BLACKDUCK | Hub

Hub Installation Guide (Docker)

Version 3.7.0

This edition of the *Hub Installation Guide (Docker)* refers to version 3.7.0 of the Black Duck Hub.

This document created or updated on Friday, May 12, 2017.

Please send your comments and suggestions to:

Black Duck Software, Incorporated
800 District Avenue, Suite 201
Burlington, MA 01803-5061 USA

Contents

Preface

The Hub documentation

The documentation for the Hub consists of online help and these documents:

Title	File	Description
Release Notes	release_notes_bd_hub.pdf	Contains information about the new and improved features, resolved issues, and known issues in the current and previous releases.
Hub Installation Guide (Docker)	hub_install_docker.pdf	Contains information about installing and upgrading the Hub based on the Docker architecture.
Hub Installation Guide (AppMgr)	installing_the_bd_hub.pdf	Contains information about installing and upgrading the Hub based on AppMgr.
Getting Started	hub_getting_started.pdf	Provides first-time users with information on using the Hub.
Scanning Best Practices	hub_scanning_best_practices.pdf	Provides best practices for scanning.
Getting Started with the Hub SDK	getting_started_hub_sdk.pdf	Contains overview information and a sample use case.
Report Database	report_db_bd_hub.pdf	Contains information on using the report database.

Hub integration documentation can be found on Confluence.

Customer support

If you have any problems with the software or the documentation, please contact Black Duck Customer Support.

You can contact Black Duck Support in several ways:

- Online: https://www.blackducksoftware.com/support/contact-support
- Email: support@blackducksoftware.com
- Phone: +1 781.891.5100, ext. 5

- Fax: +1 781.891.5145
- Hours: Monday to Friday, 8:00 AM to 6:00 PM Eastern Standard Time (EST), excluding weekends and holidays

Note: Customers who have an Enhanced Customer Support Plan can contact customer support 24 hours a day, 7 days a week to obtain Tier 1 support.

Another convenient resource available at all times is the online customer portal: https://www.blackducksoftware.com/support/customer-success-portal

Training

Black Duck Academy is a one-stop resource for all your Black Duck education needs. It provides you with 24x7 access to online training courses and how-to videos.

New videos and courses are added monthly.

At Black Duck Academy, you can:

- Learn at your own pace.
- Review courses as often as you wish.
- Take assessments to test your skills.
- Print certificates of completion to showcase your accomplishments.

Learn more at https://www.blackducksoftware.com/services/training

View the full catalog of courses and try some free courses at https://academy.blackducksoftware.com

When you are ready to learn, log in or sign up for an account: https://academy.blackducksoftware.com

Customer Success Community

The Black Duck Customer Success Community is our primary online resource for customer support, solutions and information. The Customer Success Community allows users to quickly and easily open support cases and monitor progress, learn important product information, search a knowledgebase, and gain insights from other Black Duck customers. The many features included in the Customer Success Community center around the following collaborative actions:

- Connect – Open support cases and monitor their progress, as well as, monitor issues that require Engineering or Product Management assistance
- Learn – Insights and best practices from other Black Duck product users to allow you to learn valuable lessons from a diverse group of industry leading companies. In addition, the Customer Hub puts all the latest product news and updates from Black Duck at your fingertips, helping you to better utilize our products and services to maximize the value of open source within your organization.
- Solve – Quickly and easily get the answers you're seeking with the access to rich content and product knowledge from Black Duck experts and our Knowledgebase.
- Share – Collaborate and connect with Black Duck staff and other customers to crowdsource solutions and share your thoughts on product direction.

Access the Customer Success Community. If you do not have an account or have trouble accessing the system, please send an email to communityfeedback@blackducksoftware.com or call us at +1 781.891.5100 ext. 5.

Chapter 1: Overview

This document provides instructions for installing Black Duck Hub in a Docker environment.

There are two methods to install the Hub: using system services (installed using AppMgr) or by using Docker containers. Using Docker containers is the preferred method as it provides improvements to the Hub, as described below.

Hub Architecture

The Black Duck Hub is deployed as a set of nine Docker containers. "Dockerizing" the Hub so that different components are containerized allows third-party orchestration tools such as Compose, Swarm, and Kubernetes, to manage all of the individual containers.

The Docker architecture brings these significant improvements to the Hub:

- Improved performance
- Easier installation and updates
- Scalability
- Product component orchestration and stability

For more information on Docker, visit the Docker website: https://www.docker.com/

