

# **TIBCO Hawk® – Container Edition**

## **Installation and Configuration Guide**

*Software Release 2.0*  
*March 2019*

## Important Information

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE "LICENSE" FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

ANY SOFTWARE ITEM IDENTIFIED AS THIRD PARTY LIBRARY IS AVAILABLE UNDER SEPARATE SOFTWARE LICENSE TERMS AND IS NOT PART OF A TIBCO PRODUCT. AS SUCH, THESE SOFTWARE ITEMS ARE NOT COVERED BY THE TERMS OF YOUR AGREEMENT WITH TIBCO, INCLUDING ANY TERMS CONCERNING SUPPORT, MAINTENANCE, WARRANTIES, AND INDEMNITIES. DOWNLOAD AND USE OF THESE ITEMS IS SOLELY AT YOUR OWN DISCRETION AND SUBJECT TO THE LICENSE TERMS APPLICABLE TO THEM. BY PROCEEDING TO DOWNLOAD, INSTALL OR USE ANY OF THESE ITEMS, YOU ACKNOWLEDGE THE FOREGOING DISTINCTIONS BETWEEN THESE ITEMS AND TIBCO PRODUCTS.

This document is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIBCO, the TIBCO logo, Two-Second Advantage, TIB, Information Bus, Rendezvous, TIBCO Rendezvous, TIBCO Hawk, TIBCO Hawk Container Edition, TIBCO Hawk Microagent for TIBCO BusinessWorks Container Edition, TIBCO BusinessWorks Container Edition, and TIBCO LogLogic Log Management Intelligence (LMI) are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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.

This 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. Please see the readme.txt file for the availability of this software version 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. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

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 "READ ME" FILES.

This and other products of TIBCO Software Inc. may be covered by registered patents. Please refer to TIBCO's Virtual Patent Marking document (<https://www.tibco.com/patents>) for details.

Copyright © 2018-2019. TIBCO Software Inc. All Rights Reserved.

# Contents

---

<b>Figures .....</b>	<b>5</b>
<b>TIBCO Documentation and Support Services .....</b>	<b>6</b>
<b>TIBCO Hawk® Container Edition Overview .....</b>	<b>7</b>
Hawk Container Edition Features .....	7
Key Hawk Concepts .....	8
<b>Hawk Container Edition Architecture and Components .....</b>	<b>10</b>
Hawk Agent .....	11
Hawk Container Edition Microagents .....	11
Hawk Cluster Manager .....	12
Hawk Console .....	13
<b>Dockerize Hawk Container Edition .....</b>	<b>14</b>
Dockerfiles for Hawk Container Edition .....	14
Building Hawk Container Edition Components Docker Images .....	15
Running Hawk Container Edition Docker Containers in Standalone Mode .....	16
Viewing Container Logs .....	17
Environment Variables for Hawk Container Edition Components .....	17
<b>Running Hawk Container Edition on AWS Based Kubernetes Cluster .....</b>	<b>24</b>
Setting up a Kubernetes Cluster on AWS .....	24
Deploying Hawk Container Edition Containers on AWS Based Kubernetes Cluster .....	25
Hawk Container Edition Components YAML Files .....	26
<b>Rulebase Repository Management in Hawk Console .....</b>	<b>30</b>
Creating Agent Groups .....	31
Creating Rulebase Mapping .....	31
Migrating Rulebases and Schedules from Hawk Agent to Rulebase Repository .....	32
Actions on Rulebase Repository Configuration Objects .....	33
<b>Adding Custom Hawk Plug-Ins to the Hawk Agent .....</b>	<b>37</b>
<b>Hawk Container Edition Programming .....</b>	<b>38</b>

# Figures

---

Hawk Agent in Hawk Container Edition (Docker) ..... 10

Hawk Cluster (Domain) in Hawk Container Edition ..... 10

# TIBCO Documentation and Support Services

---

## How to Access TIBCO Documentation

Documentation for TIBCO products is available on the TIBCO Product Documentation website, mainly in HTML and PDF formats.

The TIBCO Product Documentation website is updated frequently and is more current than any other documentation included with the product. To access the latest documentation, visit <https://docs.tibco.com>.

## Product-Specific Documentation

The following documents for this product is available on the [Hawk® Container Edition](#) product documentation page.

- *TIBCO Hawk® Container Edition Release Notes*
- *TIBCO Hawk® Container Edition Installation and Configuration Guide*
- *TIBCO Hawk® Container Edition Microagent Reference Guide*

## How to Contact TIBCO Support

You can contact TIBCO Support in the following ways:

- For an overview of TIBCO Support, visit <http://www.tibco.com/services/support>.
- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the TIBCO Support portal at <https://support.tibco.com>.
- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to <https://support.tibco.com>. If you do not have a user name, you can request one by clicking Register on the website.

## How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO 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 TIBCO products. In addition, users can submit and vote on feature requests from within the [TIBCO Ideas Portal](#). For a free registration, go to <https://community.tibco.com>.

# TIBCO Hawk® Container Edition Overview

---

Hawk® Container Edition is a tool for monitoring distributed applications in the container environment. Currently, you can monitor only applications deployed in Linux Docker containers by using Hawk Container Edition.

A container consists of an entire runtime environment: an application; all its dependencies, libraries, and other binaries; and configuration files needed to run it bundled into one package. You need not worry about the differences of operating system distribution in case of container application.

To understand concepts of Hawk Container Edition, you must be aware of concepts of Docker and TIBCO Hawk®. For information about Docker concepts, such as, Dockerfile, Docker Image, Container, and so on, see Docker documentation at <https://docs.docker.com/>. For information about key Hawk concepts, see [Key Hawk Concepts](#).

In Hawk Container Edition, a containerized Hawk Agent runs on each Docker host on the network and monitors local conditions. Each agent uses collections of locally loaded rules organized into rulebases to apply monitoring logic. By using rulebase, an agent monitors particular application or system resources and takes actions when specific conditions are detected. Hawk includes pre-bundled microagents and prebuilt rulebases that monitor basic system level parameters, and administrators can build additional rulebases by using editors in Hawk Console. For more information on Hawk Console, see *TIBCO Hawk® documentation*. Rulebases can be selectively loaded to an agent or group of agents on a temporary or permanent basis. For more information about the components of Hawk Container Edition, see [Hawk Container Edition Architecture and Components](#).

## Hawk Container Edition Features

Hawk Container Edition uses TCP Transport for Hawk for communication and inherits many of its benefits. These benefits include a flexible architecture, enterprise-wide scalability, and location transparent product components that are simple to configure.

Hawk Container Edition is based on open architecture and is flexible and lightweight.

### Flexibility

Hawk Container Edition is designed to make your container monitoring strategy seamless.

The Hawk Container Edition containers are configured to maintain all configurations and dependencies internally. The configuration for the component is provided through environment variables instead of configuration files.

### Openness and Extensibility

One of the greatest benefits of Hawk Container Edition is portability. You can run Hawk Container Edition in most of the Docker-based Platform as a Service (PaaS) environments.

Hawk Container Edition also provides public APIs which allow you to develop custom activities (rulebases, AMIs, security, and so on) as per your requirements, see [Hawk Container Edition Programming](#).

You can add additional monitoring capabilities to Hawk Container Edition by adding custom plug-ins to it, see [Adding Custom Hawk Plug-Ins to the Hawk Agent](#).

In Hawk Container Edition, you can use REST APIs to build a custom Hawk Console. The feature to create your own custom console enables you to implement only the console features that are suitable for your business.

## Lightweight

Hawk Container Edition is a lightweight tool for monitoring and managing distributed applications. Hawk Container Edition consumes low resources and runs on a low footprint in your container ecosystem.

## Key Hawk Concepts

Hawk Container Edition has Hawk at its core while extending its capabilities to the container environment.

To understand the basic concepts of Hawk Container Edition, you can go through the following key concepts

### Hawk Agent

A Hawk Agent is an autonomous process on each host to monitor systems and applications on that computer. The Hawk agent operates autonomously and uses sets of rules, called "Rulebases". These rules help Hawk agents to configure system management, status reporting, and automation tasks.

### Hawk Microagent

The term microagent is a generic term used to refer to the Hawk managed set of methods exposed by an application. This is done using either instrumenting the application using Hawk AMI API, or via a Hawk plug-in, or using an adapter. For more details on the microagent methods available with Hawk Container Edition, see *TIBCO Hawk® Container Edition Microagent Reference Guide* [TIBCO Hawk® Container Edition Microagent Reference Guide](#).

