a combination of CPU percentage utilization and server average response time data from the WebFOCUS Reporting Server, or the number of failed resources (`edaserver_resource_fails_total`) consistently for a specific period. If any of these conditions are met, then Kubernetes autoscale the `edaserver` pod.

Installing (Enabling) the Metrics Server

The CPU data that is used for custom metrics and CPU-based autoscaling calculations is provided by the Metrics server. To install the Metrics server with the Prometheus adapter, use the following flag:

```
global.prometheusAdapter.metricsServer=true
```

Note:

- This flag is disabled by default (=false).
- If the Metrics server is already running in the Kubernetes cluster, then keep it disabled.

Installing (Enabling) the Prometheus Server

Helmfile installs the Prometheus server. If the Prometheus server is already running, then disable it by using the following flag:

```
global.prometheus.enabled=false
```

Note: This flag is enabled by default (=true).

You can provide configuration details for the Prometheus server in:

```
global.prometheusAdapter.prometheus.url

global.prometheusAdapter.prometheus.port
```

Modifying Threshold Values for Horizontal Pod Autoscaling

You can modify (overwrite) the default threshold values that are used for autoscaling pods.

For `appserver` and `cachemanager`, where CPU-based autoscaling is implemented:

```
autoscaling:
   enabled: false
   minReplicas: 1
   maxReplicas: 10
   targetCPUUtilizationPercentage: 80
```

For example: `appserver.autoscaling.targetCPUUtilizationPercentage=60`

This set/update the threshold value for the `appserver`. If the CPU usage goes above 60%, then Kubernetes automatically scale up by increasing the number of available pods.

For `edaserver`, where custom metrics are implemented:

```
edaserver:
  autoscaling
    custom
      systemRunningAvgLoadIndexRatio: 2
```

For example: `edaserver.autoscaling.custom.systemRunningAvgLoadIndexRatio=10`

Note: You can also add this setting in a Go template file to be applied in a specific environment. For example:

```
edaserver:
  imagePullSecrets:
  - name: ibi-docker-hub
  autoscaling:
   custom:
    systemRunningAvgLoadIndexRatio: 2
```

Retrieving a List of Horizontal Pod Autoscaling Components

Use the following command to retrieve a list (and current status) of WebFOCUS components impacted by Horizontal Pod Autoscaling:

```
$> kubectl -n webfocus get hpa
```

For example:

```
Workstation:~/ibi/source/docker-miscellaneous/webfocus-deployment/scripts/helmfile$ kubectl -n webfocus get hpa
NAME           REFERENCE                  TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
appserver      StatefulSet/appserver      0%/80%    1         10        1          10m
cachemanager   StatefulSet/cachemanager   0%/80%    1         100       1          21m
edaserver      StatefulSet/edaserver      0/2       1         3         1          14m
Workstation:~/ibi/source/docker-miscellaneous/webfocus-deployment/scripts/helmfile$
```

Modifying Autoscaling Behavior

The following syntax can be added for each Horizontal Pod Autoscaling definition:

```
behavior:
  scaleDown:
    stabilizationWindowSeconds: 300 # (wait for 5 minutes before scaling
down)
    policies:
    - type: Percent
      value: 100
      periodSeconds: 15 # (Scale down 1 pod every 15 seconds)
  scaleUp:
    stabilizationWindowSeconds: 120 # (wait for 2 minutes before scaling
up)
    policies:
    - type: Percent
      value: 100
      periodSeconds: 15 # (Scale up 1 pod every 15 seconds)
    - type: Pods
      value: 1
      periodSeconds: 30 # (i.e., scale up 1 pod every 30 seconds)
    selectPolicy: Max
```

Useful Commands

The following is a set of useful commands for debugging custom metrics:

```
$> kubectl get --raw /apis/custom.metrics.k8s.io/v1beta1 | jq .

$> kubectl get --raw
"/apis/custom.metrics.k8s.io/v1beta1/namespaces/webfocus/pods/*/
edaserver_running_avg_response_time" | jq .

$> kubectl get --raw
"/apis/custom.metrics.k8s.io/v1beta1/namespaces/webfocus/pods/*/
edaserver_resource_fails_total" | jq .
```

Modifying the Graceful Shutdown Time of the WebFOCUS Reporting Server (edaserver) Pod

When terminating pods during Horizontal Pod Autoscaling, to change the default shutdown (termination) time for the WebFOCUS Reporting Server (edaserver) pod, which is 60 seconds, change the following configuration value:

```
terminationGracePeriodSeconds: 60
```

This configuration value is located in the edaserver statefulset YAML file.

Removing (Undeploying) ibi WebFOCUS - Container Edition

To remove (undeploy) WebFOCUS - Container Edition from a Kubernetes cluster, enter the following commands in your terminal window/console:

```
$> cd <webfocus-ce>/scripts/helmfile/

$> source export-defaults.sh

$> helmfile -e <env-name> destroy

$> cd <webfocus-ce>/scripts/helmfile/infra

$> helmfile destroy
```

Note: The `<env-name>` parameter refers to the environment (dev or cloud). Destroy the Kubernetes cluster in the same environment that it was previously created.

Upgrading ibi WebFOCUS - Container Edition

WebFOCUS - Container Edition supports in-place upgrades in a Kubernetes cluster, where the previous configuration and logs are preserved. Note that Persistent Volumes (PVs) are not recreated.

The following components are updated during the in-place upgrade:

- WebFOCUS Reporting Server (Edaserver)
- WebFOCUS Client (AppServer)
- Cluster Manager (CLM)
- Cache Manager
- ReportCaster

Optionally, the following components can also be upgraded:

- PostgreSQL
- Apache Solr
- Prometheus (server and adapter)

Important: Export the configuration changes that were made in the previous version of your configuration files to the new version of these files. You can use an external YAML file to implement all configuration changes instead of making these changes in the default files, and then add them into the external file of the respective environment or inject them during the sync command.

Use the following steps to upgrade the WebFOCUS - Container Edition:

Note: It is mandatory to set `global.ACCEPT_EUA=Y` and `global.WF_CUSTOMERID=<customerid>` in `<webfocusce>/scripts/helmfile/environments/wf.integ.yaml.gotmpl` before proceeding to WebFOCUS - Container Edition deployment. If you do not know your customer ID, then visit our product Support website.

Procedure

1. Download the new .tar file.
2. Create (build) the Docker images. Ensure that you use the new image tag (for example, `ibi2020/webfocus:wfs-9.3.5`). For more information, see Creating or Using Docker Images.
3. Enter the following command in your terminal window/console:
   `<webfocus-ce>/scripts/build-images.sh`

   The following are the sample output image ID, size of the image may differ:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ibi2020/webfocus	wfc-9.3.5	985ce3d2c6fd	31 seconds ago	1.71GB
ibi2020/webfocus	wfs-etc-9.3.5	299ff83abcbd	2 minutes ago	2.97GB
ibi2020/webfocus	wfs-9.3.5	7d7143535eb1	4 minutes ago	919MB
ibi2020/webfocus	cm-9.3.5	7c4ac8476aed	3 minutes ago	242MB

4. Export the new image tags:

```
export PLATFORM_NAME=webfocus
export INFRA_NS=webfocus
export WF_TAG=wfc-9.3.5
export RS_TAG=wfs-9.3.5
export ETC_TAG=wfs-etc-9.3.5
export CM_TAG=wfs-etc-9.3.5
```

5. (Optional) Ensure the helm-diff plug-in (https://github.com/databus23/helm-diff) is available in your environment. To verify, run the following command:

```
$> helmfile -e <env-name> diff
```

   where:

   `<env-name>`

   Is the specific environment (dev or cloud).

This command shows the changes to be reviewed in your environment before upgrading. It is recommended to run `diff` before upgrading to verify all changes.