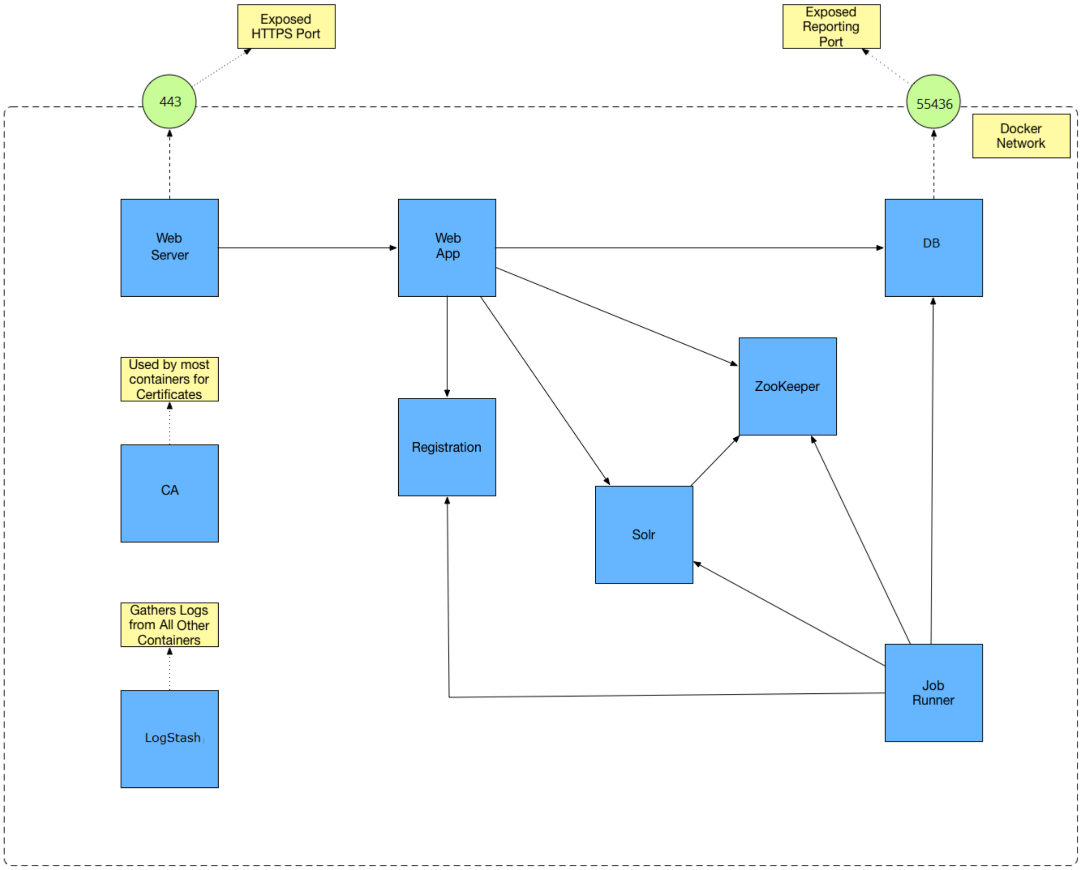
Docker containers

These are the nine containers within the Docker network that comprise the Hub application:

1. Web App
2. Job Runner App
3. Solr
4. Registration
5. DB
6. WebServer
7. Zookeeper

8. LogStash
9. CA

The following diagram shows the basic relationships among the containers and which ports are exposed outside of the Docker network.



This diagram makes no assumptions about which Docker hosts are running which container: it is possible that each container runs on a separate Docker host. All containers are contained within a Docker network. The only two ports exposed outside of the Docker network are the HTTPS port for Hub (via NGiNX) and a read-only database port from Postgres for reporting. All other external communication will go through a proxy or another NGiNX instance. All other communication will be among the containers within the Docker network.

The following tables provide more information on each container.

Web App container

Container Name: Web App	
Image Name	blackducksoftware/hub-webapp:3.7.0
Description	The Web App container is the container that all Web/UI/API requests are made against. It also processes any UI requests. In the diagram, the ports for the Web App are not exposed outside of the Docker network. There is an NGiNX reverse proxy (as described in the WebServer container) that is exposed outside of the Docker network instead.
Scalability	There should only be a single instance of this container. It should not be scaled.
Links/Ports	The Web App container needs to connect to these containers/services: ■ postgres ■ solr ■ zookeeper ■ registration ■ logstash ■ cfssl The container needs to expose port 8080 to other containers that will link to it.
Alternate Host Name Environment Variables	There are times when running in other types of orchestrations that it is useful to have host names set for these containers that are not the default that Docker Compose or Docker Swarm use. These environment variables can be set to override the default host names: ■ postgres: $HUB_POSTGRES_HOST ■ solr: This should be taken care of by ZooKeeper. ■ zookeeper: $HUB_ZOOKEEPER_HOST ■ registration: $HUB_REGISTRATION_HOST ■ logstash: $HUB_LOGSTASH_HOST ■ cfssl: $HUB_CFSSL_HOST
Constraints	■ Default max Java heap size: 4GB ■ Container memory: 4GB ■ Container CPU: 1 CPU
Volumes	log-volume:/opt/blackduck/hub/logs
Environment File	hub-proxy.env

Job runner container

Container Name: Job Runner	
Image Name	blackducksoftware/hub-jobrunner:3.7.0
Description	The Job Runner container is the container that is responsible for running all Hub jobs. This includes matching, BOM building, reports, data updates, and so on. This container does not have any exposed ports.
Scalability	This container can be scaled.
Links/Ports	The Job Runner container needs to connect to these containers/services: ■ postgres ■ solr ■ zookeeper ■ registration ■ logstash ■ cfssl
Alternate Host Name Environment Variables	There are times when running in other types of orchestrations that it is useful to have host names set for these containers that are not the default that Docker Compose or Docker Swarm use. These environment variables can be set to override the default host names: ■ postgres: $HUB_POSTGRES_HOST ■ solr: This should be taken care of by ZooKeeper. ■ zookeeper: $HUB_ZOOKEEPER_HOST ■ registration: $HUB_REGISTRATION_HOST ■ logstash: $HUB_LOGSTASH_HOST ■ cfssl: $HUB_CFSSL_HOST
Constraints	■ Default max Java heap size: 4GB ■ Container memory: 4GB ■ Container CPU: 1 CPU
Volumes	N/A
Environment File	hub-proxy.env

Solr container

Container Name: Solr	
Image Name	blackducksoftware/hub-solr:3.7.0
Description	Solr is an open source enterprise search platform. The Hub uses Solr as its search server for project data. This container has Apache Solr running within it. There is only a single instance of this container. The Solr container exposes ports internally to the Docker network, but not outside of the Docker network.
Scalability	This container should not be scaled.
Links/Ports	The Solr container needs to connect to these containers/services: ■ zookeeper ■ logstash The container needs to expose port 8080 to other containers that will link to it.
Alternate Host Name Environment Variables	There are times when running in other types of orchestrations that it is useful to have host names set for these containers that are not the default that Docker Compose or Docker Swarm use. These environment variables can be set to override the default host names: ■ zookeeper: $HUB_ZOOKEEPER_HOST ■ logstash: $HUB_LOGSTASH_HOST
Constraints	■ Default max Java heap size: 512MB ■ Container memory: 512MB ■ Container CPU: Unspecified
Volumes	N/A
Environment File	N/A

Registration container

Container Name: Registration	
Image Name	blackducksoftware/hub-registration:3.7.0
Description	The container is a small service that handles registration requests from the other containers. At periodic intervals, this container connects to the Black Duck Registration Service and obtains registration updates.