### Hawk Application Management Interface (AMI)

Hawk Application Management Interface (AMI) is a set of APIs that allows developer community to extend and enhance instrumentation of various infrastructure components in the network by plugging into the Hawk system and making their applications manageable via the Hawk Agent. For more information, refer to the *TIBCO Hawk® Programmers Guide*.

### Hawk Plug-In

Hawk plug-ins are Java components that reside and run inside the process space of a Hawk Agent. They are used to connect to a third party application using its specific protocols and expose them as microagents to Hawk, thereby enabling them to be managed by Hawk.

### TCP Transport for Hawk

TCP Transport for Hawk is a TCP based transport for Hawk components using the Akka clustering designs. Also, the TCP Transport for Hawk removes the dependency on an external server or transport as the TCP communication happens peer to peer. For more information, refer to the *TIBCO Hawk® Installation, Configuration, and Administration Guide*.

### Rulebases and Rules

A rulebase is a configuration object that allows the Hawk agent to manage systems and applications on the network. A rulebase is a collection of one or more rules, whereas a rule is a user defined monitoring criteria. It specifies a data source in the form of a microagent method, one or more tests that check for conditions, and one or more actions to perform if the test condition evaluates to true. Rules can monitor parameters of an operating system, application or other managed object and perform tasks.



## Monitoring Strategy

To create a monitoring strategy, you must identify the potential sources of problems and analyze how they can be prevented. Most monitoring tasks can be automated, but you must identify the unique situations that occur in your infrastructure. For more information on how to analyze the situations on your infrastructure that could be monitored and how to translate monitoring requirements into elements of Hawk rulebase, refer to the *TIBCO Hawk<sup>®</sup> Concepts Guide*.

For more information about Hawk-related concepts, see [TIBCO Hawk<sup>®</sup>](#) documentation.

# Hawk Container Edition Architecture and Components

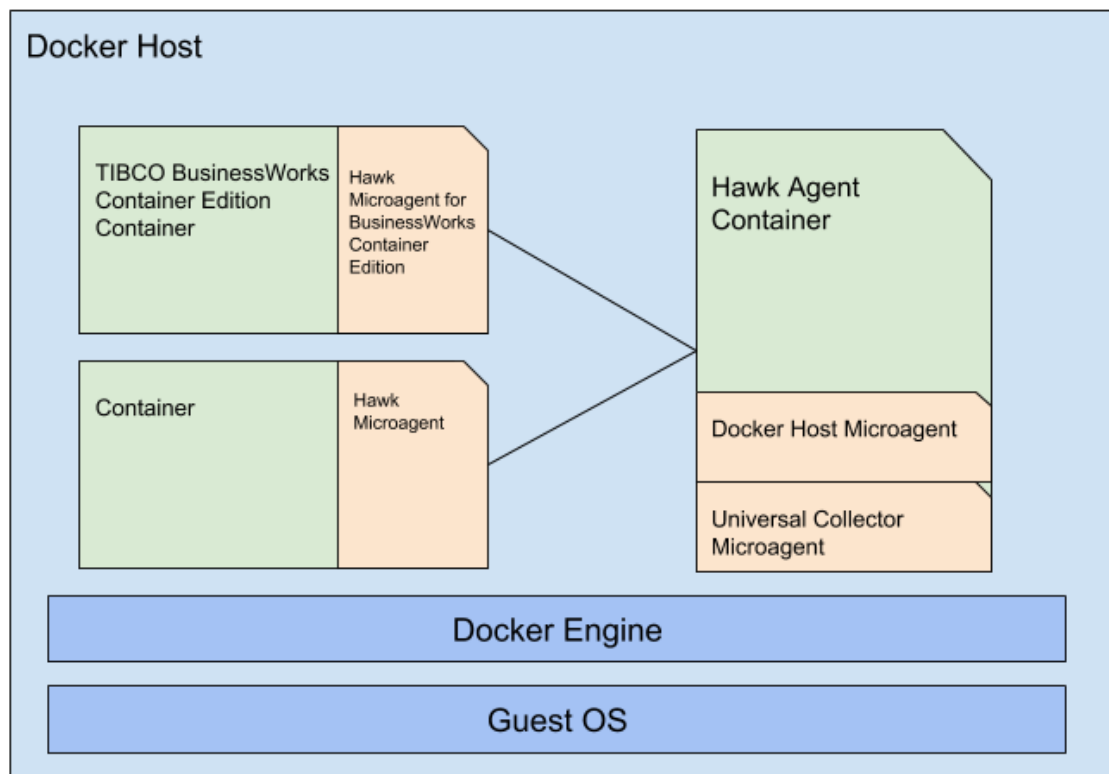
To monitor applications in Docker environments, each component of Hawk Container Edition should run in a Docker container.

To implement the monitoring capabilities of Hawk in Docker environment, you must create a Docker image of each component. These component images run in their Docker containers and communicate with each other using the TCP based transport. For more information about TCP Transport for TIBCO Hawk, see [TIBCO Hawk® Documentation](#).

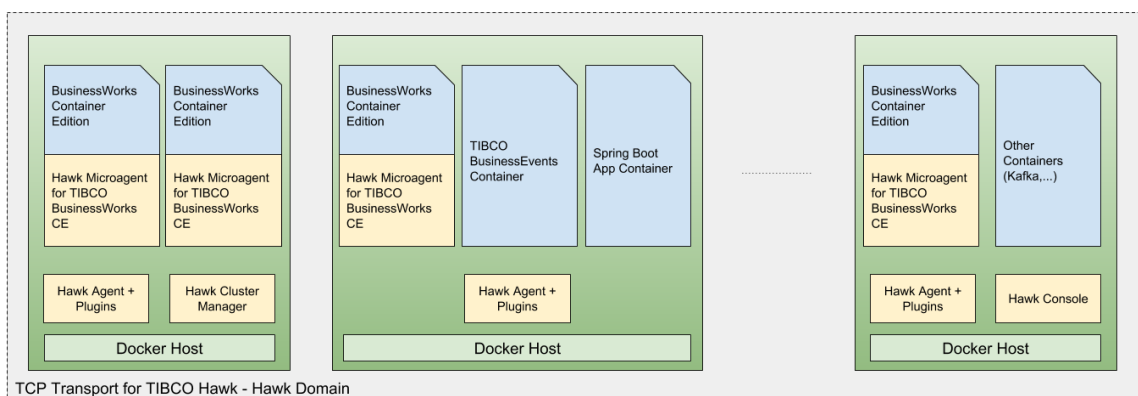
For better understanding of Hawk Container Edition architecture, see the following architecture diagrams.

## Hawk Container Edition Architecture in Docker

### *Hawk Agent in Hawk Container Edition (Docker)*



### *Hawk Cluster (Domain) in Hawk Container Edition*



The following components of Hawk Container Edition are containerized:

- [Hawk Agent](#)
- [Hawk Cluster Manager](#)
- [Hawk Console](#)

These components run as separate lightweight Docker containers. Like Hawk, these components can be configured as per your requirement. You can configure these components using the environment variables in a YAML file. For more information on the environment variables, see [Environment Variables for Hawk Container Edition Components](#).

## Hawk Agent

To monitor a system or application on the network using Hawk, install and run a Hawk Agent on the host machine. The Hawk Agent is a process that monitors activity on a particular machine by querying microagents.

For more information on the Hawk Agent, refer to the [TIBCO Hawk® Documentation](#).

In Hawk Container Edition, the Hawk Agent (`hkce_agent`) runs in a Docker container. This `hkce_agent` container monitors and manages all the containers, images and volumes on that Docker host. The `hkce_agent` container connects to Hawk Cluster Manager (`hkce_clustermanager`) containers to form a cluster using the TCP Transport for Hawk.

Each Hawk Agent can be configured using the environment variables, see [Environment Variables for Hawk Container Edition Components](#).

## Hawk Container Edition Microagents

Hawk agent in Hawk Container Edition comprises some microagent from Hawk and some additional microagents. The Hawk agent uses these microagents to collect information and operate using that information.

### Hawk Microagents

In Hawk Container Edition, the Hawk agent has the following microagents from Hawk:

#### Self

The Self microagent gathers version information for the local agent, the security policy in effect, and available microagents.

#### SysInfo

The SysInfo microagent gathers basic information on the operating system, hardware architecture, computer name and IP address of the agent container.

#### RuleBaseEngine

The RuleBaseEngine microagent gathers information about the rulebases and takes actions on rulebases that affect the Hawk agent.

#### TcpClusterStatus

The TcpClusterStatus microagent provides methods to monitor the health of the TCP transport cluster and TCP daemons.

#### TcpMessaging

The TcpMessaging microagent provides methods to send and receive messages by using the TCP Transport for Hawk.