6. Upgrade infra by navigating to the `~/scripts/helmfile/infra` directory and running the following command:

```
$> helmfile sync
```

> **Note:** Upgrading infra does not delete existing data in the database. This step is optional and ran if there is any change in Postgres/solr/Prometheus or Prometheus-adapter.

7. Review the resulting changes and then return to the ~/scripts/helmfile directory.
8. Run the following command to upgrade the WebFOCUS components:

```
$> helmfile -e <env-name> -l upgradable=true sync
```

After you run this command, all containers are restarted and the in-place upgrade is complete.

> **Note:** If you are running a low CPU configuration, then it is recommended to pass the edaserver.testStartDelaySeconds=30 parameter with --state-values-set while upgrading.

9. (Optional) To verify that the chart is updated to the latest version, run the following command:

```
$> helm list -n <namespace>
```

where:

`<namespace>`

Is the PLATFORM name set by the export variable in step 4 (for example, `PLATFORM_NAME=webfocus`).

You see output returned similar to the following:

NAME	NAMESPACE	REVISION	STATUS	CHART	APP VERSION
appserver	webfocus	1	deployed	appserver-9.3.5	9.3.5

10. (Optional) To verify that the required pods are upgraded correctly, run the following command:

```
$> kubectl get pods -n <namespace>
```

where:

`<namespace>`

Is the PLATFORM name set by the export variable in step 4 (for example, `PLATFORM_NAME=webfocus`).

Ensure that the following pods are in a running state after the upgrade:

- appserver-0
- cachemanager-0
- clm-0
- edaserver-0
- etcd-0
- reportcaster-0

Upgrading Consideration for WebFOCUS Reporting Server (Edaserver), CLM, and Cache Manager

These components have a dependency on the `eda-etc` persistent volume. During the upgrade process, content is deleted from this volume and copied from the new image. In addition, no configuration changes are made, only the binaries are updated.

WebFOCUS Client (AppServer) Pre-upgrade Scripts

The following scripts are run before upgrading WebFOCUS Client (AppServer):

```
/bin/sh-x
${IBI_DOCUMENT_ROOT}/utilities/dbupdate/db_check_version.shUSERNAME=
{configuration} PASSWORD={configuration}
COMPLETION_STATUS_FILE=${IBI_DOCUMENT_ROOT}/logs/db_check_version_
status.log

/bin/sh-x
${IBI_DOCUMENT_ROOT}/utilities/dbupdate/db_inplace_update.shUSERNAME=
{configuration} PASSWORD={configuration}

/bin/sh-x
${IBI_DOCUMENT_ROOT}/utilities/WFReposUtil/update_repos.shUSERNAME=
$WF_ADMIN_USER PASSWORD=$WF_ADMIN_PASS
```

Performing a Health Check or Smoke Test in the Kubernetes Cluster

The smoke-test utility can be run after you deploy WebFOCUS - Container Edition in a Kubernetes cluster. Test results can be reviewed in the deployment logs. You can also run this utility separately at any point in your Kubernetes cluster to check the overall health of the cluster that is running.

Enter the following commands:

```
$> cd <webfocus-ce>/scripts/helmfile

$> helmfile -e <env>  -l name=smoke-test sync
```

Important: If you used --state-values-set or --state-values-file with the main sync command during deployment, ensure you pass the same here. Otherwise the smoke-test utility does not provide accurate results.

Example:

```
$> helmfile -e dev --state-values-set
global.imageInfo.image.repository=ibi2020/webfocus -l name=smoke-test
sync
```

Updating Docker Images Without Destroying a Kubernetes Cluster

This section describes how to update Docker images without destroying a Kubernetes cluster. There are two options that you can use:

- When Not Changing the Container Image Name or Tag
- When Changing the Container Image Name or Tag

When Not Changing the Container Image Name or Tag

To update Docker images without destroying a Kubernetes cluster when not changing the container image name or tag:

Procedure

1. Ensure that the `imagePullPolicy` property is set to `Always` in the configuration.
2. Scale the application down to 0 and back to the original size using replicas. Note, the following commands use the WebFOCUS Client (AppServer) pod as an example.
    a. Scale down your application to 0:

    ```
    $> kubectl scale statefulset.apps/appserver --replicas=0 -n
    webfocus
    ```

    b. Scale up your application to the original size (for example, 1):

    ```
    $> kubectl scale statefulset.apps/appserver --replicas=1 -n
    webfocus
    ```

When Changing the Container Image Name or Tag

Running the `helmfile sync` command is useful when new versions of the images are available and you want to upgrade your Kubernetes cluster to use these images.

Before continuing, ensure that the new Docker images are available in the registry. In addition, ensure the helm-diff plug-in (https://github.com/databus23/helm-diff) is available in your environment.

If this plug-in is currently not installed, enter the following command in your terminal window/console:

```
$> helm plugin install https://github.com/databus23/helm-diff
```

Procedure

1. Export the following environment variables with the new image tag names:

   ```
   export WF_TAG=<new-image-tag>
   export RS_TAG=<new-image-tag>
   export ETC_TAG=<new-image-tag>
   export PLATFORM_NAME=webfocus
   ```

2. Before running `helmfile sync`, run a `diff` to verify that Helm picked up the new image tags. Navigate to the `scripts/helmfile` directory and enter the following commands:

   ```
   $> cd <webfocus-ce>/scripts/helmfile
   $> helmfile -e <env-name> diff
   ```

   Note: Ensure you reference the proper environment by using `-e`.

3. Run `helmfile sync` if you see the desired image in `diff`:

   ```
   $> helmfile -e <env-name> -l upgradable=true sync
   ```

   Once you run the `helmfile apply` command, the running pods are terminated and new pods are created using the updated images.

To check your result (if required), enter the following command in your terminal window/console:

```
$> helm list -n webfocus
```

You see chart revisions increased by one.

You can also see the history of a particular service and roll back or apply a specific revision. This is useful in the event of a failure or undesired result and you need to revert/roll back changes.

To view a history of WebFOCUS Client (AppServer) Helm charts, enter the following command in your terminal window/console:

```
$> helm history appserver -n webfocus
```

WebFOCUS - Container Edition Health Check Tool

WebFOCUS - Container Edition Health Check Tool is a utility available in WebFOCUS - Container Edition, which allows users to check and troubleshoot issues with WebFOCUS - Container Edition.

This utility is intended to be used before a user proceeds with building images and before deploying WebFOCUS - Container Edition. The utility can also be used after deployment.

The tool resides in a particular folder structure and cannot be moved into any other folder. However, the tool can be run from other locations (that is, using full or relative paths).

Key features include the ability to:

- Run "deploy" and "destroy" of WebFOCUS - Container Edition cluster with and without infra components.
- Run the template command to print helm charts templates.
- Added "top" command for getting memory usage of webfocus deployed pods.
- Confirm current user permissions to run commands for building images.
- Verify the installation of the required software (for example, Docker, Helm).
- Confirm the availability of sufficient disk storage.
- Print a hint link (if any error or warning occurs), so users can reference more information about the specific error or warning message.
- Check the current running state of your cluster, and more.