Container Name: Registration	
Scalability	The container should not be scaled.
Links/Ports	The Registration container needs to connect to this containers/services: - logstash The container needs to expose port 8080 to other containers that link to it.
Alternate Host Name Environment Variables	There are times when running in other types of orchestrations that it is useful to have host names set for these containers that are not the default that Docker Compose or Docker Swarm use. These environment variables can be set to override the default host names: - logstash: $HUB_LOGSTASH_HOST
Constraints	- Default max Java heap size: 256MB - Container memory: 256MB - Container CPU: Unspecified
Volumes	config-volume:/opt/blackduck/hub/registration/config
Environment File	hub-proxy.env

DB container

Container Name: DB	
Image Name	blackducksoftware/hub-postgres:3.7.0
Description	The DB container holds the PostgreSQL database which is an open source object-relational database system. The Hub uses the PostgreSQL database to store data. There is a single instance of this container. This is where all Hub data is stored. This is the connection that the Hub App, Job Runner, and potentially other containers use. This port is secured via certificate authentication. A second port is exposed outside of the Docker network. This allows a read-only user to connect via a password set using the `hub_reportdb_changepassword.sh` script. This port and user can be used for reporting and data extraction. Refer to the *Report Database* guide for more information on the report database.
Scalability	There should only be a single instance of this container. It should not be scaled.

Container Name: DB	
Links/Ports	The DB container needs to connect to these containers/services: - logstash - cfssl The container needs to expose port 5432 to other containers that will link to it within the Docker network. This container exposes port 55436 outside of the Docker network for database reporting.
Alternate Host Name Environment Variables	There are times when running in other types of orchestrations that it is useful to have host names set for these containers that are not the default that Docker Compose or Docker Swarm use. These environment variables can be set to override the default host names: - logstash: $HUB_LOGSTASH_HOST - cfssl: $HUB_CFSSL_HOST
Constraints	- Default max Java heap size: N/A - Container memory: 2GB - Container CPU: 1 CPU
Volumes	data-volume:/var/lib/postgresql/data
Environment File	N/A

WebServer container

Container Name: WebServer	
Image Name	blackducksoftware/hub-nginx:3.7.0
Description	The WebServer container is a reverse proxy for the Hub Web App. It has a port exposed outside of the Docker network. This is the container configured for HTTPS. There are config volumes here for configuration of HTTPS.
Scalability	The container should not be scaled.
Links/Ports	The Web App container needs to connect to these containers/services: - webapp - cfssl This container exposes port 443 outside of the Docker network.

Container Name: WebServer	
Alternate Host Name Environment Variables	There are times when running in other types of orchestrations that it is useful to have host names set for these containers that are not the default that Docker Compose or Docker Swarm use. These environment variables can be set to override the default host names: ■ webapp: $HUB_WEBAPP_HOST ■ cfssl: $HUB_CFSSL_HOST
Constraints	■ Default max Java heap size: N/A ■ Container memory: 512MB ■ Container CPU: Unspecified
Volumes	webserver-volume:/opt/blackduck/hub/webserver/security
Environment File	hub-webserver.env

ZooKeeper container

Container Name: Zookeeper	
Image Name	blackducksoftware/hub-zookeeper:3.7.0
Description	This container stores data for the Hub App, Job Runners, Solr, and potentially other containers. It exposes ports within the Docker network, but not outside the Docker network.
Scalability	This container should not be scaled.
Links/Ports	The Zookeeper container needs to connect to this container/service: ■ logstash The container needs to expose port 2181 within the Docker network to other containers that will link to it.
Alternate Host Name Environment Variables	There are times when running in other types of orchestrations that it is useful to have host names set for these containers that are not the default that Docker Compose or Docker Swarm use. These environment variables can be set to override the default host names: ■ logstash: $HUB_LOGSTASH_HOST
Constraints	■ Default max Java heap size: 256MB ■ Container memory: 256MB ■ Container CPU: Unspecified

Container Name: Zookeeper	
Volumes	N/A
Environment File	N/A

LogStash container

Container Name: LogStash	
Image Name	blackducksoftware/hub-logstash:3.7.0
Description	The LogStash container collects and store logs for all containers.
Scalability	There should only be a single instance of this container. It should not be scaled.
Links/Ports	The container needs to expose port 5044 within the Docker network to other containers/services that will link to it.
Constraints	▪ Default max Java heap size: 1GB ▪ Container memory: 1GB ▪ Container CPU: Unspecified
Volumes	log-volume:/var/lib/logstash/data
Environment File	N/A

CA container

Container Name: CA	
Image Name	blackducksoftware/hub-cfssl:3.7.0
Description	The CA container uses CFSSL which is used for certificate generation for PostgreSQL, NGiNX, and clients that need to authenticate to Postgres.
Scalability	There should only be a single instance of this container. It should not be scaled.
Links/Ports	The container needs to expose port 8888 within the Docker network to other containers/services that link to it.
Constraints	▪ Default max Java heap size: N/A ▪ Container memory: 512MB ▪ Container CPU: Unspecified
Volumes	config-volume:/etc/cfssl
Environment File	N/A

Scaling Job Runners

The Job Runner container can be scaled.

Using Docker Compose

You may need to be a user in the docker group, a root user, or have `sudo` access to run the following command.

