



BLACKDUCK | Hub

Installation Guide

Version 4.0.0



This edition of the *Installation Guide* refers to version 4.0.0 of the Black Duck Hub.

This document created or updated on Tuesday, June 27, 2017.

Please send your comments and suggestions to:

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

Copyright © 2017 by **Black Duck Software, Inc.**

All rights reserved. All use of this documentation is subject to the license agreement between Black Duck Software, Inc. and the licensee. No part of the contents of this document may be reproduced or transmitted in any form or by any means without the prior written permission of Black Duck Software, Inc.

Black Duck, Know Your Code, and the Black Duck logo are registered trademarks of Black Duck Software, Inc. in the United States and other jurisdictions. Black Duck Code Center, Black Duck Code Sight, Black Duck Hub, Black Duck Protex, and Black Duck Suite are trademarks of Black Duck Software, Inc. All other trademarks or registered trademarks are the sole property of their respective owners.

Chapter 1: Overview	1
Hub Architecture	1
Components hosted on Black Duck servers	1
Chapter 2: Installation planning	2
Getting started	2
New installations	2
Upgrading from a previous version of the Hub	2
Hardware requirements	2
Docker requirements	3
Software requirements	3
Network requirements	4
Database requirements	4
Proxy server requirements	5
Configuring your NGINX server to work with the Hub	5
Amazon services	6
Chapter 3: Installing the Hub	8
Installation media	9
Distributions	9
Installing the Hub	10
Understanding the default sysadmin user	11
Chapter 4: Post-installation tasks	12
Using custom certificates	12
Accessing log files	13
Obtaining logs	14
Viewing log files for a container	14
Scaling job runners	15
Configuring secure LDAP	15
Chapter 5: Uninstalling the Hub	20
Chapter 6: Upgrading the Hub	21
Upgrading from the AppMgr architecture	21
Migrating your PostgreSQL database	21
Upgrading the Hub	23
Upgrading from a single-container AppMgr Hub	24

Migrating your PostgreSQL database	24
Upgrading the Hub	25
Upgrading from an existing Docker architecture	26
Backing up your PostgreSQL database	26
Upgrading the Hub	26
Appendix A: Configuration settings	28
Configuring Web server settings	29
Configuring the hostname	29
Configuring the host port	29
Disabling IPv6	30
Configuring Proxy settings	30
Configuring an external PostgreSQL instance	31
Appendix B: Docker Run commands	34
Appendix C: Docker containers	37
Web App container	39
Job runner container	40
Solr container	41
Registration container	41
DB container	42
WebServer container	43
ZooKeeper container	44
LogStash container	45
CA container	45

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.
Installation Guide	hub_install.pdf	Contains information about installing and upgrading the Hub.
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.

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

Hub Architecture

The Black Duck Hub is deployed as a set of Docker containers. "Dockerizing" the Hub so that different components are containerized allows third-party orchestration tools such as Compose or Swarm 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

See [Docker containers](#), for more information on the Docker containers that comprise the Hub application.

Visit the Docker website: <https://www.docker.com/> for more information on Docker. To obtain installation information, go to <https://docs.docker.com/engine/installation/>.

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 all containers:

- 4 CPUs
- 16 GB RAM (or 15 GB if you are constrained running on Amazon Web Services or other cloud providers)
- 250 GB of free disk space for the database and other Hub containers
- Commensurate space for database backups

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 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.

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).

This does *not* apply to installations using an [external PostgreSQL instance](#).

- 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.

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

The distribution content for each orchestration tool is described [here](#).

Note: For scalability, Black Duck recommends running the Hub on a single node Swarm deployment.

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 59.0.3071.104 (Official Build) (64-bit)
- Firefox 53.0.03 (64-bit)/(32-bit)
- Internet Explorer 11.0.43 (KB4021558)
- Microsoft Edge 38.14393.1066.0
- Microsoft EdgeHTML 14.14393
- Safari 10.1.1 (12603.2.4)

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.

Database requirements

The Hub uses the PostgreSQL object-relational database to store data.

Prior to installing the Hub, determine whether you want to use the database container that is automatically installed or an external PostgreSQL instance: the Hub supports using [Amazon Relational Database Service \(RDS\)](#) for the external PostgreSQL instance.

⚙ To use an external PostgreSQL instance:

1. Set up your external PostgreSQL instance using Amazon RDS.

When creating your RDS instance, set the "Master User" to **blackduck**.

2. Configure your [database connection settings](#).
3. [Install](#) or [upgrade](#) the Hub.

Currently, the Hub requires PostgreSQL 9.4.x.

Proxy server requirements

The Hub supports:

- No Authentication
- Digest
- Basic
- 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](#).

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.

If the proxy server will rewrite requests to the Hub, 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	<p>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.</p> <p>Example:</p> <pre>X-Forwarded-Host: "10.20.30.40,my.example, 10.1.20.20"</pre>
X-Forwarded-Port	<p>Contains a single value representing the port used for the original request.</p> <p>Example:</p> <pre>X-Forwarded-Port: "9876"</pre>
X-Forwarded-Proto	<p>Contains a single value representing the protocol scheme used for the original request.</p> <p>Example:</p> <pre>X-Forwarded-Proto: "https"</pre>
X-Forwarded-Prefix	<p>Contains a prefix path used for the original request.</p> <p>Example:</p> <pre>X-Forwarded-Prefix: "prefixPath"</pre> <p>To successfully scan files, you must use the context parameter</p>

Amazon services

You can:

- Install the Hub on Amazon Web Services (AWS)

Refer to your [AWS documentation](#) and your [AMI documentation](#) for more information on AWS.

- Use Amazon EC2 Container Service (Amazon ECS).