#### Custom

The Custom microagent provides methods to run programs and scripts in Hawk Container Edition.

## Additional Hawk Container Edition Microagents

In addition to the microagents from Hawk, the Hawk Container Edition software provides the following microagents for container environments.

### The DockerHostMA Microagent

The DockerHostMA microagent is available on each Docker host as a part of the Hawk agent. It monitors and manages all containers running on that particular Docker host. Some of the key features of the DockerHostMA microagent are as follows:

- The microagent communicates with the Docker host using the Docker client API. For more information about Docker client API, see [Docker Client User Manual](#).
- The microagent contains methods to perform the following functions:
  - Monitoring container statistics (CPU, memory, network I/O)
  - Managing containers (starting, stopping, pausing)
  - Monitoring container logs

### The UniversalCollectorMicroAgent Microagent

The Universal Collector microagent enables you to monitor the log files, Docker container logs, and Hawk rulebases. You can also forward these logs and rulebase data to TIBCO LogLogic® Log Management Intelligence (LMI) or another Syslog. LogLogic® LMI is an log management solution that you can use to consume and archive all the log data. You can also implement the method to send logs to LogLogic LMI as an action in the rulebase. The microagent uses collectors and forwarders to distribute the data from different sources to the specified destinations. To forward the data, the microagent uses ULDP for LogLogic LMI and TCP/TLS syslog for other Syslog consumers.

For details about LogLogic LMI, see [TIBCO LogLogic® Log Management Intelligence \(LMI\) documentation](#).

All the forwarder and collector configurations created using the UniversalCollectorMicroAgent methods are stored in a YAML file (`universalcollector.yaml`). This file is generated at the location set by the `hma_plugin_dir` environment variable. The configuration file created in a container has a short life. Thus, removing the `hkce_agent` container might lead to loss of forwarder and collector configurations that were stored in the `universalcollector.yaml` file. To avoid this issue, you must map a Docker volume to the `hma_plugin_dir` environment variable.

For more information about methods of these microagents, see *TIBCO Hawk® Container Edition Microagent Reference Guide* [TIBCO Hawk® Container Edition Microagent Reference Guide](#).

## Hawk Cluster Manager

The Cluster Manager in Hawk, which is the seed node for the cluster, is termed as Hawk Cluster Manager in Hawk Container Edition.

In Hawk Container Edition, the Hawk Cluster Manager is containerized in the `hkce_clustermanager` Docker container. Similar to the Cluster Manager of Hawk, the `hkce_clustermanager` container is the seed node for the cluster. In this cluster, the Hawk agent containers connect to the Hawk Cluster Manager container and communicate using the TCP Transport for TIBCO Hawk.

You can configure the Hawk Cluster Manager using the environment variables. For details, see [Environment Variables for Hawk Container Edition Components](#).

For fault tolerant mode, you can start multiple Hawk Cluster Manager containers. These multiple Hawk Cluster Manager containers can be configured as a comma-separated list in the `tcp_daemon_url` environment variable.

## Hawk Console

Hawk Console is a web application that provides a central view of all the distributed components interacting within Hawk system. It is a pictorial view of each of the infrastructure component that Hawk components monitor. The Hawk Console must be run in a Docker container. This Hawk Console container (hkce\_console) connects with the Hawk Cluster Manager container and communicates with the Hawk agent using the TCP Transport for TIBCO Hawk.

In Hawk Container Edition, you can use REST APIs to build a custom Hawk Console. You can view these REST APIs at the Swagger page URL `http://<Console_host_IP>:<Host_port>/HawkConsole/v1/docs` after you start the Hawk Console container.

For more information about the Hawk Console, see [TIBCO Hawk® Documentation](#).

# Dockerize Hawk Container Edition

To run the Hawk Container Edition in Docker, you must build images of its components and run those Docker images inside Docker containers.

Hawk Container Edition does not have any TIBCO Universal Installer. It only contains Dockerfiles with preloaded settings for creating Docker image of each component. After Docker image of each component is created, you can run it in Docker container to monitor and manage your system.

For details on Dockerizing Hawk Container Edition, see:

- [Dockerfiles for Hawk Container Edition](#)
- [Building Hawk Container Edition Components Docker Images](#)
- [Running Hawk Container Edition Docker Containers in Standalone Mode](#)
- [Running Hawk Container Edition on AWS Based Kubernetes Cluster](#)

## Dockerfiles for Hawk Container Edition

Hawk Container Edition provides Dockerfiles for installation of Hawk Container Edition components inside the Docker containers.

Dockerfile is a script, which consists of various commands to automatically perform actions on a base image to create a new one. Dockerfiles simplify the process of deployment. The default base image on the provided Dockerfiles is Debian Linux with OpenJDK 11.

### Dockerfile Key Commands

The Dockerfiles begin with FROM command which specifies the image that starts the build process. The default base image on the provided Dockerfile is Debian Linux with OpenJDK 11. For example,

```
FROM openjdk:11.0.1-jre-slim-stretch
```

After specifying the base image, define various other methods, commands and arguments (or conditions), in return, provide a new image which is to be used for creating Docker containers. The Dockerfile is then supplied to the Docker daemon to build an image.

Syntax of the Dockerfile command is:

```
Command argument argument ...
```

For example,

```
# Print "Hello docker!"
RUN echo "Hello docker!"
```

For more details on each command of Dockerfile, refer to the Docker documentation at <https://docs.docker.com/engine/reference/builder/>.

### Sample Hawk Container Edition Dockerfiles

The following Dockerfiles are provided with the Hawk Container Edition installation:

- `hkce_agent_Dockerfile` - The `hkce_agent_Dockerfile` contains the commands to create Docker image of the Hawk Agent.
- `hkce_clustermanager_Dockerfile` - The `hkce_clustermanager_Dockerfile` contains the commands to create Docker image of the Hawk Cluster Manager.
- `hkce_console_Dockerfile` - The `hkce_console_Dockerfile` contains the commands to create Docker image of the Hawk Console.

The sample Dockerfile (`hkce_clustermanager_Dockerfile`) for Hawk Cluster Manager provided with the Hawk Container Edition is as follows:

```
FROM openjdk:11.0.1-jre-slim-stretch
COPY ./hkce/2.0/bin/startclustermanager.sh tibco.home/hkce/2.0/bin/
```

```
COPY ./hkce/2.0/lib/common-core tibco.home/hkce/2.0/lib/
COPY ./hkce/2.0/lib/common-ext tibco.home/hkce/2.0/lib/ext
WORKDIR tibco.home/hkce/2.0/bin/
ENTRYPOINT ["bash", "startclustermanager.sh"]
```

## Building Hawk Container Edition Components Docker Images

Before you can run TIBCO Hawk Container Edition components, you must create Docker images for those components.



The default base image in the provided sample Dockerfile is Debian Linux with OpenJDK 11. If required, you can change the base image reference in the Dockerfile. For information on Dockerfile, see <https://docs.docker.com/engine/reference/builder/>.

### Prerequisites

- Install Docker on the machine and perform the initial setup based on your operating system. For complete details on Docker installation, refer to the Docker documentation at <https://docs.docker.com>.
- Download Hawk Container Edition software package from the TIBCO Software Product Download Site (<https://edelivery.tibco.com/>). Extract the Hawk Container Edition archive file to a temporary directory on the machine.
- Ensure that you have Dockerfile for each components for which you are creating the Docker image.

### Procedure

1. Open a console window and navigate to the temporary directory where you extracted the Hawk Container Edition archive file at `<TEMP_DIRECTORY>/hkce-runtime-<version>/tibco.home/hkce/<version>/docker`.
2. Run the script `build_hkce_clustermanager.sh`. The script uses the `hkce_clustermanager_Dockerfile` to build the Docker image of Hawk Cluster Manager.
3. Run the script `build_hkce_agent.sh`. The script uses the `hkce_agent_Dockerfile` to build the Docker image of Hawk Agent.
4. Run the script `build_hkce_console.sh`. The script uses the `hkce_console_Dockerfile` to build the Docker image of Hawk Console.
5. Run the following command to verify if the Docker images are built:

```
docker images
```

### Result

The `hkce_clustermanager:2.0`, `hkce_agent:2.0`, and `hkce_console:2.0` Docker images are built.

### What to do next

After building the Docker images, you can run the Hawk containers of these images. You can run the Hawk component in two different modes:

- In standalone mode, see [Running Hawk Container Edition Docker Containers in Standalone Mode](#).
- In multi-host environment, see [Running Hawk Container Edition on AWS Based Kubernetes Cluster](#).