This example adds a second Job Runner container:

```
docker-compose -p scale jobrunner=2
```

You can remove a Job Runner container by specifying a lower number than the current number of Job Runners. The following example scales back the Job Runner container to a single container:

```
docker-compose -p scale jobrunner=1
```

Using Docker Swarm

You may need to be a user in the docker group, a root user, or have `sudo` access to run the following command.

This example adds a second Job Runner container:

```
docker service scale hub_jobrunner=2
```

You can remove a Job Runner container by specifying a lower number than the current number of Job Runners. The following example scales back the Job Runner container to a single container:

```
docker service scale hub_jobrunner=1
```

Components hosted on Black Duck servers

The components hosted on Black Duck servers are:

- **Registration server**: Used to validate the Hub license.
- **Black Duck KnowledgeBase server**: Cloud-based version of the Black Duck KnowledgeBase (KB) that contains many of the most popular open source projects in the KB. Hosting the Black Duck KB in the cloud means that the Hub can display the most up-to-date information about open source software (OSS) projects without requiring regular updates on your host machine.
- **Documentation server**: The online help and PDF documents for the Hub are accessed on an external server instead of being stored locally on the Hub server. This means that you always access the most up-to-date versions of the documentation.

Chapter 2: Installation planning

This chapter describes the pre-installation planning and configuration that must be performed before you can install the Black Duck Hub.

Getting started

The process for installing the Hub depends on whether you are installing the Hub for the first time or upgrading from a previous version of the Hub (either based on the AppMgr architecture or based on the Docker architecture).

New installations

For new installation of the Hub:

1. Read this planning chapter to review all requirements.
2. After ensuring that you meet all requirements, go to Chapter 3 for installation instructions.
3. Review Chapter 4 for any post-installation tasks.

Upgrading from a previous version of the Hub

1. Read this planning chapter to review all requirements,
2. After ensuring that you meet all requirements, go to Chapter 6 for upgrade instructions.
3. Review Chapter 4 for any post-installation tasks.

Hardware requirements

The following is the minimum hardware that is needed to run a single instance of each container:

- 4 CPUs
- 16 GB RAM (or 15 GB if you are constrained running on Amazon Web Services or other cloud providers)
- 200 GB of free disk space for the database
- 2.5 GB container for database backups
- 30 GB of free disk space for other containers

The descriptions of each container document the individual requirements for each container if it will be running on a different machine or if more than one instance of a container will be running (currently only supported for the Job Runner container.)

Note: The amount of required disk space is dependent on the number of projects being managed, so individual requirements can vary. Consider that each project requires approximately 200 MB.

Black Duck recommends monitoring disk utilization on the Hub servers to prevent disks from reaching capacity which could cause issues with the Hub.

Docker requirements

Docker Version

The Hub installation supports Docker Version 17.03.x (CE or EE)

Supported orchestration tools

- Docker Compose - a tool for running multi-container Docker applications.

  The minimum supported version of docker-compose must be able to read Docker Compose 2.1 files.

- Docker Swarm - a clustering and scheduling tool for Docker containers. With Docker Swarm you can manage a cluster of Docker nodes as a single virtual system.

  The minimum supported version of docker-compose must be able to read Docker Compose 3.1 files.

- Docker Run - a tool for running multi-container Docker applications.

The content for each orchestration tool is described here.

Docker Swarm requirements

There are two restrictions when using the Hub in Docker Swarm:

- The PostgreSQL database must run on the same node so that data is not lost (hub-database service).

- The hub-webapp service and the hub-logstash service must run on the same host.

  This is required so that Web App can access the logs that need to be downloaded.

Software requirements

The Hub is a web application that has an HTML interface. You access the application via a web browser. The following web browser versions have been tested with the Hub:

- Chrome 58.0.3029.81 (64 bit)
- Firefox 53.0 (64-bit)/(32-bit)
- Internet Explorer 11.0.41 (KB4014661)
- Edge 38.14393.0.0

- EdgeHTML 14.14393
- Safari 10.0.3 (12602.4.8)

Note: These browser versions are the currently-released versions on which Black Duck has tested Hub. Newer browser versions may be available after the Hub is released, and may or may not work as expected. Older browser versions may work as expected, but have not been tested and may not be supported.

The Hub UI requires a minimum horizontal screen resolution of 1280. The following screen resolutions have been tested using the minimum horizontal requirement:

- 1280 x 720
- 1280 x 768
- 1280 x 800
- 1280 x 960
- 1280 x 1024

Network requirements

The Hub requires the following ports to be externally accessible:

- Port 443 – Web server HTTPS port for the Hub via NGiNX
- Port 55436 – Read-only database port from PostgreSQL for reporting

If your corporate security policy requires registration of specific URLs, connectivity from your Hub installation to Black Duck hosted servers is limited to communications via HTTPS/TCP on port 443 with the following servers:

- updates.suite.blackducksoftware.com (to register your software)
- kb.blackducksoftware.com (access the Black Duck KB data)
- doc.blackducksoftware.com (open the online help system)

Note: If you are using a network proxy, these URLs must be configured as destinations in your proxy configuration.

Proxy server requirements

The Hub supports:

- No Authentication
- Digest
- NTLM

If you are going to make proxy requests to the Hub, work with the proxy server administrator to get the following required information:

- The protocol used by proxy server host (http or https).
- The name of the proxy server host
- The port on which the proxy server host is listening.

Use the hub-proxy.env file to configure your proxy settings.

Additionally, work with the proxy server administrator to determine if the proxy server will rewrite requests to the Hub. If requests will be re-written, let the proxy server administrator know that the following HTTP headers can be used to preserve the original requesting host details.