The primary objectives of the WebFOCUS - Container Edition Health Check Tool are listed as follows:

- **Pre-check (before image building)**
    - Helps you validate if any required software is missing building the images.
    - Check if the system user has all of the required permissions to build images.
- **Pre-check (before deployment)**

    - Helps you validate local Helm files and also validate the remote Kubernetes cluster.
    - Checks the availability of all tools that are required for deployment.
    - Checks the remote Kubernetes cluster to verify that all components are properly configured for deployment.
    - Checks permissions for create/edit/delete using kubectl commands.
    - Checks the existence of POD, Deployment, Job, Statefulset, Ingress, HPA, Network Policy, PVC, and PV.
    - Checks the image pull policy.
- **Post-check (after deployment)**
    - Validates the deployment of WebFOCUS - Container Edition in the Kubernetes cluster.
    - Check to ensure all WebFOCUS - Container Edition components are running and confirms their connectivity.
- **collectLogs**
    - It collects logs from a running cluster that can be used for debugging purposes.
- **Describe**
    - Shows the overall health of the WebFOCUS - Container Edition components running in the cluster.
- **Upgrade**
    - Checks the WebFOCUS - Container Edition configuration and cluster environment for potential upgrading problems. The check upgrade command performs a series of checks to validate that the WebFOCUS - Container Edition cluster-wide resources are configured correctly and that all required software is installed properly. If the command encounters a failure, it prints additional information about the failure and exit with a non-zero exit code.
- **Run**
    - Checks the pre-deploy healthcheck and validates the environment. If everything works fine, then it proceeds deployment and destroys the webfocus resources. It logs all the commands run, which the end user can be used for future

reference.

- Print the template of helm charts.

- **Top**
  - It shows the memory/CPU usage of the running WebFOCUS - Container Edition cluster resources.

Utility Name: `webfocusce`

Utility Location: `cd <webfocus-ce>/scripts`

The following table lists and describes the available commands, subcommands, and their usage.

Command	Subcommand	Usage	Options/Flags	Options Usage
help		Prints usage of all commands.		
check		Runs write permission check and location check.	-h, --help	
	build	Checks the WebFOCUS - Container Edition installation and resources for potential building problems.	-h, --help	Help for image building.
	collectLogs	Check that the user has valid permission to	--context	Name of the kubeconfig context to use.

Command	Subcommand	Usage	Options/Flags	Options Usage
		fetch and export WebFOCUS - Container Edition log.		

Command	Subcommand	Usage	Options/Flags	Options Usage
			--kubeconfig	Path to the kubeconfig file to use for webfocusce commands.
			--output	Output format. It could be table, JSON or short (default "table").
			--outputfile	Output file. Only applies to JSON output.
			--verbose	Turn on debug logging.
			--wait	Maximum allowed execution time to trigger retry for failed tests (default 5m0s).
	deploy	Checks the WebFOCUS - Container Edition configuration and cluster environment for potential deploying problems.	-f, --helmfile	Full path of the helmfile you want to use.
			-e, --helmfile-env	Environment selector of helmfile.
			-h, --help	Help for deploy.
			--state-values-file	Specify state values in a YAML file.
			--state-values-set	Set state values on the command line. You can specify multiple or

Command	Subcommand	Usage	Options/Flags	Options Usage
				separate values using commas (for example, key1=val1,key2=val2).
	Post-deploy	Checks the WebFOCUS - Container Edition resources in the deployed environment for potential problems.	-h, --help The same flags are available as for deploy.	Help for post-deploy.
	upgrade	Checks the WebFOCUS - Container Edition resources in the deployed environment for potential upgrade.	-h, --help The same flags are available as for deploy.	
describe		Describes all resources on the deployed component.	-h, --help	Help for describe.
run	sync	Checks the WebFOCUS - Container Edition configuration and cluster environment for potential	-h, --help The same flags are available as for deploy. Additional flags for run: --only-run	

Command	Subcommand	Usage	Options/Flags	Options Usage
		deploying problems. If everything is fine, proceed with webfocus deployment. It generates the log file for future reference.	--ignore-healthcheck-errors --selector, -l	
	destroy	It destroys the WebFOCUS - Container Edition resources and generates the log file.	The same flags are available as for deploy. Additional flags for destroy: --only-run --selector, -l	
	template	It prints and logs the WebFOCUS - Container Edition helm chart templates, it also generates the log file.	The same flags are available as for deploy. Additional flags for template: --only-run --selector, -l	
	upgrade	It upgrades the WebFOCUS - Container Edition resources, and it also generates the log file.	The same flags are available as for deploy. Additional flags for upgrade: --only-run --selector, -l	

Command	Subcommand	Usage	Options/Flags	Options Usage
top		It shows the memory/CPU usage of the running WebFOCUS - Container Edition cluster resources.		
version		Shows the current version of the WebFOCUS - Container Edition Health Check tool.		
collectLogs		It collects logs from a running cluster that can be used for debugging purposes.		

The following table lists and describes the available global options.

Global Option	Description
-context	Name of the kubeconfig context to use.
-kubeconfig	Path to the kubeconfig file to use for CLI requests.
-output	Output format, which is either basic, JSON, or short (default "table").
-outputfile	Name of the output file.

Global Option	Description
-help	Provides a list of all commands and subcommands available for the tool.
-verbose	Enables debug logging.
-wait	Maximum allowed time for all tests with retry to pass (default 5m0s). **Note:** Applicable for the check command.

Important:

For Windows User:

Note: Use command prompt to run health check tool command.

1. To export variables for WebFOCUS - Container Edition Check Tool, run the following command in the command prompt:

```
$> cd <webfocus-ce>\scripts
$> helmfile\export-defaults.bat
```

2. To run the health check tool, use this command:

```
$> cd <webfocus-ce>\scripts
$> webfocusce.exe help
```

Note: On windows, replace “./webfocusce” with “webfocusce.exe” file for execute the healthcheck tool commands.

For Linux Users:

1. To export variables for webfocus Container Edition Check Tool, use the following commands:

```
$> cd <webfocus-ce>/scripts
$> source helmfile/export-defaults.sh
```

2. To run the health check tool, use this command:

```
$> cd <webfocus-ce>/scripts
$> ./webfocusce help
```

Additional Command Details

```
$> cd <webfocus-ce>/scripts

$> ./webfocusce help
```

Provides details of all commands and subcommands along with their available options and parameters.

```
$> ./webfocusce check
```

Performs common checks, such as whether the tool is running from the correct location, the user has required permissions.

```
$> ./webfocusce check build
```

Performs checks, such as whether Docker is installed or not, required Docker images are pulled from the registry, required disk space is available.

```
$> ./webfocusce check deploy -e <env> -f <path-to-helmfile>
```

Performs checks when deploying required tools for WebFOCUS - Container Edition, such as helm, whether kubectl is installed, the namespace exists or not, permissions are available to perform create/edit/delete operations, pull secrets are correct or not, and <env> is "dev" or "cloud".

```
$> ./webfocusce check post-deploy  -e <env> -f <path-to-helmfile>
```

Performs checks after WebFOCUS - Container Edition is deployed or running in the cluster. This also checks whether all pods are in the correct state, and ensures that all services, statefulset, and deployments are as expected.

```
$> ./webfocusce describe
```

The "describe" command is to show the status of resources of WebFOCUS container edition. It prints name, status, and other information.

```
$> ./webfocusce run sync -e <env> -f <path-to-helmfile>
```

Performs pre deployment healthcheck, then it proceeds with deployment. If it passes, it proceeds to deployment for both "infra" and "webfocus" components. If you want to deploy "infra" or “webfocus" components separately, use the parameter/flag “–only-run”. This command also generates a log file of sync under logs folders.

```
$> ./webfocusce run destroy -e <env> -f <path-to-helmfile>
```

It skips the healthcheck and destroys the WebFOCUS - Container Edition resources running in the cluster. If you want to destroy “infra" and “webfocus" components separately, then use the parameter/flag “–only-run”. This command also generates a log file of destroy under logs folders.

```
$> ./webfocusce run template -e <env> -f <path-to-helmfile>
```

It prints the WebFOCUS - Container Edition helm chart template. This command also generates a log file of the template under the logs folders.

```
$> ./webfocusce collectLogs
```

It collects logs from a running cluster that can be used for debugging purposes. This command generates a log tar file of the webfocus container edition resources under the logs folders.

```
$> ./webfocusce top pod
```

It prints the CPU and memory usage by the WebFOCUS - Container Edition pods in the cluster.