## Running Hawk Container Edition Docker Containers in Standalone Mode

The containers of Hawk Container Edition components communicate with each other using the TCP Transport for Hawk.

To create a Hawk TCP cluster, Hawk Agent and Hawk Console connect to the Hawk Cluster Manager, which is the seed of the cluster. The connection configuration for these three components can be done using the environment variables for them. For more information on the environment variables available for each component, see [Environment Variables for Hawk Container Edition Components](#). You can either provide these environment variables in Dockerfile or you can supply them in a single YAML file.

Docker provides a Compose tool for defining and running multi-container Docker applications. With the Compose tool you can provide all the configurations for all your containers in a single YAML file (docker-compose.yml). Then, using only a single command you can start the containers with the specified configurations. For more information about the Docker Compose tool, see the [Docker Compose](#) documentation.

Following is the sample content of a docker-compose.yml file that you can use in Hawk Container Edition:

```
version: '3'
services:
  hawkclustermanager:
    image: hkce_clustermanager:2.0
    ports:
      - '2561:2561'
    environment:
      tcp_daemon_url: hawkclustermanager:2561
      tcp_self_url: hawkclustermanager:2561
      hawk_domain: tcpdomain
  hawkagent:
    image: hkce_agent:2.0
    ports:
      - '2551:2551'
      - '2571:2571'
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    environment:
      tcp_daemon_url: hawkclustermanager:2561
      tcp_self_url: hawkagent:2551
      hawk_domain: tcpdomain
      ami_tcp_session: hawkagent:2571
      DOCKER_HOST: unix:///var/run/docker.sock
      config_path: /tibco.home/hkce/2.0/config/
      hma_plugin_dir: /tibco.home/hkce/2.0/plugin/
  hawkconsole:
    image: hkce_console:2.0
    environment:
      tcp_daemon_url: hawkclustermanager:2561
      tcp_self_url: hawkconsole:2551
      hawk_domain: tcpdomain
      hawk_console_repository_path: /tibco.home/hkce/2.0/repo
    ports:
      - '2589:2589'
      - '8083:8083'
```

### Procedure

1. Create the docker-compose.yml file with required configurations in a temporary folder.  
For more information about the Docker Compose tool, see the [Docker Compose](#) documentation.
2. In the command line, browse to the docker-compose.yml file and run the following command to run all Hawk component containers with specified configurations:

```
docker-compose up -d
```

3. You can verify that all containers are running by using the following command:

```
docker ps
```



## What to do next

If you have Hawk console container running, you can access it at `http://<Console_host_IP>:<Host_port>/HawkConsole`.

## Viewing Container Logs

All component containers of Hawk Container Edition publish their logs on `stdout`.

### Procedure

- To view logs of a particular container, run the following command:

```
docker logs <container_id>
```

## Environment Variables for Hawk Container Edition Components

Each component of Hawk Container Edition can be configured using the environment variables. These environment variables can be provided in a YAML file. Supply this YAML file to the Docker compose utility to run the component containers with these configurations.

### JMX Connectivity for Hawk Container Edition Containers

All three Hawk Container Edition containers can be configured to expose JMX port. This configuration can be done using environment variable `JAVA_OPTS`.

For example,

```
JAVA_OPTS="-Dcom.sun.management.jmxremote=true
-Dcom.sun.management.jmxremote.local.only=false
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false
-Djava.rmi.server.hostname=<DOCKER_HOST_IP>
-Dcom.sun.management.jmxremote.port=9999
-Dcom.sun.management.jmxremote.rmi.port=9999"
```



`-Dcom.sun.management.jmxremote.port` and `-Dcom.sun.management.jmxremote.rmi.port` must be published to Docker host using `-p`. Also, use different ports for each container.

### Encrypted Passwords

You can use simple text or encrypted passwords in the password related environment variables. Hawk Container Edition provides a utility (`tibhawkpassword`) to encrypt your password. The `tibhawkpassword` utility is located at `/tibco.home/hkce/2.0/bin`.

The syntax of the command is as follows:

```
tibhawkpassword -encrypt
```


The utility prompts you to enter the password that you want to encrypt. After encryption, you can use the encrypted password in the YML files by enclosing the password in single quotes. The encrypted password starts with the hash (`#`) symbol. Thus, without the single quotes, the YML file parser interprets encrypted password as comment.

You can use the simple text or encrypted passwords in the following environment variables:

- `tcp_key_store_password`
- `tcp_key_password`
- `tcp_trust_store_password`
- `email_smtp_password`
- `hawk_console_ssl_password`

- hawk\_console\_ssl\_key\_store\_password

### *The Hawk Agent (hkce\_agent) Environment Variables*

Environment Variable	Mandatory	Default Value	Description
agent_domain	No	"none"	The agent_domain environment variable sets the Hawk agent domain.
agent_name	No	Hostname of hkce_agent container	The agent_name environment variable sets the name of Hawk agent. If not provided then set it to the hostname of the hkce_agent container.
auto_config_dir	No	None	<p>The auto_config_dir environment variable specifies the directory from where the configuration objects are loaded for the agent to run in auto-configuration mode. When this option is not used, the agent operates in manual configuration mode. In case of the manual configuration mode, use the config_path variable.</p> <p> By default the auto_config_dir is created within the hkce_agent container. Since any file or folder created within the container has a transient nature, removing hkce_agent container might lead to loss of rulebases that were stored in the directory specified in auto_config_dir. Thus, to avoid this issue, use the Docker volume to persist the rulebases and set the auto_config_dir to the destination of the Docker volume within the hkce_agent container.</p>
config_path	No	None	<p>The config_path environment variable specifies the directory from where the configuration objects are loaded for the agent to run in manual configuration mode. This variable cannot be used with the auto_config_dir variable.</p> <p>The delimiter for path entries is the colon (:) symbol.</p>
hawk_domain	No	"default"	The hawk_domain environment variable sets the Hawk domain name.
hma_plugin_dir	No	-	The hma_plugin_dir environment variable specifies the directory used for Hawk microagent plug-in configuration.

Environment Variable	Mandatory	Default Value	Description
log_level	No	7	<p>The log_level environment variable identifies the log level. The values of the log_level environment variable are:</p> <ul style="list-style-type: none"> <li>• 4 (ERROR)</li> <li>• 6 (WARN)</li> <li>• 7 (INFO)</li> <li>• 8 (DEBUG)</li> <li>• 16 (TRACE)</li> </ul>
tcp_daemon_url	Yes	None	<p>The tcp_daemon_url environment variable specifies the daemon URL for TCP Transport for TIBCO Hawk. The URL is in the form <i>&lt;DAEMON_IP_ADDRESS_1&gt;: &lt;PORT&gt;, &lt;DAEMON_IP_ADDRESS_2&gt;: &lt;PORT&gt;</i>.</p>
tcp_self_url	Yes	None	<p>The tcp_self_url environment variable specifies the self URL for the TCP Transport for TIBCO Hawk. The URL is in the form <i>&lt;SELF_IP_ADDRESS&gt;: &lt;PORT&gt;</i>.</p>
<b>Email Configurations</b>			
email_smtp_server	No	None	<p>The email_smtp_server environment variable identifies the SMTP server host name for sending emails.</p>
email_smtp_port	No	25	<p>The email_smtp_port environment variable identifies the SMTP server port</p>
email_smtp_auth_required	No	false	<p>The email_smtp_auth_required environment variable specifies whether the SMTP server authentication is required or not</p>
email_smtp_tls_required	No	false	<p>The email_smtp_tls_required environment variable specifies whether the SMTP server requires TLS or not.</p>
email_smtp_socket_factory_port	No	25	<p>The email_smtp_socket_factory_port environment variable specifies the SMTP socketFactory port needed for TLS.</p>
email_smtp_user	No	None	<p>The email_smtp_user environment variable SMTP server user name. This variable is required only if SMTP server authentication is configured to true.</p>

Environment Variable	Mandatory	Default Value	Description
email_smtp_password	No	None	The email_smtp_password environment variable specifies the user password for the SMTP server.  This variable is required only if SMTP server authentication is configured to true.
<b>TCP Transport for TIBCO Hawk SSL Environment Variables</b>			
tcp_key_store	No	None	Path of the key store file
tcp_trust_store	No	None	Path of the trust store file
tcp_key_store_password	No	None	Password for the key store file
tcp_key_password	No	None	Encrypted key password
tcp_trust_store_password	No	None	Password for the trust store file
tcp_ssl_protocol	No	TLSv1.2	Protocol for a secure connection
tcp_enabled_algorithms	No	TLS_RSA_WITH_AES_128_CBC_SHA	Algorithm to be used for the security protocol. You can specify multiple algorithms as comma-separated list without space.