HTTP Header	Description
X-Forwarded-Host	Tracks the list of hosts that were re-written or routed to make the request. The original host is the first host in the comma-separated list. **Example:** `X-Forwarded-Host: "10.20.30.40,my.example, 10.1.20.20"`
X-Forwarded-Port	Contains a single value representing the port used for the original request. **Example:** `X-Forwarded-Port: "9876"`
X-Forwarded-Proto	Contains a single value representing the protocol scheme used for the original request. **Example:** `X-Forwarded-Proto: "https"`
X-Forwarded-Prefix	Contains a prefix path used for the original request. **Example:** `X-Forwarded-Prefix: "prefixPath"` To successfully scan files, you must use the **context** parameter

Configuring your NGiNX server to work with the Hub

If you have an NGINX server acting as an HTTPS server/proxy in front of the Hub, you must modify the NGINX configuration file so that the NGINX server passes the correct headers to the Hub. The Hub then generates the URLs that use HTTPS.

Note: Only one service on the NGINX server can use https port 443.

To pass the correct headers to the Hub, edit the `location` block in the nginx.config configuration file to:

```
location / {

   client_max_body_size 1024m;
```

```
  proxy_pass http://127.0.0.1:8080;
  proxy_pass_header X-Host;
  proxy_set_header Host $host:$server_port;
  proxy_set_header X-Real-IP $remote_addr;
  proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
  proxy_set_header X-Forwarded-Proto $scheme;
}
```

If the X-Forwarded-Prefix header is being specified in a proxy server/load balancer configuration, edit the `location` block in the nginx.config configuration file:

```
location/prefixPath {
  proxy_set_header X-Forwarded-Prefix "/prefixPath";
}
```

To scan files successfully, you must use the **context** parameter in the Hub Scanner.

> **Note:** Although these instructions apply to an NGINX server, similar configuration changes would need to be made for any type of proxy server.

Amazon services

You can install the Hub on Amazon Web Services (AWS) and also use Amazon EC2 Container Service (Amazon ECS).

- Refer to your AWS documentation and your AMI documentation for more information on AWS.
- Refer to your Amazon Container Service documentation for more information on Amazon container services.

Chapter 3: Installing the Hub

Prior to installing the Hub, ensure that:

- Port 443 is not in use by another process.
- The server must be able to access the following URL: `updates.suite.blackducksoftware.com` which is used to validate the Hub license.
- Obtain the Hub Docker images and orchestration files:
  - Hub Docker images are available for download at Docker Hub repository: https://hub.docker.com/u/blackducksoftware/.
  - The orchestration files are located here: https://github.com/blackducksoftware/hub/raw/master/archives/hub-docker-3.7.0.tar

    Unpack the `.tar` file on the server you want to install the Hub.

Note: The `.tar` file includes environment files used to configure web server and proxy settings. You can configure these settings before or after you install the Hub.

Docker Compose and Docker Swarm distributions

This is the content for the Docker Compose and Docker Swarm distributions:

- `docker-compose.dbmigrate.yml`: Docker Compose file used to migrate the PostgreSQL database.
- `docker-compose.yml`: Docker Compose file.
- `hub-proxy.env`: Environment file to configure proxy settings.
- `hub-webserver.env`: Environment file to configure web server settings.

In the `bin` directory:

- `hub_create_data_dump.sh`: Script used to back up the PostgreSQL database.
- `hub_db_migrate.sh`: Script used to migrate the PostgreSQL database.
- `hub_reportdb_changepassword.sh`: Script used to set and change the report database password. Refer to the Report Database guide for more information.
- `hub_webserver_use_customer_cert_key.sh`: Script used to use custom certificates.

Docker Run distribution

This is the content for the Docker Run distribution:

- `docker-hub.sh`: Docker Run script. See the next section, Docker Run commands, for more information.
- `docker-run.sh`: Docker Run script. See the next section, Docker Run commands, for more information.
- `docker-stop.sh`: Docker Run script. See the next section, Docker Run commands, for more information.
- `hub-proxy.env`: Environment file to configure proxy settings.
- `hub-webserver.env`: Environment file to configure web server settings.

In the `bin` directory:

- `hub_create_data_dump.sh`: Script used to back up the PostgreSQL database.
- `hub_db_migrate.sh`: Script used to migrate the PostgreSQL database.
- `hub_reportdb_changepassword.sh`: Script used to set and change the report database password. Refer to the Report Database guide for more information.
- `hub_webserver_use_customer_cert_key.sh`: Script used to use custom certificates.

Docker Run commands

The `docker-run` directory has these files:

- `docker-run.sh`

  A script useful for starting the Hub using standard Docker CLI commands.

- `docker-stop.sh`

  A script useful for stopping a running Hub instance using standard Docker CLI commands.

- `docker-hub.sh`

  A multi-purpose orchestration script used for starting, stopping, and removing the Hub using standard Docker CLI commands.

Users running these scripts may need to be a user in the docker group, a root user, or have `sudo` access.

About docker-run.sh

The docker-run.sh script accepts one mandatory argument: the version of the Hub installed on the system. This argument is in the following format: MM.mm.ff, for example, 3.7.0.

To start the Hub:

```
docker-run.sh 3.7.0
```

About docker-stop.sh

The docker-stop.sh script does not accept any arguments.

To stop the Hub:

```
docker-stop.sh
```

About docker-hub.sh

The docker-hub.sh script has these parameters.

Parameter	Description
-d, **--down**	Stops and removes the containers. If **--volumes** is provided, it also removes the volumes.
-m, **--migrate**	Migrates data from the PostgreSQL dump file.
-r, **--release**	The Hub version to deploy. This parameter is mandatory when using the **--up** parameter.
-s, **--stop**	Stops the containers, but leaves them on the system. Does not affect volumes.
-u, **--up**	Starts the containers. Creates volumes if they do not already exist.
-v, **--volumes**	If provided with **--down**, removes the volumes and all data stored within them. The **--volumes** parameter cannot be run in the same command as the **--up** or **--stop** parameters. **Caution: The --volumes flag will *delete data* from the system. This is irreversible. Be sure you understand the ramifications before running this command.**

Note that some arguments are mutually exclusive and cannot be combined: you cannot run the **--up**, **--stop**, and **--down** parameters in the same command. Error messages will appear if you do not follow these rules, however the running system will not be affected.