Note: It requires a metrics server enabled on the cluster.

Adding Option to Mount Extra Volume to the ibi WebFOCUS Reporting Server (Edaserver) Pod

Procedure

1. Navigate to the `scripts/helmfile/environments` location, uncomment `extraVolumes` and `extraVolumeMounts` in `edaserver`. Update `claimName` (PVC_name) and `mountPath` in `edaserver` section of the `wf.integ.yaml.gotmpl` file.

```
edaserver:
  extraVolumes:
  - name: extra
    persistentVolumeClaim:
      claimName: pvc2
  extraVolumeMounts:
  - name: extra
    mountPath: /opt/ibi/srv/temp/extra
```

> **Note:** You can use your PVC to mount an extra volume.

2. Deploy `edaserver` to mount extra volume, navigate to the `scripts/helmfile` path and run the following command:

```
$> cd <webfocus-ce>/scripts/helmfile/
$> source export-defaults.sh
$> helmfile -e dev -l name=edaserver sync
```

3. After successful deployment, access the `edaserver` pod with the following command and check that the `extra` volume is created under the `edaserver` pod.

```
kubectl exec -it edaserver-0 -n webfocus -- /bin/sh
o/p:=>
$> ls
$> clm-0  edaserver-0  extra
```

Backing Up and Restoring Kubernetes Cluster on AWS S3 using Velero

Velero is an open-source tool. It can be run with a Cloud Provider or on premises software. You can use Velero to back-up and restore your Kubernetes cluster resources and also replicate your production cluster to development and testing clusters.

To take a backup and restore your Kubernetes cluster, perform the following steps:

Procedure

1. Install Velero
2. Create a WebFOCUS backup
3. Restore a WebFOCUS backup
4. Add multiple (S3 bucket) back-up locations
5. Deploy Velero in EKS using IAM role
6. Deploy Velero using the helm chart

Install Velero

Procedure

1. To install the Velero command-line interface on your system, use the following commands:
   a. Set the latest version of velero in `VELERO_VERSION`.

   ```
   VELERO_VERSION=v1.10.0
   ```

   Note: To check the latest version, see Velero Version.

   b. Download the .tar file using the `wget` command:

> **Caution:** Code snippets in the PDF could have undesired line breaks due to space constraints and should be verified before directly copying and running them in your program

```
wget
https://github.com/vmware-tanzu/velero/releases/download/$VELERO_
VERSION/velero-$VELERO_VERSION-linux-amd64.tar.gz
```

c. Extract the downloaded file using the `tar` command:

```
tar -xvf velero-$VELERO_VERSION-linux-amd64.tar.gz
```

d. Set the file permissions using the `chmod` command:

```
chmod +x velero-$VELERO_VERSION-linux-amd64/velero
```

e. Use the `cp` command to copy the file to your given location.

> **Caution:** Code snippets in the PDF could have undesired line breaks due to space constraints and should be verified before directly copying and running them in your program

```
sudo cp velero-$VELERO_VERSION-linux-amd64/velero
/usr/local/bin
```

2. Create the "s3-credentials" file and update aws_access_key_id, aws_secret_access_key, aws_session_token with valid parameters, refer to the below s3-credentials sample file.

```
[default]
aws_access_key_id=ASIA2TQTLBEVZQHU2WV2
aws_secret_access_key=+K9EHu+y5XXmlPBVoBsvPTfVjwiMUHdDKjjY+/cZ
aws_session_
token=FwoGZXIvYXdzEGgaDAd3IszIO1V7X2yenCK0ASb9s0ysMBeJO6
xbgg3LojCr5Xmmk9yvrZioyh0RvPM4BFwSMqjZItIDpTGlcAJK357TkbhI1CCG3IZtu
C
```

```
UqKLaZhfNQ7NQOzHWKtBU5CGx75uI1NQLwLenGwK6evehlK5igMUmYdaLxVS28sMVDY
t
qhS09BeLQB6TAqmxUf2K2ByUMjlG2h+DQ+ZjzFwsp3+pWFob+iNC62IlnyFC6K+PP5A
W
9dL0Zld64+wLWvAFNty5SpuCiKtbOpBjItvDF/AHBzRg4ci3tZiytoG4p5qCWJMXD3j
O
CPbpWWzSrGMSmjvkVVrXOBUI4u
```

3. To deploy Velero with Kopia, use the following command:

   - Create the S3 bucket on AWS console with the appropriate region. Set the `BUCKET` name and `REGION` using the following command:

     ```
     BUCKET=<bucket_name>
     REGION=<aws_region_name>
     ```

   - Use the created "s3-credential" file and follow the following command:

     ```
     velero install \
           --provider aws \
           --plugins velero/velero-plugin-for-aws \
           --bucket $BUCKET \
           --backup-location-config region=$REGION \
           --snapshot-location-config region=$REGION \
           --secret-file ./s3-credentials \
           --default-volumes-to-fs-backup \
           --uploader-type kopia \
           --use-node-agent
     ```

     Tip: Kopia is a quick and safe open-source backup and restore application that enables you to make encrypted copies of your data, and save the images to local disk, network-attached storage, or a server of your choosing, as well as remote or cloud storage of your choice.

     Kopia works well with the `efs-sc` storage class on AWS, but it does not support the socket and pipe file types. To do this, we create a `.kopiaignore` file in the clm pod "/opt/ibi/srv/temp". Use the following command:

```
kubectl -n webfocus exec -it clm-0 -- /bin/bash
echo ".tscom300">/opt/ibi/srv/temp/.kopiaignore
exit
```

Create a WebFOCUS backup

- Use the `backup` command to create the webfocus backup.

> **Note:** It is recommended to provide a valid `dockerconfigjson` in `<webfocus-ce>/scripts/helmfile/environments/wf.integ.yaml.gotmpl` file, if you do not have `dockerconfigjson` then delete the `ibi-docker-hub` secret before the back-up process. If you skip the above info, then you may face the `imagepullbackoff` error while restoring WebFOCUS - Container Edition resources due to invalid `dockerconfigjson`.

```
velero backup create <backup_name> --include-namespaces <namespace>
--wait
```

> **Caution:** Code snippets in the PDF could have undesired line breaks due to space constraints and should be verified before directly copying and running them in your program

- Use the `describe` command to check the details for specific webfocus backup.

```
velero describe backup <backup_name> --details
```

Restore a WebFOCUS backup

- Use the `restore` command, to get back the webfocus backup.

```
velero restore create --from-backup <backup_name>
```

- Velero can restore resources into different namespaces using the "--namespace-mappings" flag. Use the following command:

```
velero restore create <restore_name> \
--from-backup <backup_name> \
--namespace-mappings <old-namespace>:<new-namespace>
```

  - After namespace mapping, run the restore command.
  - After restoring the WebFOCUS - Container Edition resources, update the namespace name from Appserver UI.

    Then go to **Management Center → Administration Console → Reporting Servers → Cluster Manager → EDASERVE** and update the namespace for Remote CLM Host location. For example: `clm.new-namespace`.
  - Lastly, redeploy the failing "prom-adapter-prometheus-adapter" pod.

Note: Velero backup and restore does not work as expected, if you are deploying WebFOCUS - Container Edition using infra Postgresql database. It is must use the Velero backup and restore with Amazon RDS or external databses only.