*The Hawk Cluster Manager (hkce\_clustermanager) Environment Variables*

Environment Variable	Mandatory	Default Value	Description
hawk_domain	No	"default"	The hawk_domain environment variable sets the Hawk domain name.
log_level	No	7	The log_level environment variable identifies the log level. The values of the log_level environment variable are: <ul style="list-style-type: none"> <li>• 4 (ERROR)</li> <li>• 6 (WARN)</li> <li>• 7 (INFO)</li> <li>• 8 (DEBUG)</li> <li>• 16 (TRACE)</li> </ul>

Environment Variable	Mandatory	Default Value	Description
tcp_daemon_url	Yes	None	The tcp_daemon_url environment variable specifies the daemon URL for TCP Transport for TIBCO Hawk. The URL is in the form <code>&lt;DAEMON_IP_ADDRESS_1&gt; : &lt;PORT&gt;</code> , <code>&lt;DAEMON_IP_ADDRESS_2&gt; : &lt;PORT&gt;</code> .
tcp_self_url	Yes	None	The tcp_self_url environment variable specifies the self URL for the TCP Transport for TIBCO Hawk. The URL is in the form <code>&lt;SELF_IP_ADDRESS&gt; : &lt;PORT&gt;</code> .
<b>TCP Transport for TIBCO Hawk SSL Environment Variables</b>			
tcp_key_store	No	None	Path of the key store file
tcp_trust_store	No	None	Path of the trust store file
tcp_key_store_password	No	None	Password for the key store file
tcp_key_password	No	None	Encrypted key password
tcp_trust_store_password	No	None	Password for the trust store file
tcp_ssl_protocol	No	TLSv1.2	Protocol for a secure connection
tcp_enabled_algorithms	No	TLS_RSA_WITH_AES_128_CBC_SHA	Algorithm to be used for the security protocol. You can specify multiple algorithms as comma-separated list without space.

*The Hawk Console (hkce\_console) Environment Variables*

Environment Variable	Mandatory	Default Value	Description
hawk_domain	No	Default	The hawk_domain environment variable sets the Hawk domain name.
tcp_self_url	Yes	None	The tcp_self_url environment variable specifies the self URL for the TCP Transport for TIBCO Hawk. The URL is in the form <code>&lt;SELF_IP_ADDRESS&gt; : &lt;PORT&gt;</code> .

Environment Variable	Mandatory	Default Value	Description
tcp_daemon_url	Yes	None	The tcp_daemon_url environment variable specifies the daemon URL for TCP Transport for TIBCO Hawk. The URL is in the form <code>&lt;DAEMON_IP_ADDRESS_1&gt; : &lt;PORT&gt;</code> , <code>&lt;DAEMON_IP_ADDRESS_2&gt; : &lt;PORT&gt;</code> .
hawk_console_repository_path	No	Current working directory of Hawk Console	Path of the repository configuration file in the Hawk Console container. Update to configuration objects of the repository is stored in the path specified.
hawk_console_retention_count_notification	No	100000	Retention limit for notifications. After the retention limit is reached, the notifications are purged. The purge rate is 25%.
hawk_console_retention_count_low_alerts	No	100000	Retention limit for high alerts. After the retention limit is reached, the high alerts are purged. The purge rate is 25%.
hawk_console_retention_count_medium_alerts	No	100000	Retention limit for medium alerts. After the retention limit is reached, the medium alerts are purged. The purge rate is 25%.
hawk_console_retention_count_high_alerts	No	100000	Retention limit for low alerts. After the retention limit is reached, the notifications are purged. The purge rate is 25%.
<b>TCP Transport for TIBCO Hawk SSL Environment Variables</b>			
tcp_key_store	No	None	Path of the key store file
tcp_trust_store	No	None	Path of the trust store file
tcp_key_store_password	No	None	Password for the key store file
tcp_key_password	No	None	Encrypted key password
tcp_trust_store_password	No	None	Password for the trust store file
tcp_ssl_protocol	No	TLSv1.2	Protocol for a secure connection

Environment Variable	Mandatory	Default Value	Description
tcp_enabled _algorithms	No	TLS_RSA_WITH_ AES_128_CBC_S HA	Algorithm to be used for the security protocol. You can specify multiple algorithms as comma-separated list without space.
<b>Hawk Console SSL Environment Variables</b>			
hawk_consol e_ssl_key_a lias		None	Key alias
hawk_consol e_ssl_passw ord		None	Encrypted key password
hawk_consol e_ssl_key_s tore		None	Path of the key store file
hawk_consol e_ssl_key_s tore_passwo rd		None	Password for the key store file
hawk_consol e_ssl_proto col		TLSv1.2	Protocol for a secure connection
hawk_consol e_ssl_ciphe rs		TLS_RSA_WITH_ AES_128_CBC_S HA	Algorithm to be used for the security protocol. You can specify multiple algorithms as comma-separated list without space.

# Running Hawk Container Edition on AWS Based Kubernetes Cluster

Kubernetes is an open source system for managing containerized applications across multiple hosts, providing basic mechanisms for deployment, maintenance, and scaling of applications.

For more information about Kubernetes, refer to the Kubernetes documentation at <https://kubernetes.io/docs/concepts/>.

## Prerequisites

Download and install following CLI tools on your system:

CLI	Download and Installation Instruction Link
kops	<a href="https://github.com/kubernetes/kops/blob/master/docs/aws.md">https://github.com/kubernetes/kops/blob/master/docs/aws.md</a>
kubectl	<a href="https://kubernetes.io/docs/tasks/tools/install-kubectl/">https://kubernetes.io/docs/tasks/tools/install-kubectl/</a>
aws	<a href="https://aws.amazon.com/cli/">https://aws.amazon.com/cli/</a>

## Procedure

1. Set up a Kubernetes cluster on Amazon Web Services (AWS).  
For more information, see [Setting up a Kubernetes Cluster on AWS](#).
2. Create Docker image of Hawk Container Edition components.  
For more information, see [Building Hawk Container Edition Components Docker Images](#).
3. Go to the EC2 Container Services dashboard and create a repository with the same name as the Docker image of Hawk Container Edition component. Upload the component image to the repository and for help you might use the View Push Commands button.



AWS Repository name must be the same as the Docker image name of Hawk Container Edition component. For more information on how to create a repository in Amazon AWS, refer to <https://docs.aws.amazon.com/AmazonECR/latest/userguide/repository-create.html>.

4. Deploy Hawk Container Edition components on AWS.  
For more information, see [Deploying Hawk Container Edition Containers on AWS Based Kubernetes Cluster](#).

## Setting up a Kubernetes Cluster on AWS

Set up a Kubernetes cluster with AWS for running Hawk Container Edition components.

### Procedure

1. Create an S3 storage to store the cluster configuration and state. You can use either AWS CLI or AWS console to create the storage.

The sample AWS CLI command for creating S3 storage is:

```
aws s3 mb s3://hkce-bucket
```

For more information about Amazon Simple Storage Service (Amazon S3), see [Amazon S3 Documentation](#).



2. Create the Kubernetes cluster on AWS using the following command:

```
kops create cluster --zones us-west-2a --master-zones us-west-2a --master-size t2.large --node-size t2.large --name hkcecluster.k8s.local --state s3://<s3-bucket-name> --yes
```

Where,

- `s3-bucket-name` is a name of the s3 bucket created earlier (`hkce-bucket`).
- `hkcecluster.k8s.local` is the name of the cluster being created. Use `k8s.local` prefix to identify a gossip-based Kubernetes cluster and you can skip the DNS configuration.

For more information about the `kops create cluster` command either use the `help` parameter or see [kops Documentation](#).

3. Validate your cluster using the `validate` command.

```
kops validate cluster
```

Node and master must be in ready state. The `kops` utility stores the connection information at `~/.kops/config`, and `kubectl` uses the connection information to connect to the cluster.

4. If needed, you can delete the cluster using the following command:

```
kops delete cluster hkcecluster.k8s.local --state=s3://<s3-bucket-name> --yes
```

## Deploying Hawk Container Edition Containers on AWS Based Kubernetes Cluster

Hawk Container Edition can be deployed on AWS using the configuration files (YAML format), which contain the configuration details for deployment including environment variables.

### Prerequisites

For deploying Hawk Container Edition cluster on AWS, you must first set up Kubernetes cluster on AWS and then upload your Docker image on AWS. For more information, see [Running Hawk Container Edition on AWS Based Kubernetes Cluster](#).