Examples:

Start the Hub:

```
docker-hub.sh -r 3.7.0 -u
```

Stop the Hub:

```
docker-hub.sh -s
```

Uninstalling the Hub but leaving volumes in place:

```
docker-hub.sh -d
```

Uninstalling the Hub and removing volumes. This removes *all* data.

```
docker-hub.sh -d -v
```

Configuration settings

There are two environment files, located in the located in the `bin` directory of the orchestration directory you used to install the Hub. Use these files to configure web server and proxy settings:

- hub-webserver.env
- hub-proxy.env

You can define the settings in these files before or after you install the Hub:

- To set configuration settings *before* installing the Hub, edit the file as described below and save your changes.
- To modify existing settings *after* installing the Hub, for Docker Compose and Docker Run, do the following:

  1. Stop the containers.

     Docker Compose:

     ```
     docker-compose -p hub stop
     ```

     Docker Run:

     ```
     docker-stop.sh
     ```

  2. Open the file and edit the settings as described below.
  3. Reconnect the containers.

     Docker Compose:

     ```
     docker-compose -p hub up
     ```

     Docker Run:

     ```
     docker-run.sh
     ```

  For Docker Swarm, modify the settings and then redeploy the services in the stack by entering:

  ```
  docker stack deploy -c docker-compose.yml hub
  ```

Web server settings

Edit the hub-webserver.env file to configure the hostname so the certificate host name matches. The environment variable has the service name as the default value.

When the web server starts up, it generates an HTTPS certificate if certificates are not configured. You must specify a value for the PUBLIC_HUB_WEBSERVER_HOST environment variable to tell the web server the hostname it will listen on so that the hostnames can match. Otherwise, the certificate will only have the service name to use as the host name.

Proxy settings

Edit the hub-proxy.env file to configure proxy settings. You will need to configure these settings if a proxy is required for external internet access.

There are three containers that need access to services hosted by Black Duck:

- Registration
- Job runner
- Web App

Proxy environment variables are:

- HUB_PROXY_HOST. Name of the proxy server host.
- HUB_PROXY_PORT. The port on which the proxy server host is listening.
- HUB_PROXY_SCHEME. Protocol to use to connect to the proxy server.
- HUB_PROXY_USER. Username to access the proxy server.

The environment variables for NTLM proxies are:

- HUB_PROXY_WORKSTATION. The workstation the authentication request is originating from. Essentially, the computer name for this machine.
- HUB_PROXY_DOMAIN. The domain to authenticate within.

The variables for configuring Cross Origin Resource Sharing (CORS) are:

- BLACKDUCK_HUB_CORS_ENABLED
- BLACKDUCK_CORS_ALLOWED_ORIGINS_PROP_NAME
- BLACKDUCK_CORS_ALLOWED_HEADERS_PROP_NAME
- BLACKDUCK_CORS_EXPOSED_HEADERS_PROP_NAME

Refer to the *Getting Started with the Hub SDK* guide for more information on these variables.

The variable to enable access to the API documentation when using a reverse proxy with Hub mounted under a sub-path is:

- BLACKDUCK_SWAGGER_PROXY_PREFIX

Refer to the *Getting Started with the Hub SDK* guide for more information on this variable.

Proxy password

The following services require the proxy password:

- Web App
- Registration
- Job Runner

There are two methods for specifying a proxy password:

- Mount a directory that contains a file called HUB_PROXY_PASSWORD_FILE to `/run/secrets`. This is the most secure option.
- Specify an environment variable called HUB_PROXY_PASSWORD that contains the proxy password.

For Docker Swarm, you can use these methods or use the docker secret command to create a secret called HUB_PROXY_PASSWORD_FILE:

1. Use the docker secret command to tell Docker Swarm the secret. The name of the secret must include in the stack name. In the following example, the stack name is 'hub':

   ```
   docker secret create hub_HUB_PROXY_PASSWORD_FILE <file containing
   password>
   ```

2. Add to the services section of the Web App, Registration, and Job Runner services the password secret:

   ```
        secrets:
         - HUB_PROXY_PASSWORD_FILE
   ```

You can use the hub-proxy.env file to specify an environment variable if it is not specified in a separate mounted file or secret:

1. Remove the pound sign (#) located in front of HUB_PROXY_PASSWORD so that it is no longer commented out.
2. Enter the proxy password.
3. Save the file.

Installing the Hub

To install the Hub, users running the following command may need to be a user in the docker group, a root user, or have `sudo` access.

Docker Compose:

```
docker-compose -f docker-compose.yml -p hub up -d
```

Docker Swarm:

```
docker stack deploy -c docker-compose.yml hub
```

Note that there are some versions of Docker where if the images live in a private repository, docker stack will not pull them unless the following flag is added to the above command:

```
--with-registry-auth
```

Docker Run:

```
docker-run.sh 3.7.0
```

The Black Duck Hub is installed.

Once all of the containers for Hub are up, the web application for the Hub will be exposed on port 443 to the docker host. You can access the Hub by entering the following:

https://hub.example.com

Understanding the default sysadmin user

When you install the Black Duck Hub, there is a default system administrator (sysadmin) account already configured. The default sysadmin user has all roles and permissions associated with it.

Tip: As a best practice, you should use the default sysadmin account for your initial log in and then immediately change the default password—blackduck—so that the server is secure.

Chapter 4: Post-installation tasks

Optionally after installing the Hub, you can:

- Replace the existing self-signed certificate for the Web Server with a custom certificate.
- Access log files.