Add multiple (S3 bucket) back-up locations

Procedure

1. You can change the default back-up storage location at any time by setting the "--default" flag using the velero backup-location set command and configure a different location to be the default. For example, refer to the following command:

```
velero backup-location create backups-primary \
    --provider aws \
    --bucket <bucket_1> \
    --config region=$REGION
```

```
velero backup-location create backups-secondary \
    --provider aws \
    --bucket <bucket_2> \
    --config region=$REGION
```

2. Use the below command to set the S3 bucket back-up location.

```
velero backup-location set backups-secondary --default
```

3. Use the below command to check the default back-up location.

```
velero get backup-location
```

Deploy Velero in EKS using IAM role

To deploy velero in EKS using the IAM role, refer to the following link: Backup and restore your Amazon EKS cluster resources using Velero.

Deploy Velero using the helm chart

To deploy velero using a helm chart, refer to the following link: Velero using helm chart.

Useful flags for Velero

If no filtering options are used, Velero includes everything in the backup or restore objects. For more information, see Resource Filtering.

Troubleshooting

This section provides notes for troubleshooting purposes as needed.

Issue	Workaround
Getting an error "The provided token has expired", when running the	• Delete "cloud-credentials" secret from velero namespace:

Issue	Workaround

backup or restore velero command.	**Caution:** Code snippets in the PDF could have undesired line breaks due to space constraints and should be verified before directly copying and running them in your program
	`kubectl delete secret cloud-credentials -n velero`
	• Create an AWS credentials file for S3 access. Refer to Create the "s3-credentials" file and update aws_access_key_id, aws_secret_access_key, aws_session_token with valid parameters, refer to the below s3-credentials sample file.
	• Create "cloud-credentials" secret with new AWS credentials:
	`kubectl create secret generic cloud-credentials \`
	`--namespace velero \`
	`--from-file cloud=s3-credentials`

Creating Kubernetes User with Limited Permissions

When deploying the WebFOCUS - Container Edition on the Kubernetes cluster, it is crucial to ensure that you have appropriate permissions for deployment. You can create Kubernetes user and bind to a role or cluster role with limited permission using RBAC. RBAC helps to restrict users and you can perform only the necessary actions on the cluster.

Use the following steps to create a Kubernetes user and provide the necessary limitation.

Procedure

1. Create a Linux user, or you can use the current login user.
2. To create a private key, use the following command:

```
$> openssl genrsa -out <username>.key 2048
```

3. To create a CSR (Certificate Signing Request), use the following command:

```
$> openssl req -new -key <username>.key \
   -out <username>.csr \
   -subj "/CN=<username>"
```

4. Sign in to the CSR with the Kubernetes CA, use the following command:

```
$> sudo openssl x509 -req -in <username>.csr \
  -CA /etc/kubernetes/pki/ca.crt \
  -CAkey /etc/kubernetes/pki/ca.key \
  -CAcreateserial \
  -out <username>.crt -days 1500
```

> **Note:** This certificate is valid for 1500 days.

5. Create a `.cert` directory and move files into it, use the following command:

```
$> mkdir .certs && mv <username>.crt <username>.key .certs
```

6. To create a Kubernetes user, use the following command:

```
$> kubectl config set-credentials <username> \
    --client-certificate=/home/<username>/.certs/<username>.crt \
    --client-key=/home/<username>/.certs/<username>.key
```

7. To create a Kubernetes context for the user, use the following command:

```
$> kubectl config set-context <username>-context \
    --cluster=kubernetes --user=<username>
```

8. User can set the limitation with roles and clusterrole. With Roles user can set the permission within the namespace and with clusterrole user can set the permission for cluster-wide level. Following is the example for clusterrole.

```
cat <<EOF | kubectl --kubeconfig=/home/<username>/.kube/config
apply -f -
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: webfocus
  name: limitedclusterrole1
rules:
- apiGroups: [""]
  resources: ["namespaces", "nodes","pods/log",
"persistentvolumes", "pods", "services", "persistentvolumeclaims"]
  verbs: ["get", "list", "create"]
- apiGroups: ["apps"]
  resources: ["deployments", "statefulsets", "replicasets"]
  verbs: ["get", "list", "create", "patch", "delete"]
- apiGroups: ["storage.k8s.io"]
  resources: ["storageclasses"]
  verbs: ["get", "list"]
- apiGroups: ["rbac.authorization.k8s.io"]
  resources: ["clusterroles"]
  verbs: ["get", "list"]
EOF
```

9. To bind cluster role to user, use the following command:

```
cat <<EOF | kubectl --kubeconfig=/home/<username>/.kube/config
apply -f -
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: clusterlimitedview
  namespace: webfocus
subjects:
- kind: User
  name: <username>
  namespace: webfocus
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: limitedclusterrole1
EOF
```

10. To use the `current-context`, use the following command:

```
$> kubectl config use-context <context-name>
```

Adding Grafana Dashboard in the Red Hat OpenShift Cluster

Procedure

1. Set up a CRC OpenShift cluster. For more information, see Setup CRC OpenShift Cluster.
2. To enable cluster monitoring, use the `crc config set enable-cluster-monitoring true` command and then stop and start the CRC.
3. Use the following commands to install Grafana:

```
$> helm repo add grafana https://grafana.github.io/helm-charts

$> helm repo update

$> helm install grafana grafana/grafana --set
securityContext.runAsUser=Null,securityContext.fsGroup=Null -n
<namespace>
```

4. Edit Grafana service using the `oc edit svc grafana` command and update `type` from `clusterIP` to `NodePort` as shown in the following section:

```
$> oc edit svc grafana
```

```
spec:
  type: NodePort
```

5. To get the port number of Grafana, use the following command:

```
$> oc get svc
```

6. Access the Grafana UI. http://<hostname>:30052/

   If you are running locally, then use http://localhost:30052/

7. Connect the Prometheus to our Custom Grafana, use the following command:

```
$> oc adm policy add-cluster-role-to-user cluster-monitoring-view -
z grafana
```

8. The bearer token for the service account is used to authenticate access to Prometheus in the openshift-monitoring namespace. Use the following command to display the token.:

```
$> oc create token grafana --duration=8760h -n <namespace>
```

9. Navigate to **Home > Connection > Data sources** and select the default DS as Prometheus.
10. Add the following values to connect with Prometheus:
    - Prometheus server URL: https://thanos-querier.openshift-monitoring.svc.cluster.local:9091
    - In **Auth** section, enable the `Skip TLS Verify`.
    - In **Custom HTTP Headers** section, type Header as `Authorization` and value as for example "Bearer eyJhbGciOiJSUzI1NiIsImtpZCI6Ik9ILThhblJjdTlqU .....".
    - Click **Save and Test**. Check it should be successful.
11. Navigate to **Home -> Dashboards**. Click the **New** button dropdown and select **Import**.
12. Add Grafana ID in **Import via grafana.com** section, select default Data Source - Prometheus and click **Load**.

Third-Party Licenses for the Container Image

Analyze container images and their associated licenses

Base container image

WebFOCUS - Container Edition includes a redhat/ubi9 official container image as the common base image layer for building images. For details on Debian licenses and software package types, see the license information.

As with all container images, the Red Hat ubi9 container image may contain other software (such as bash, glibc, zlib, and others from the base distribution, along with any direct or indirect dependencies of the primary software included in the built image) that is subject to other licenses.

Note: The image user is responsible for ensuring that any use of the image complies with all relevant licenses for all software contained within.

Additional software packages

Building images often installs additional software packages (fetched from the official distribution software repositories, from other user-added repositories, or from specific locations), in addition to installing the packages provided by the base image. You can inspect the Dockerfiles to identify these additional packages. For example, when you read the Dockerfile, you see a list of packages that are installed in the image, as specified in the Dockerfile.

Note: Each such specified package can, in turn, install other software packages as dependencies.

Manually retrieve installed package information

You can inspect a container image and retrieve its contents with standard container and package management tools.

Use the `yum-query` command to retrieve the full list of installed packages in a container image.