### Procedure

1. Create Kubernetes resources, required for deploying Hawk Container Edition cluster, using the YAML files. These resources include deployment and services for the cluster. Thus, to deploy a Hawk Container Edition cluster, create:

- a Hawk Cluster Manager node (pod) to start the cluster
- a service to connect to Hawk Cluster Manager node
- a Hawk agent node which connects to the Hawk Cluster Manager node service
- a Hawk console node which connects to the Hawk Cluster Manager node service

For more information on the YAML files configurations of Hawk Container Edition components, see [Hawk Container Edition Components YAML Files](#).

2. Run the `create` command of `kubectl` utility by using the YAML files to deploy the Hawk Container Edition cluster.

```
kubectl create -f <component_file>.yaml
```

For example, the following are the YAML files for Hawk Container Edition components:

- `daemonstateful.yaml` - Hawk Cluster Manager
- `agentdaemonset.yaml` - Hawk Agent
- `consolepod.yaml` - Hawk Console

Run the `kubectl create` command to deploy Hawk Container Edition cluster:

```
kubectl create -f daemonstateful.yml
kubectl create -f agentdaemonset.yml
kubectl create -f consolepod.yml
```

3. You can also get the external IP to the external service of the cluster by using the `get services` command. You can then use that IP to connect to the cluster.

For example:

```
kubectl get services hkce-console-service
```

4. You can check the logs of individual Hawk component container pods using the following command:

```
kubectl logs <pod>
```

## Hawk Container Edition Components YAML Files

As per their requirement, each component (Hawk agent, Hawk Cluster Manager, and Hawk Console) of Hawk Container Edition uses different strategy for deployment.

### Hawk Cluster Manager YAML File for AWS

The Hawk Cluster Manager (`hkce_clustermanager`) is a seed node to which all other containers connect. Thus, this container should be reachable from all the containers. Also the network information should not be lost even when it is restarted or rescheduled.

Use the Kubernetes *StatefulSets* for deploying Hawk Cluster Manager. Like a Deployment, a StatefulSet manages Pods that are based on an identical container specifications. Unlike a Deployment, a StatefulSet maintains a sticky identity for each of their Pods. For details about Kubernetes StatefulSets, see the [Kubernetes documentation](#). Also, create a headless service so that all other containers can reach Hawk Cluster Manager container. In a headless service you specify the `clusterIP` as `None`. For details about headless service, see to the [Kubernetes documentation](#). For the Pods created by StatefulSets with headless service, DNS entries are created in the form `<pod-name>.<headless-service-name>`.

For example, for a StatefulSet with *N* replicas, each Pod in the StatefulSet is assigned an integer ordinal, from 0 up through *N*-1, that is unique over the Set. So if your StatefulSets has a name "my-stateful-set" with 3 replicas, it creates three pods with names `my-stateful-set-0`, `my-stateful-set-1` and `my-stateful-set-2`. Also, if you create a headless service with the name "my-service", then you can connect to these StatefulSets pods using the host name `my-stateful-set-0.my-service`, `my-stateful-set-1.my-service`, and so on.

For the Hawk Cluster Manager, create only one `hkce_clustermanager` pod using StatefulSets with the name `hkce-clustermanager-set` and headless service with the name `hkce-clustermanager-service`. Thus, other components can connect to this pod using the hostname `hkce-clustermanager-set-0.hkce-clustermanager-service`. Set the `clusterIP` as `None` for the headless service.

Set up the container (the `containers` tag) with the Docker image (`image`) of the Hawk Cluster Manager.

### Sample Content for Hawk Cluster Manager YAML File

```
apiVersion: v1
kind: Service
metadata:
  name: hkce-service
spec:
  ports:
    - port: 2561
      protocol: TCP
      targetPort: 2561
  selector:
    app: hkce-clustermanager
    clusterIP: None
---
apiVersion: apps/v1
kind: StatefulSet
```

```

metadata:
  labels:
    app: hkce-clustermanager
    name: hkce-clustermanager-set
spec:
  serviceName: hkce-service
  selector:
    matchLabels:
      app: hkce-clustermanager
  template:
    metadata:
      labels:
        app: hkce-clustermanager
    spec:
      containers:
        - name: hkce-clustermanager
          image: <aws_docker_registry>/hkce_clustermanager:2.0
          imagePullPolicy: Always
          env:
            - name: tcp_self_url
              value: hkce-clustermanager-set-0.hkce-service:2561
            - name: tcp_daemon_url
              value: hkce-clustermanager-set-0.hkce-service:2561
          ports:
            - containerPort: 2561
              protocol: TCP

```

### Hawk Agent YAML File for AWS Deployment

Hawk agent must be deployed on all nodes. Thus, for Hawk agent the resource type should be DaemonSet. A DaemonSet ensures that all (or some) nodes run a copy of a Pod.

Set up the container (the containers tag) with the Docker image (image) of the Hawk agent. The `tcp_self_url` should be set to the hostname of the pod in which Hawk agent container is running. You can use the Kubernetes downward API to get the hostname of the pod. For this set `fieldPath` to `status.podIP`. This ensures that each `hkce_agent` has a unique `tcp_self_url`. Also the `ami_tcp_session` for each agent should be set to the node IP on which this `hkce_agent` container is deployed. You can use the Kubernetes downward API to get the hostname of the node. For this set `fieldPath` to `status.hostIP`. The AMI applications running on that node can use same `status.hostIP` in their `daemon_url`. Specify the protocol and port to connect to this service.

### Sample Content for Hawk Agent YAML File

```

apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: hkce-agent-set
  labels:
    app: hkce-agent
spec:
  selector:
    matchLabels:
      name: hkce-agent-set
  template:
    metadata:
      labels:
        name: hkce-agent-set
    spec:
      containers:
        - name: hkce-agent
          image: <aws_docker_registry>/hkce_agent:2.0
          imagePullPolicy: Always
          env:
            - name: HOST_NAME
              valueFrom:
                fieldRef:
                  apiVersion: v1
                  fieldPath: status.podIP
            - name: HOST_IP
              valueFrom:

```

```

      fieldRef:
        fieldPath: status.hostIP
    - name: tcp_self_url
      value: ${HOST_NAME}:2551
    - name: tcp_daemon_url
      value: hkce-clustermanager-set-0.hkce-service:2561
    - name: ami_tcp_session
      value: ${HOST_IP}:2571
    - name: config_path
      value: /tibco.home/hkce/2.0/config/
    - name: DOCKER_HOST
      value: unix:///var/run/docker.sock
  volumeMounts:
    - mountPath: /var/run/docker.sock
      name: docker-sock-volume
  ports:
    - containerPort: 2551
      name: agentport
      protocol: TCP
    - containerPort: 2571
      hostPort: 2571
      protocol: TCP
  volumes:
    - name: docker-sock-volume
      hostPath:
        path: /var/run/docker.sock

```

### Hawk Console YAML File for AWS Deployment

Setup the a pod with load-balancer service for deploying Hawk Console. The load balancer provides an externally-accessible IP address that sends traffic to the correct port on your cluster nodes.

The load-balancer service would enable us to access the console 8083 port from outside. Set up the container ( the containers tag) with the Docker image (the image tag) of the Hawk Console. Also like Hawk agent, the Hawk Console uses the Kubernetes downward API to access hostname of the pod. For this set fieldPath of HOST\_NAME to status.podIP and use this in the tcp\_self\_url.

### Sample Content for Hawk Console YAML File

```

apiVersion: v1
kind: Service
metadata:
  name: hkce-console-service
spec:
  type: LoadBalancer
  ports:
    - port: 8083
      targetPort: 8083
  selector:
    app: hkce-console
---
apiVersion: v1
kind: Pod
metadata:
  name: hkce-console
  labels:
    name: hkce-console
    app: hkce-console
spec:
  containers:
    - name: hkce-console
      image: <aws_docker_registry>/hkce_console:2.0
      imagePullPolicy: Always
      env:
        - name: HOST_NAME
          valueFrom:
            fieldRef:
              apiVersion: v1
              fieldPath: status.podIP
        - name: tcp_self_url

```

```
value: ${HOST_NAME}:2551
- name: tcp_daemon_url
  value: hkce-clustermanager-set-0.hkce-service:2561
- name: hawk_domain
  value: default
- name: hawk_console_repository_path
  value: /tibco.home/hkce/2.0/repo
ports:
- containerPort: 2551
  name: consoleport
  protocol: TCP
- containerPort: 8083
```

# Rulebase Repository Management in Hawk Console

---

The Hawk Console contains a rulebase repository that stores configuration objects and then distributes and deploys them to Hawk agents. The configuration objects include rulebases, schedules, and rulebase map. You can create new objects or update existing objects in the repository. Hawk Console deploys the configuration objects from the repository to Hawk agents when they start.