You can also modify the web server and proxy server configuration settings after installing the Hub

Using custom certificates

The Web Server container has a self-signed certificate obtained from Docker. You may want to replace this certificate with a custom certificate.

To use a custom certificate, run the `hub_webserver_use_custom_cert_key.sh` script located in the `./bin` directory. This script replaces the existing certificate and key with your custom certificate and key file.

```
./bin/hub_webserver_use_custom_cert_key.sh <path to certificate> <path to key
file>
```

Accessing log files

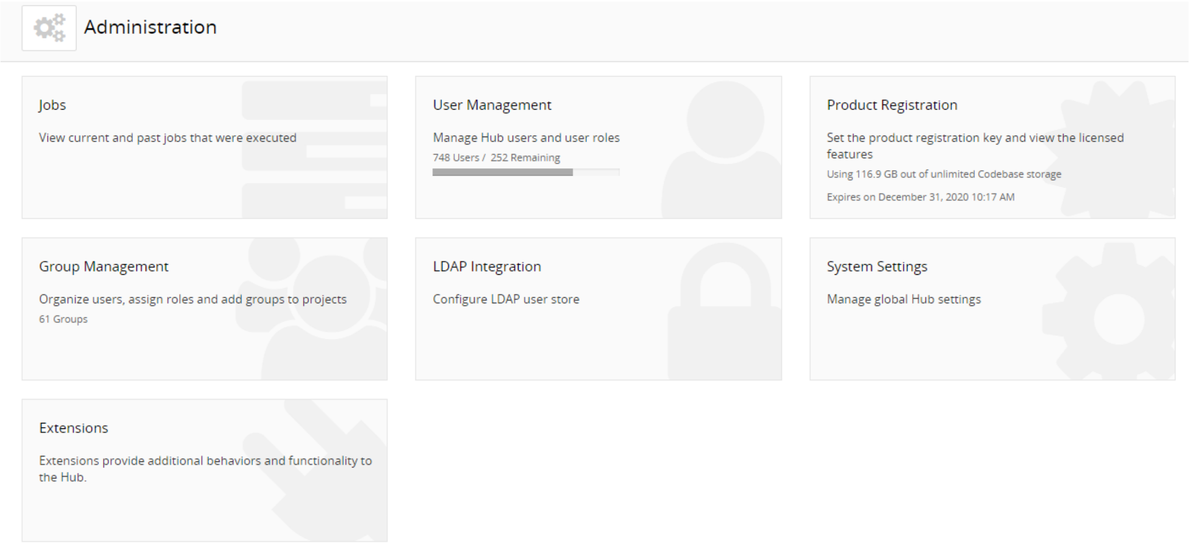
You may need to troubleshoot an issue or provide log files to Customer Support.

Users with the System Administrator role can download a zipped file that contains the current log files.

To download the log files from the Hub UI

1. Log in to the Hub with the System Administrator role.
2. Click the expanding menu icon (≡) and select **Administration**.

   The Administration page appears.



Administration

Jobs

View current and past jobs that were executed

User Management

Manage Hub users and user roles

748 Users / 252 Remaining

Product Registration

Set the product registration key and view the licensed features

Using 116.9 GB out of unlimited Codebase storage

Expires on December 31, 2020 10:17 AM

Group Management

Organize users, assign roles and add groups to projects

61 Groups

LDAP Integration

Configure LDAP user store

System Settings

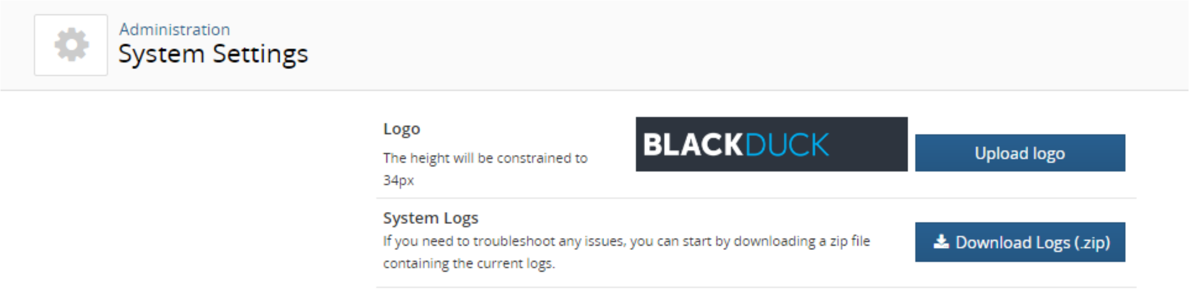
Manage global Hub settings

Extensions

Extensions provide additional behaviors and functionality to the Hub.

3. Select **System Settings**.

   The System Settings page appears.



Administration

System Settings

Logo

The height will be constrained to 34px

System Logs

If you need to troubleshoot any issues, you can start by downloading a zip file containing the current logs.

4. Click **Download Logs (.zip)**.

   It may take a few minutes to prepare the log files.

Viewing log files for a container

Use the docker-compose `logs` command to view all logs:

```
docker-compoase logs
```

Chapter 5: Uninstalling the Hub

To uninstall the Hub, do one of the following:

- Stop and remove the containers and remove the volumes.

  Docker Compose:

  ```
  docker-compose -p hub down -v
  ```

  Docker Run:

  ```
  docker-hub.sh -d -v
  ```

  Docker Swarm:

  ```
  docker stack rm hub
  ```

- Stop and remove the containers but keep the volumes. For example:

  Docker Compose

  ```
  docker-compose -p hub down
  ```

  Docker Run:

  ```
  docker-hub.sh -d
  ```

  Docker Swarm:

  ```
  docker volume prune
  ```

  Caution: This command removes *all* unused volumes: volumes not referenced by *any* container are removed. This includes unused volumes not used by other applications.