Example: To retrieve the list of installed packages in the WebFocus Reporting server image, run the following command:

```
$> docker run --rm --entrypoint yum <image-name> list installed
```

Result

This system is not registered with an entitlement server. You can use a subscription-manager to register.

Installed Packages		
acl.x86_64	2.3.1-3.el9	@System
alsa-lib.x86_64	1.2.7.2-1.el9	@ubi-9-appstream-rpms
alternatives.x86_64	1.20-2.el9	@System
audit-libs.x86_64	3.0.7-103.el9	@System
avahi-libs.x86_64	0.8-12.el9	@ubi-9-baseos-rpms
basesystem.noarch	11-13.el9	@System
bash.x86_64	5.1.8-6.el9_1	@System
bzip2-libs.x86_64	1.0.8-8.el9	@System
ca-certificates.noarch	2022.2.54-90.2.el9_0	@System
cmake-filesystem.x86_64	3.20.2-7.el9	@ubi-9-appstream-rpms
copy-jdk-configs.noarch	4.0-3.el9	@ubi-9-appstream-rpms

Manually retrieve installed packages licenses

Use the `rpm` command to retrieve the license for any package, use the following command:

```
docker run --rm --entrypoint rpm <image-name> rpm -qi <package-name> |
grep License
```

Example: To retrieve the license for the installed package libXtst-1.2.3-16.el9.x86_64, run the following command:

```
docker run --rm --entrypoint rpm <image-name> rpm -qi libXtst-1.2.3-
16.el9.x86_64 | grep License
```

```
Output:
License     : GPLv2+
License     : MIT
```

```
docker run --rm --entrypoint /bin/bash <image-name> -c 'for package in
$(rpm -qa); do echo "Package: $package"; rpm -qi $package | grep
License; done'
```

Manually retrieve installed files

Procedure

1. Use the command docker to extract the contents of a container for further inspection.

   > **Note:** If you create a new container from an image, you can inspect it without running the container.

   Example: To create a temporal container called temp-container, which is based on the unknown-image:latest image, run the following command:

   ```
   $> docker create --name temp-container unknown-image:latest
   ```

2. Extract the container file system as a `.tar` file. Following command is an example.

```
$> docker export temp-container > temp-container.tar
```

You can also directly extract the files list without creating a `.tar` archive. Following command is an example.

```
$> docker export temp-container | tar t > temp-container-files.txt
```

The previous commands create and export the contents of a stopped container. Both commands are direct ways to extract the image's final file system, which is a composite view of a container instance.

Alternatively, to create an image `.tar` file, run the `docker image save` command. Following command is an example:

```
$> docker image save unknown-image:latest > temp-image.tar
```

The previous commands produce an archive that exposes the image format, not the containers created from it. The `.tar` file includes a `manifest.json` file that describes the image's layers and a set of separate directories containing the content of each of the individual layers. This method is helpful when you want to evaluate each layer's role in building the image.

For more information, see the Docker CLI documentation.

For more information on the container image format, see the OCI image format specification.

Manually retrieve installed package sources

You can use the `rpm` command to retrieve the source for an installed package.

Example: To retrieve the source for the installed package, use the following command:

```
docker run --rm --entrypoint /bin/bash <image-name> -c "for package in
\$(rpm -qa); do echo -e \"Package: \$package Source RPM: \$(rpm -qi
\$package | awk '/Source RPM/ {print \$4}')\"; done"
```

Troubleshooting

RHEL 9.x or Rocky Linux 9.x

Issue: RHEL 9.x has a default of 1 billion file descriptors. WebFOCUS - Container Edition takes too long to deal with such a large number, causing the setup process to be unresponsive. If the setup is unresponsive, the installation is incomplete, leading to failures in WebFOCUS - Container Edition scripts.

Solution: Added a conditional check to update the ulimit via ulimit -n 200000. This change has been applied to the isetup pod, CLM, EDA server pod, and upgrade pods that run the inu command.

For more information, see https://access.redhat.com/solutions/1479623?utm_source=chatgpt.com.

Appendix A: Configuration Parameters Reference

This section provides a reference for the configuration parameters that are included in the Helm charts packaged with WebFOCUS - Container Edition.

Export Environment Variable Parameters

This section lists and describes the export environment variable parameters overriding the default parameters of the export-default.sh file.

Parameter	Description	Default Value
PLATFORM_NAME	WebFOCUS components deploy on the given namespace.	webfocus
INFRA_NS	Infra components deploy on the given namespace.	webfocus
WF_TAG	Image tag for WebFOCUS Client or appserver.	wfc-9.3.5
RS_TAG	Image tag for WebFOCUS reporting server or edaserver.	wfs-9.3.5
ETC_TAG	Image tag for eda-etc.	wfs-etc-9.3.5
CM_TAG	Image tag for cachemanager.	cm-9.3.5

Common Parameters

This section lists and describes the common parameters.

Parameter	Description	Default Value
platform.servingDomain	Used when an ingress controller is enabled.	jenkins.dev-cloud.ibi.com
platform.rsNamespace	Not required for single tenant configuration.	PLATFORM_NAME
platform.tlsSecretName	Used when an ingress controller is enabled.	N/A
platform.dockerconfigjson	If you are using a private Docker registry to pull images, then you must update the pull Secret in your configuration. The value of the `.dockerconfigjson` field is a base64 representation of your Docker credentials.	N/A

Note: The PLATFORM_NAME environment variable is set to webfocus by default in the `readme.md` file. This can be modified by exporting a different value for this environment variable.

Global Parameters

This section lists and describes the global parameters.

Parameter	Description	Default Value
global.WF_CUSTOMERID	Customer ID required to deploy the WebFOCUS - Container Edition. Each customer have their own unique Customer ID.	
global.WF_LICENSE	Path of the License file.	{{ readFile "../../license/license.txt" | b64enc }}
global.ACCEPT_EUA	Mandatory to accept End User License Agreement. You must set the value to "Y" to proceed with the deployment, otherwise deployment gets fail.	
global.imageInfo.image.repository	Repository name for your images on registry.	ibi2020/webfocus
global.createNamespace	It creates a	true

Parameter	Description	Default Value
	namespace if it does not exist.	
global.deletePvcOnDestroy	It destroys pvc when you destroy WebFOCUS - Container Edition.	true
global.useExistingPvc	You can use your pre-created PVCs, if you do not want to use the default PVCs.	false
global.config.IBI_INFOSEARCH_SOLR_URL	Solr search server URL.	http://solr.{{ env "INFRA_NS" | default "default" }}.svc.cluster.local:8983/solr
global.config.solrUser	Username of Apache Solr used by edaserver and appserver.	{{ requiredEnv "PLATFORM_NAME" }}
global.config.solrPassword	Password of Apache Solr used by edaserver and appserver.	"llPO0ytSDfk3u7sdfk"
global.config.solrCollection	Solr collection name.	{{ requiredEnv "PLATFORM_NAME" }}
global.config.postgresUser	PostgresSQL username.	{{ requiredEnv "PLATFORM_NAME" }}
global.config.postgresPassword	PostgresSQL password.	llPO0ytSDfk3u7sdfk

Parameter	Description	Default Value
global.config.pgDBName	PostgresSQL database name.	{{ requiredEnv "PLATFORM_NAME" }}
global.postgres.useExistingDB	If you want to existing postgres database.	false
global.postgres.install	Determines whether to install the postgres infra component or not.	true
global.postgres.adminUsername	Postgres admin username.	postgres
global.postgres.adminPassword	Postgres admin user password.	postgres123
global.postgres.host	Postgres database hostname.	postgresql.default.svc.cluster.local
global.postgres.port	Postgres database port.	5432
global.solr.install	Determines whether to install the solr infra component or not.	true
global.solr.createIbiProtectedCfgSet	It create ibi protected config set on the cluster.	true

Parameter	Description	Default Value
global.solr.adminUser	Solr admin username.	admin
global.solr.adminPassword	Solr admin user password.	solr123
global.solr.host	Solr server hostname.	solr.{{ env "INFRA_NS" | default "default" }}.svc.cluster.local
global.solr.port	Solr server port.	8983
global.prometheus.enabled	Determines whether to install prometheus server or not from infra.	true
global.prometheusAdapter.enabled	Determines whether to install prometheusAdapter or not from infra.	true
global.prometheusAdapter.metricsServer		false

ibi WebFOCUS Client (AppServer) Parameters

This section lists and describes the parameters that are used by the WebFOCUS Client (AppServer).