## Rulebase Mapping

A rulebase mapping defines a mapping between rulebases and Hawk agents. The Hawk Console deploys the mapped rulebases to the Hawk agent during start. The mapping can be between a rulebase and Hawk agent, or between a rulebase and a group of Hawk agents. You can create a rulebase mapping by using the Hawk Console.

For details about creating a rulebase mapping, see [Creating Rulebase Mapping](#).

## Agent Groups

The Hawk agents that have similar rulebase needs are grouped together in an agent group. The agent groups are of two types: system-defined and user-defined.

### System-defined groups

Hawk Container Edition automatically creates the following agent groups:

- **Operating system group** - Hawk Container Edition groups Hawk agents based on their operating system. The name of agent groups starts with "++" symbol followed by the operating system name. For example, all the Hawk agent running on the Solaris operating system are part of the ++Solaris group.
- **All group** - Hawk Container Edition groups all registered Hawk agents into one group. The name of the agent group is ++.

### User-defined groups

You can group any Hawk agent into a group. The name of the group starts with the '+' symbol.

For details about creating an agent group, see [Creating Agent Groups](#).

## Rulebases and Schedules

In the rulebase repository, rulebase and schedule in a rulebase repository works in the same way as they work in Hawk agents. For details about rulebases and schedules in Hawk Console, see *TIBCO Hawk Console User's Guide* in [TIBCO Hawk](#) documentation.

You can also use existing rulebases and schedules from Hawk agents and store them in a rulebase repository. For details, see [Migrating Rulebases and Schedules from Hawk Agent to Rulebase Repository](#) on page 32.

Also, for details about the effect of actions performed on rulebases and schedules in the rulebase repository, see [Actions on Rulebase Repository Configuration Objects](#).

## Rulebase Repository Configuration

You must configure the repository in the Hawk Console by using the `hawk_console_repository_path` environment variable. It specifies the path of repository inside the `hawk_console` container. Hawk Console loads and saves the configuration objects in the repository at `<hawk_console_repository_path>/<domain_name>`.



The `hkce_console` container hosts the rulebase repository and removing the container results in loss of configuration object of the repository. To avoid this, you must use the Docker volume to persist the rulebase repository. Also, you must set the `hawk_console_repository_path` environment variable to destination of the Docker volume in the `hkce_console` container.

### Rulebase Repository Actions

Hawk Console updates the configuration objects based on the action performed on the configuration objects.

For details about the effect of actions performed in repository, see [Actions on Rulebase Repository Configuration Objects](#).

## Creating Agent Groups

The Hawk agents that have similar rulebase needs are grouped together in an agent group. You can create an agent group by using the Hawk Console.

### Procedure

1. In Hawk Console, open the domain for which you have to create an agent group.
2. On the Domain page, click **Rulebase Repository**.
3. On the Rulebase Repository page, select the **Groups** tab.
4. On the **Groups** tab, click the **Add Group** icon .
5. Enter the name of the new agent group and click **OK**.
6. From the **Available Members** list, click the **Add Members** icon for the Hawk agent that you want to add to the agent group.
7. Verify the **Members Mapped** list for the agent group and click **Save Mapping**.

### What to do next

You can now map the rulebase to this agent group. For more details about rulebase mapping, see [Creating Rulebase Mapping](#).

## Creating Rulebase Mapping

A rulebase mapping defines a mapping between rulebases and Hawk agents. You can create a rulebase mapping by using the Hawk Console.

### Procedure


1. In Hawk Console, open the domain for which you have to create a rulebase mapping.
2. On the Domain page, click **Rulebase Repository**.
3. On the Rulebase Repository page, select the **Rulebase-Map** tab.
4. On the **Rulebases-Map** tab, select a rulebase which you want to map to Hawk agents or groups.
5. From the **Available Members** list, click the **Add Members** icon for the group or Hawk agent that you want to map to the rulebase.
6. Verify the **Members Mapped** list for the rulebase and click **Save Mapping**.

## Migrating Rulebases and Schedules from Hawk Agent to Rulebase Repository

For the optimized use of the rulebase repository, you must migrate existing rulebases and schedules from Hawk agents to the rulebase repository. In Hawk Console, you can create a rulebase mapping for these migrated rulebases. Based on these rulebase mapping, the rulebase repository deploys the rulebases and schedules to Hawk agents.

### Procedure

- You can migrate the rulebases and schedules from Hawk agents to a rulebase repository by following either of these procedures:

Scenario	Steps
<b>Move each rulebase one by one</b>	<ol style="list-style-type: none"> <li>1. In Hawk Console, open the Hawk Agent page from which you want to migrate the rulebase.</li> <li>2. On the Agent page, select the <b>Rulebases</b> tab.</li> <li>3. On the <b>Rulebases</b> tab, from the rulebases list, under the <b>Actions</b> column, select the <b>Send to Repository</b> option for the rulebase that you want to migrate.</li> <li>4. Click <b>Yes</b> to confirm the migration.</li> </ol> <p>For more details about the Hawk Console, see <a href="#">TIBCO Hawk® Documentation</a></p>
<b>Move each schedule one by one</b>	<ol style="list-style-type: none"> <li>1. In Hawk Console, open the Hawk Agent page from which you want to migrate the schedule.</li> <li>2. On the Agent page, select the <b>Schedules</b> tab.</li> <li>3. On the <b>Schedules</b> tab, select the schedule that you want to migrate.</li> <li>4. Click the <b>Send to Repository</b> icon .</li> <li>5. Click <b>Yes</b> to confirm the migration.</li> </ol> <p>For more details about the Hawk Console, see <a href="#">TIBCO Hawk® Documentation</a></p>
<b>Move rulebases and schedules in bulk</b>	<ol style="list-style-type: none"> <li>1. Copy all the rulebase files (.hrb) and schedule files (.hsf) from your Hawk agent to the domain folder in the Hawk Console repository path. The repository path is specified by the hawk_console_repository_path environment variable. Thus, the path to copy the rulebase and schedule files is <code>&lt;hawk_console_repository_path&gt;/&lt;domain_name&gt;</code>.</li> <li>2. Start the Hawk Console to load these rulebases and schedules in the rulebase repository.</li> </ol>

### Result

The rulebase repository in Hawk Console, lists all the migrated rulebases and schedules. You can then perform different operations on these rulebases and schedules.



## Actions on Rulebase Repository Configuration Objects

Hawk Console contains a rulebase repository that stores configuration objects and then distributes and deploys them to Hawk agents. The configuration objects include rulebases, schedules, and rulebase map.

### Rulebase Repository Configuration Objects Files

You must configure the repository in Hawk Console by using the `hawk_console_repository_path` environment variable. Hawk Console loads and saves configuration objects in the repository at `<hawk_console_repository_path>/< domain_name>`. Hawk Console stores the configuration objects in the following files:

Configuration Object	File Extension
Rulebase mapping and agent group mapping	.hrm The default file is <code>rbmap.hrm</code> .
Rulebase	.hrb
Schedule	.hsf The default file is <code>schedules.hsf</code> .

### Hawk Console Actions

The following table lists the effect of the action performed on the Hawk Console to the configuration objects:

#### *Effects of Actions in Rulebase Repository*

Event / Action	Effect on Rulebase Map	Effect on Rulebase	Effect on Schedule
Hawk Console starts up	Hawk Console loads rulebase mapping and agent groups from the <code>rbmap.hrm</code> file present in the rulebase repository.	Hawk Console loads all rulebase files from the rulebase repository.	Hawk Console loads schedules from the <code>schedules.hsf</code> file present in the rulebase repository.
Hawk agent starts up	If the agent group for Hawk agent operating system is not present, Hawk Console adds a new agent group to the list of operating system groups of the rulebase repository.	Hawk Console deploys rulebases to the respective Hawk agent based on the rulebase mapping.	Hawk Console deploys all the schedules from the rulebase repository to the Hawk agent.