Note that the PostgreSQL database is not backed up. Use these instructions to back up the database.

Click here for more information on Docker Run commands.

Chapter 6: About upgrading the Hub

The Hub supports upgrading to any available version, giving you the ability to jump multiple versions in a single upgrade.

The upgrade instructions depend on whether you are upgrading from a previous version of the Hub based on the AppMgr architecture or upgrading based on the Docker architecture.

- If you are upgrading from a previous version of the Hub based on the AppMgr architecture to the Docker architecture:
    1. Migrate your PostgreSQL database.

       This is an optional step if you want to retain your existing database data.
    2. Upgrade the Hub.
- If you are upgrading from a previous version of the Hub that was based on the Docker architecture to a newer version of the Hub,
    1. Optionally, back up your PostgreSQL database.
    2. Upgrade the Hub.

Migrating PostgreSQL data

To use your existing PostgreSQL data you must migrate the database data.

The process to migrate the PostgreSQL database consists of:

1. Backing up the original PostgreSQL database.
2. Restoring the data.

Step 1. Backing up the database

To back up the AppMgr PostgreSQL database

These procedures back up the PostgreSQL database to a dump file.

1. Log in to the Hub server as the **blckdck** user.

   > **Note:** This is the user that owns the Hub database and installation directory.

2. Run the following commands to dump to a compressed file.

   ```
   export PATH=$PATH:/opt/blackduck/hub/postgresql/bin
   export PGPORT=55436
   pg_dump -Fc -f /tmp/bds_hub.dump bds_hub
   ```

   Tip: Ensure that you dump the database to a location with sufficient free space. This example uses `/tmp`.

   This command puts the information from the `bds_hub` database into a file called `bds_hub.dump` in the `/tmp` directory. It ignores several scratch tables that do not need to be backed up.

3. Save the `bds_hub.dump` file on another system or offline.

Tip: If you find that dumping the database takes too long, you can greatly increase the speed by dumping it to an uncompressed file. The trade-off is that while the dump is completed up to 3 times faster, the resulting file may be 4 times larger. To experiment with this on your system, add the `--compress=0` parameter to your `pg_dump` command.

✻ To back up the PostgreSQL database for a single-container AppMgr Hub

1. Run the following command to create a PostgreSQL dump file:

   ```
   docker exec -it <containerid or name> pg_dump -U blackduck -Fc -f
   /tmp/bds_hub.dump bds_hub
   ```

2. Copy the dump file out of the container by running the following command:

   ```
   docker cp <containerid>:<path to dump file in container> .
   ```

✻ To back up the PostgreSQL database for an existing Hub Container

The following command creates a dump file from the PostgreSQL database in the `/tmp` container directory and copies it over to your local machine (outside of the container) and then deletes the file in the `/tmp` container directory:

1. `./hub_create_data_dump.sh` *`/destination/path`*

Step 2. Restoring the data

Restoring the database data so that it can be used with the multi-image Docker version of the Hub consists of:

1. Starting PostgreSQL.
2. Restoring the database data.
3. Stopping the existing containers (Docker Compose).

✻ To restore the PostgreSQL data

Docker Compose:

1. Use the `docker-compose.dbmigrate.yml` file. It starts the containers and volumes needed to migrate the database.

   ```
   docker-compose -f docker-compose.dbmigrate.yml -p hub up -d
   ```

2. After the DB container has started, run the migration script. This script restores the data from the existing database dump file.

   ```
   ./bin/hub_db_migrate.sh <path to dump file>
   ```

3. Stop the containers and services that were started in step 1.

   ```
   docker-compose -f docker-compose.dbmigrate.yml -p hub stop
   ```

After stopping the containers, you can install the multi-image Docker version of the Hub.

Docker Swarm:

1. Use the `docker-compose.dbmigrate.yml` file. It starts the containers and volumes needed to migrate the database.

   ```
   docker stack deploy -c docker-compose.dbmigrate.yml hub
   ```

   Note that there are some versions of Docker where if the images live in a private repository, docker stack will not pull them unless the following flag is added to the above command:

   ```
   --with-registry-auth
   ```

2. After the DB container has started, run the migration script. This script restores the data from the existing database dump file.

   ```
   ./bin/hub_db_migrate.sh <path to dump file>
   ```

You can now install the multi-image Docker version of the Hub.

Docker Run:

1. Run the following command to start the containers, run the migration script, and then stop the containers:

   ```
   docker-hub.sh -r 3.7.0 -m <path/to/dump/file>
   ```

You can now install the multi-image Docker version of the Hub.

Error messages

When the dump file is restored from the an AppMgr installation of the Hub, you may receive error messages such as:

> "ERROR: role "blckdck" does not exist"

along with other error messages. Also, at the end of the migration, you may see the following:

> WARNING: errors ignored on restore: 7

These error messages and warnings can be ignored. They will not affect the restoration of the data.

Upgrading the Hub

⚙ To upgrade the Hub from a previous version of the Hub based on the Docker architecture

1. Stop the existing containers.

   Docker Compose:

   ```
   docker-compose -p hub stop
   ```

2. Run the following command.

   Using the docker-compose file included in the newer version of the Hub for Docker Compose:

   ```
   docker-compose -f docker-compose.yml -p hub up -d
   ```

⚙ To upgrade the Hub from a previous version of the Hub based on the AppMgr architecture

1. Run one of the following commands:

   Docker Compose (using the docker-compose file included in the newer version of the Hub):

   ```
   docker-compose -f docker-compose.yml -p hub up -d
   ```

   Docker Run (this example upgrades to release 3.7.0):

   ```
   docker-hub.sh -r 3.7.0 -u
   ```

   Docker Swarm:

   ```
   docker stack deploy -c docker-compose.yml hub
   ```