Parameter	Description	Default Value
appserver.name		{{ requiredEnv "PLATFORM_NAME" }}
appserver.searchCollectionName		{{ requiredEnv "PLATFORM_NAME" }}
appserver.usePgSQL	If you want to use external database, then set this value as `false`.	true
appserver.dbUser	External database username.	{{ requiredEnv "PLATFORM_NAME" }}
appserver.dbPassword	External database password.	llPO0ytSDfk3u7sdfk
appserver.dbUrl	External database url.	jdbc:postgresql://postgresql.{{ env "INFRA_NS" | default "default" }}.svc.cluster.local:5432/{{ requiredEnv "PLATFORM_NAME" }}?currentSchema=public
appserver.image.repository	Repository where the Docker image is stored.	ibi2020/webfocus
appserver.image.tag	Docker image tag is obtained from the WF_TAG environment variable.	wfc-9.3.5
appserver.image.pu	The imagePullPolicy and the tag of the	IfNotPresent

Parameter	Description	Default Value
llPolicy	image affect when the kubelet attempts to pull the specified image. For more information, reference the following link: https://kubernetes.io/docs/concepts/configuration/overview/#container-images	
appserver.replicaCount	Number of WebFOCUS Client (AppServer) copies (replicas).	1
appserver.service.type	Allows you to specify the type of service. NodePort is default, others are ClusterIP, ingress.	NodePort
appserver.service.nodePortWFC	WebFOCUS Client port exposed for NodePort service.	31080
appserver.config.install_cfg.IBI_IMPORT_DIRECTORY	The import directory location/path in the `install.cfg` file.	/opt/webfocus/cm/import
appserver.config.install_cfg.IBI_EXPORT_DIRECTORY	The export directory location/path in the `install.cfg` file.	/opt/webfocus/cm/export
appserver.config.install_cfg.IBI_SCM_STAGING_DIRECTORY	The scm directory location/path in the `install.cfg` file.	/opt/webfocus/scm
appserver.config.install_cfg.IBI_TRACE_DIRECTORY	The traces directory location/path in the install.cfg file.	/opt/webfocus/traces
appserver.config.install_cfg.IBI_TEMPORARY_	The temp directory location/path in the `install.cfg` file.	/opt/webfocus/temp

Parameter	Description	Default Value
DIRECTORY		
appserver.config.install_cfg.IBI_MAGNIFY_CONFIG	The magnify directory location/path in the install.cfg file.	/opt/webfocus/config/magnify
appserver.config.install_cfg.IBI_ADMIN_NAME	WebFOCUS Client admin username.	secret
appserver.config.install_cfg.IBI_ADMIN_PASS	WebFOCUS Client admin user password.	terces
appserver.config.install_cfg.IBI_Eula_Acceptance	End-user license agreement.	TRUE
appserver.config.install_cfg.IBI_INFOSEARCH_SOLR_URL	Solr server URL.	http://solr.default.svc.cluster.local:8983/solr
appserver.config.install_cfg.IBI_EMAIL_SERVER	Email server URL.	http://solr.default.svc.cluster.local:8983/solr
appserver.config.install_cfg.IBI_EMAIL_SERVER_PORT	Email server port.	587
appserver.config.install_cfg.IBI_EMAIL_SMTP_USER	SMTP user email.	foo_user
appserver.config.install_cfg.IBI_EMAIL_SMTP_PASS	SMTP user password.	foo_pass

Parameter	Description	Default Value
appserver.ingress.enabled	Set to true if you are using an ingress service type. Routing rules to manage external users' access to the services through HTTPS/HTTP.	false

Note: The ingress is enabled (true) by default when using the cloud environment. It is disabled (false) for the dev environment.

ibi WebFOCUS Reporting Server (Edaserver) Parameters

This section lists and describes the parameters that are used by the WebFOCUS Reporting Server (Edaserver).

Parameter	Description	Default Value
edaserver.image.repository	Repository where the Docker image is stored.	ibi2020/webfocus
edaserver.image.tag	Docker image tag is obtained from the RS_TAG environment variable.	wfs-9.3.5
edaserver.EDAUSER	WebFOCUS Reporting Server admin username.	secret
edaserver.EDAPASSWD	WebFOCUS Reporting Server admin user password.	terces
edaserver.etc.image.tag	Docker image tag for WebFOCUS Reporting Server static content is obtained from the ETC_TAG environment variable.	{{ requiredEnv "ETC_TAG" }}
edaserver.PYSERV_URL	URL of running DSML service. Recommend to update from here or from the web console.	http://dsml
edaserver.createSample	Allow edaserver deployment without dependency on postgres, if samples are not required.	true
edaserver.service.type	Allows you to specify the type of service. Set ClusterIP since we do not want to expose the WebFOCUS	ClusterIP

Parameter	Description	Default Value
	Reporting Server externally.	
edaserver.service.port1	Used to map with the nodeport service.	8120
edaserver.service.port2	Used to map with the nodeport service.	8121
edaserver.service.port3	Used to map with the nodeport service.	8122
edaserver.service.port4	Used to map with the nodeport service.	8123
edaserver.service.nodePort1	Used to expose the port for external access.	31120
edaserver.service.nodePort2	Used to expose the port for external access.	31121
edaserver.service.nodePort3	Used to expose the port for external access.	31122
edaserver.service.nodePort4	Used to expose the port for external access.	31123
edaserver.replicaCount	Number of WebFOCUS Reporting Server copies (replicas).	1
edaserver.ingress.enabled	Set to true if you are using an ingress service type. Routing rules to manage external users' access to the services through HTTPS/HTTP.	false

Note: By default and for convenience, for service.nodePort, the Kubernetes control plane allocates a port from a range (default: 30000-32767).

Cluster Manager (CLM) Parameters

This section lists and describes the parameters that are used by the Cluster Manager (CLM).

Parameter	Description	Default Value
clm.image.repository	Repository where the Docker image is stored.	ibi2020/webfocus
clm.image.tag	The Docker image tag is obtained from the RS_TAG environment variable.	wfs-9.3.5
clm.EDAUSER	CLM admin username.	secret
clm.EDAPASSWD	CLM admin user password.	terces
clm.service.type	Allows you to specify the type of service. Set ClusterIP since we do not want to expose CLM externally.	ClusterIP
clm.service.port1	Used to map with the nodeport service.	8120
clm.service.port2	Used to map with the nodeport service.	8121
clm.service.port3	Used to map with the nodeport service.	8122
clm.service.port4	Used to map with the nodeport service.	8123
clm.service.nodePort1	Used to expose the port for external access.	31120
clm.service.nodePort2	Used to expose the port for external access.	31121
clm.service.nodePort3	Used to expose the port for external access.	31122
clm.service.nodePort4	Used to expose the port for external access.	31123

Parameter	Description	Default Value
clm.replicaCount	Number of CLM copies (replicas).	1
clm.ingress.enabled	Set to true if you are using an ingress service type. Routing rules to manage external users' access to the services through HTTPS/HTTP.	false

Initialized Database Parameters (Application Server / WebFOCUS Client)

This section lists and describes the initialized database parameters that impact the application server / WebFOCUS Client.