Event / Action	Effect on Rulebase Map	Effect on Rulebase	Effect on Schedule
Create and save a new rulebase or schedule	Not applicable	<p>The Hawk Console adds the new rulebase to the rulebase repository and saves it to a .hrb file.</p> <p><b>What to do next:</b> To deploy the new rulebase to Hawk agents, you must create a rulebase mapping for it, see <a href="#">Creating Rulebase Mapping</a>.</p>	<p>Hawk Console adds a new schedule to the rulebase repository and saves it to the schedule.hsf file.</p> <p><b>What to do next:</b> To deploy this new schedule to all Hawk agents in the domain, on the <b>Schedules</b> tab, click the <b>Deploy Schedule</b> icon and confirm the action.</p>
Update and save an existing rulebase or schedule	Not applicable	<p>The Hawk Console updates the rulebase in the rulebase repository and saves the update to the respective .hrb file.</p> <p><b>What to do next:</b> To deploy the updated rulebase to all mapped Hawk agents, on the <b>Rulebase Mapping</b> tab, click <b>Save Mapping</b> for the rulebase.</p>	<p>Hawk Console updates the schedule in the rulebase repository and saves the update to the schedule.hsf file.</p> <p><b>What to do next:</b> To deploy this updated schedule to all Hawk agents in the domain, on the <b>Schedules</b> tab, click the <b>Deploy Schedule</b> icon and confirm the action.</p>
Delete a rulebase	Hawk Console removes all the rulebase mappings for the rulebase and updates the rbmap.hrm file.	Hawk Console deletes the rulebase file (.hrb) from the rulebase repository. If the deleted rulebase was mapped to Hawk agents or agent groups, then these mapped rulebases are undeployed from Hawk agents after they are restarted.	Not applicable

Event / Action	Effect on Rulebase Map	Effect on Rulebase	Effect on Schedule
Delete a schedule	Not applicable	Not applicable	Hawk Console deletes the schedule from the rulebase repository and the <code>schedule.hsf</code> file. If the deleted schedule was deployed to Hawk agents or agent groups, then these schedules are undeployed from Hawk agents after they are restarted.
Create and save a rulebase mapping	Hawk Console adds rulebase mapping to the rulebase repository and updates the <code>rbmap.hrm</code> file.	Hawk Console deploys the rulebase to the mapped Hawk agents and agent groups.	Not applicable
Update and save existing rulebase mapping	Hawk Console updates rulebase mapping in the rulebase repository and updates the <code>rbmap.hrm</code> file.	<p>If Hawk agents or agent groups are added to the rulebase mapping, Hawk Console deploys the rulebase to new members.</p> <p>If Hawk agents or agent groups are removed from the rulebase mapping, Hawk Console undeploys the respective rulebase from those Hawk agents or members of agent group.</p>	Not applicable
Create a new agent group	<p>Hawk Console creates an agent group in memory only.</p> <p><b>What to do next:</b> To save the agent group information in the <code>rbmap.hrm</code> file, either add Hawk agents to the agent group or map the group to a rulebase.</p> <p>For details, see <a href="#">Creating Agent Groups</a> and <a href="#">Creating Rulebase Mapping</a>.</p>	Not applicable	Not applicable

Event / Action	Effect on Rulebase Map	Effect on Rulebase	Effect on Schedule
Add new Hawk agents to the agent group and save the agent group mapping	Hawk Console updates the group mapping information in the <code>rbmap.hrm</code> file in the rulebase repository.	<p>If the agent group is already mapped to rulebases, then these mapped rulebases are not automatically deployed to new agents.</p> <p><b>What to do next:</b> To deploy the mapped rulebase to new Hawk agents, on the <b>Rulebase Mapping</b> tab, click <b>Save Mapping</b> for the rulebases that are mapped to the updated agent group.</p>	Not applicable
Remove Hawk agents from the agent group and save the agent group mapping	Hawk Console updates the group mapping information in the <code>rbmap.hrm</code> file in the rulebase repository.	<p>If the agent group is already mapped to rulebases, then these mapped rulebases are not automatically undeployed from the removed Hawk agents.</p> <p><b>What to do next:</b> To undeploy the mapped rulebases from the removed Hawk agents, on the <b>Rulebase Mapping</b> tab, click <b>Save Mapping</b> for those mapped rulebases.</p>	Not applicable
Delete an agent group	Hawk Console deletes the agent group from the rulebase repository and updates the <code>rbmap.hrm</code> file.	If the deleted agent group was mapped to rulebases, then these mapped rulebases are undeployed from Hawk agents after they are restarted.	Not applicable

## Adding Custom Hawk Plug-Ins to the Hawk Agent

Hawk plug-in is a Hawk Microagent that resides within the process space of a Hawk agent. These Hawk plug-ins communicate with a host of third-party applications and use the protocols of the third-party applications to monitor and manage them.

You can create your custom Hawk plug-ins and add them to the Hawk agent container. Every plug-in requires a .hma file and a .jar file. The .hma file is a Hawk Microagent configuration file, and the .jar file contains Java implementation of the methods that are exposed through the Hawk subsystem.

### Procedure

1. Create a Java implementation of methods (.jar) that you want for the plug-in.  
For more information, see the `<TEMP_DIRECTORY>/tibco.home/hkce/<version>/examples/ma_plugin/SamplePluginMicroagent.java` file.
2. Create the .hma file which defines the custom Hawk plug-in. Refer to the `<TEMP_DIRECTORY>/tibco.home/hkce/<version>/examples/ma_plugin/SamplePluginMicroagent.hma` file for guidance. Following are the main components of the HMA file:
  - a) Most important constituent of this .hma file is the startup class of the plug-in implementation mentioned under the `<classname>` tag.  
**Example:** `<classname>com.A.B.myPluginControllerClass</classname>`.
  - b) You can also specify some optional arguments that you can provide externally without changing the plug-in implementation. Such optional arguments should be mentioned under the `<arguments>` tag.  
**Example:** `<arguments><arg>-traceDir</arg><arg>C:/LogDir</arg></arguments>`.
  - c) Absolute path of the implementation .jar files and all the required third-party libraries should be mentioned under `<classpath>` tag. If both of the .hma and .jar files are in same folder, then simple .jar file name is provided in the `<classpath>` tag.  
**Example:** `<classpath><path>C:/TPCL/libs/slf4j-api-1.6.4.jar</path></classpath>`.
3. Place the .hma and .jar files of your Hawk plug-in in the folder `<TEMP_DIRECTORY>/tibco.home/hkce/<version>/plugin`.
4. Rebuild the Hawk agent Docker image by running the script `<TEMP_DIRECTORY>/tibco.home/hkce/<version>/docker/build_hkce_agent.sh`.
5. Run the containers. While running the containers, ensure that the `hma_plugin_dir` environment variable of `hkce_agent` is set to value `/tibco.home/hkce/<version>/plugin/`.
6. If Hawk Console is running, you can confirm if your plug-in is loaded in Hawk agent by checking the list of microagents in the Hawk console. Check if the methods of the plug-in microagent are listed and if you can invoke and subscribe them.

# Hawk Container Edition Programming

---

The Hawk Container Edition software, similar to Hawk, provides APIs to interact with Hawk applications.

You can use the following APIs in Hawk Container Edition:

## Console API

The Console API is a comprehensive set of Java interfaces that allow you to manage and interact with Hawk agents and monitor alerts generated by these agents. Both the TIBCO Hawk<sup>®</sup> WebConsole and TIBCO Hawk<sup>®</sup> Event Service implement the Console API to monitor and manage agent behavior. Programmers can use the Console API to write custom applications similar to these applications to monitor agent behavior, subscribe to alert messages, and invoke microagent methods.

For more information, refer to the *TIBCO Hawk<sup>®</sup> Programmers Guide* of [TIBCO Hawk<sup>®</sup> Documentation](#).

## Configuration Object API

The Configuration Object API is a Java interface for writing custom rulebases. Rulebases are used by Hawk agents to monitor and manage systems and applications. The Configuration Object API provides classes to define rules, tests and actions. Instances of these classes are put together to define a new rulebase.

For more information, refer to the *TIBCO Hawk<sup>®</sup> Programmers Guide* and *Configuration API Reference* of [TIBCO Hawk<sup>®</sup> Documentation](#).

## AMI API

The AMI API allows to monitor application statistics with the Hawk API and make them manageable using Hawk Agent.

For more information, refer to the *TIBCO Hawk<sup>®</sup> Programmers Guide* and *AMI Reference* of [TIBCO Hawk<sup>®</sup> Documentation](#).

## Security API

The TIBCO Hawk Security API is used to build security plug-in modules used for secure agent and console interactions. The security mechanism actually involves two modules, one that is used by the agent and another that is used by the console. If you use the Console API to write console applications that operate in a secure TIBCO Hawk environment, you must have access to the console-side security plug-in class to be able to perform management operations on agents. The TIBCO Hawk Console requires a security plug-in class to manage agents in a secure environment.

For more information, see the *TIBCO Hawk<sup>®</sup> Programmers Guide* of [TIBCO Hawk<sup>®</sup> Documentation](#).