Parameter	Description	Default Value
initHookSolr.SOLR_CONFIG	Solr configuration file.	ibi-protected
initHookSolr.SOLR_ADMIN_USER	Solr admin username.	admin
initHookSolr.SOLR_ADMIN_PASSWORD	Solr admin password.	solr123
storage.nfs_server	NFS server for storage.	localhost
storage.accessMode	Storage access mode. Options include: ReadWriteOnce, ReadOnlyMany, ReadWriteMany, and ReadWriteOncePod. It is recommended not to change this value.	ReadWriteMany
storage.capacity	Capacity of storage. Gi refers to Gigabytes. You can also use Ei, Pi, Ti, Gi, Mi, or Ki.	4Gi
storage.storageClass	Separate storageClass attribute for webfocus storage class for backward compatibility.	
storage.infraStorageClass	Separate storageClass attribute for infra storage class for backward compatibility.	

The following table lists and describes the configuration parameters that are used for volume permissions.

Parameter	Description	Default Value
appserver.volumePermissions.enabled	Specify true to set the required file permissions on the persistent volume, specify false for local-storage.	false
edaserver.volumePermissions.enabled	Specify true to set the required file permissions on the persistent volume, specify false for local-storage.	false
clm.volumePermissions.enabled	Specify true to set the required file permissions on the persistent volume, specify false for local-storage.	false

Note: These parameters are required only for NFS mounts. For more information, reference the following link: https://github.com/kubernetes/examples/issues/260

Resource (CPU and Memory) Parameters

This section lists and describes the resource parameters that control CPU and memory (RAM) usage.

By default, resource limits can only be applied to the cloud environment. There are no resource limits for the dev environment. However, you can modify this default behavior if required by including `resource-limit.yaml.gotmpl` for the dev environment in Helmfile.

> **Note:** Limits and requests for ephemeral-storage are measured in bytes. You can express storage as a plain integer or as a fixed-point number using one of the following suffixes: E, P, T, G, M, K

You can also use the following power-of-two equivalents: Ei, Pi, Ti, Gi, Mi, Ki

For example, the following represent roughly the same value: 128974848, 129e6, 129M, 123Mi

Parameter	Description	Default Value
appsever.env.JAVA_OPTIONS	To set the java options for appserver.	-Xms1500m -Xmx1500m
appserver.resources.requests.memory	Appserver/WebFOCUS Client pod/container reserves requested memory in the system.	1Gi
appserver.resources.requests.cpu	Appserver/WebFOCUS Client pod/container reserves requested CPU in the system (500m = 0.5 CPU).	500m
appserver.resources.limits.memory	Appserver/WebFOCUS Client container max limit for the memory (RAM).	2Gi

Parameter	Description	Default Value
appserver.resources.limits.cpu	Edaserver/WebFOCUS Reporting Server container max limit for the CPU.	1000m
edaserver.resources.requests.memory	Edaserver/WebFOCUS Reporting Server pod/container reserves requested memory in the system.	512Mi
edaserver.resources.requests.cpu	Edaserver/WebFOCUS Reporting Server pod/container reserves requested CPU in the system (1 = 1 CPU).	1
edaserver.resources.limits.memory	Edaserver/WebFOCUS Reporting Server container max limit for the memory (RAM).	4Gi
edaserver.resources.limits.cpu	Edaserver/WebFOCUS Reporting Server container max limit for the CPU (2 = 2 CPU).	2
clm.resources.requests.memory	CLM pod/container reserves requested memory in the system.	256Mi
clm.resources.requests.cpu	CLM pod/container reserves requested CPU in the system.	100m
clm.resources.limits.memory	CLM container max limit for the memory (RAM).	512Mi
clm.resources.limits.cpu	CLM container max limit for	500m

Parameter	Description	Default Value
	the CPU.	
swego.resources.requests.memory	Swego pod/container reserves requested memory in the system.	256Mi
swego.resources.requests.cpu	Swego pod/container reserves requested CPU in the system.	100m
swego.resources.limits.memory	Swego container max limit for the memory (RAM).	512Mi
swego.resources.limits.cpu	Swego container max limit for the CPU.	250m
cachemanager.resources.requests.memory	Cache manager pod/container reserves requested memory in the system.	256Mi
cachemanager.resources.requests.cpu	Cache manager pod/container reserves requested CPU in the system.	100m
cachemanager.resources.limits.memory	Cache manager container max limit for the memory (RAM).	512Mi
cachemanager.resources.limits.cpu	Cache manager container max limit for the CPU.	250m

ibi Documentation and Support Services

For information about this product, you can read the documentation, contact Support, and join Community.

How to Access ibi Documentation

Documentation for ibi products is available on the Product Documentation website, mainly in HTML and PDF formats.

The Product Documentation website is updated frequently and is more current than any other documentation included with the product.

Product-Specific Documentation

The documentation for this product is available on the ibi™ WebFOCUS® - Container Edition Documentation page.

How to Contact Support for ibi Products

You can contact the Support team in the following ways:

- To access the Support Knowledge Base and getting personalized content about products you are interested in, visit our product Support website.
- To create a Support case, you must have a valid maintenance or support contract with a Cloud Software Group entity. You also need a username and password to log in to the product Support website. If you do not have a username, you can request one by clicking **Register** on the website.

How to Join ibi Community

ibi Community is the official channel for ibi customers, partners, and employee subject matter experts to share and access their collective experience. ibi Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from ibi products. For a free registration, go to ibi Community.

Legal and Third-Party Notices

SOME CLOUD SOFTWARE GROUP, INC. (“CLOUD SG”) SOFTWARE AND CLOUD SERVICES EMBED, BUNDLE, OR OTHERWISE INCLUDE OTHER SOFTWARE, INCLUDING OTHER CLOUD SG SOFTWARE (COLLECTIVELY, “INCLUDED SOFTWARE”). USE OF INCLUDED SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED CLOUD SG SOFTWARE AND/OR CLOUD SERVICES. THE INCLUDED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER CLOUD SG SOFTWARE AND/OR CLOUD SERVICES OR FOR ANY OTHER PURPOSE.

USE OF CLOUD SG SOFTWARE AND CLOUD SERVICES IS SUBJECT TO THE TERMS AND CONDITIONS OF AN AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER AGREEMENT WHICH IS DISPLAYED WHEN ACCESSING, DOWNLOADING, OR INSTALLING THE SOFTWARE OR CLOUD SERVICES (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH LICENSE AGREEMENT OR CLICKWRAP END USER AGREEMENT, THE LICENSE(S) LOCATED IN THE “LICENSE” FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE SAME TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

ibi, the ibi logo, WebFOCUS, and TIBCO are either registered trademarks or trademarks of Cloud Software Group, Inc. in the United States and/or other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only. You acknowledge that all rights to these third party marks are the exclusive property of their respective owners. Please refer to Cloud SG’s Third Party Trademark Notices (https://www.cloud.com/legal) for more information.

This document includes fonts that are licensed under the SIL Open Font License, Version 1.1, which is available at: https://scripts.sil.org/OFL

Copyright (c) Paul D. Hunt, with Reserved Font Name Source Sans Pro and Source Code Pro.

Cloud SG software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the “readme” file for the availability of a specific version of Cloud SG software on a specific operating system platform.

THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. CLOUD SG MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S), THE PROGRAM(S), AND/OR THE SERVICES DESCRIBED IN THIS DOCUMENT AT ANY TIME WITHOUT NOTICE.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "README" FILES.

This and other products of Cloud SG may be covered by registered patents. For details, please refer to the Virtual Patent Marking document located at https://www.cloud.com/legal.