Refer to your [Amazon Container Service documentation](#) for more information on Amazon container services.

- Use Amazon Relational Database Service (RDS) for the PostgreSQL database that is used by the Hub.

Refer to your [Amazon Relational Database Service documentation](#) for more information on Amazon RDS.

Currently the Hub requires PostgreSQL version 9.4.x.

Click [here](#) for more information on configuring an external PostgreSQL server.

Chapter 3: Installing the Hub

Prior to installing the Hub, ensure that you meet the following requirements:

Hub Installation Requirements	
Hardware requirements	
<input type="checkbox"/>	You have ensured that your hardware meets the minimum hardware requirements .
Docker requirements	
<input type="checkbox"/>	You have ensured that your system meets the docker requirements .
Software requirements	
<input type="checkbox"/>	You have ensured that your system and potential clients meet the software requirements .
Network requirements	
<input type="checkbox"/>	<p>You have ensured that your network meets the network requirements. Specifically:</p> <ul style="list-style-type: none">• Port 443 and port 55436 are externally accessible.• The server has access to <code>updates.suite.blackducksoftware.com</code> which is used to validate the Hub license.
Database requirements	
<input type="checkbox"/>	<p>You have selected your database configuration.</p> <p>Specifically, you have configured database settings if you are using an external PostgreSQL instance.</p>
Proxy requirements	
<input type="checkbox"/>	<p>You have ensured that your network meets the proxy requirements.</p> <p>Configure proxy settings before or after installing the Hub.</p>
Web server requirements	
<input type="checkbox"/>	Configure web server settings before or after installing the Hub.

Installation media

The installation media is available on Github (<https://github.com/blackducksoftware/hub>).

Downloaded and unpack the `Hub.tar` file.

Distributions

Each orchestration tool has a directory with the files you need to install or upgrade 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](#) when using the database container provided by the Hub.
- `docker-compose.externaldb.yml`: Docker Compose file used with an [external PostgreSQL database](#).
- `docker-compose.yml`: Docker Compose file when using the database container provided by the Hub.
- `external-postgres-init.pgsql`: PostgreSQL.sql file used to configure an [external PostgreSQL database](#).
- `hub-postgres.env`: Environment file to configure an external PostgreSQL database.
- `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](#) when using the database container provided by the Hub.
- `hub_db_migrate.sh`: Script used to [migrate the PostgreSQL database](#) when using the database container provided by the Hub.
- `hub_reportdb_changepassword.sh`: Script used to set and change the report database password. Refer to the Report Database guide for more information.

Docker Run distribution

This is the content for the Docker Run distribution:

- `docker-hub.sh`: Docker Run script.
- `docker-run.sh`: Docker Run script.
- `docker-stop.sh`: Docker Run script.
- `external-postgres-init.pgsql`: PostgreSQL.sql file used to configure an [external PostgreSQL database](#).
- `hub-postgres.env`: Environment file to configure an external PostgreSQL database.
- `hub-proxy.env`: Environment file to [configure proxy settings](#).
- `hub-webserver.env`: Environment file to [configure web server settings](#).

See [Docker Run commands](#) for more information on the Docker Runs scripts.

In the `bin` directory:

- `hub_create_data_dump.sh`: Script used to back up the PostgreSQL database when using the database container provided by the Hub.
- `hub_db_migrate.sh`: Script used to [migrate the PostgreSQL database](#) when using the database container provided by the Hub.
- `hub_reportdb_changepassword.sh`: Script used to set and change the report database password. Refer to the Report Database guide for more information.

Installing the Hub

Prior to installing the Hub, determine if there are any [web server, proxy, or database settings](#) that need to be configured.

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

Note: These instructions are for new installations of the Hub. Refer to Chapter 6 for more information about [upgrading the Hub](#).

Docker Swarm:

- To install the Hub with the PostgreSQL database container:

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

- To install the Hub with an external PostgreSQL instance:

```
swarm init
docker stack deploy -c docker-compose.externaldb.yml hub
```

Note: 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 commands: `--with-registry-auth`

Docker Compose:

- To install the Hub with the PostgreSQL database container:

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

- To install the Hub with an external PostgreSQL instance:

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

Docker Run:

- To install the Hub with the PostgreSQL database container:

```
docker-run.sh 4.0.0
```


- To install the Hub with an external PostgreSQL instance:

```
docker-hub.sh -r 4.0.0 -u -e
```

The Black Duck Hub is installed.

You can confirm that the installation was successful by running the `docker -ps` command to view the status of each container. A "healthy" status indicates that the installation was successful. Note that the containers may be in a "starting" state for a few minutes post-installation.

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
```

A dialog box appears requesting a registration key. Enter the key provided to you to access the Hub.

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.
- Scale Job runners.
- Configure secure LDAP.

You can also modify the [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-key pair.

Docker Swarm:

1. Use the docker secret command to tell Docker Swarm the certificate and key by using HUB_WEBSERVER_CUSTOM_CERT_FILE and HUB_WEBSERVER_CUSTOM_KEY_FILE. The name of the secret must include the stack name. In the following example, the stack name is 'hub':

```
docker secret create hub_HUB_WEBSERVER_CUSTOM_CERT_FILE <certificate file>
```

```
docker secret create hub_HUB_WEBSERVER_CUSTOM_KEY_FILE <key file>
```

2. Add the secret to the services section of the Webserver service:

```
secrets:
  - HUB_WEBSERVER_CUSTOM_CERT_FILE
  - HUB_WEBSERVER_CUSTOM_KEY_FILE
```

Docker Compose:

1. Create a file named WEBSERVER_CUSTOM_CERT_FILE with the certificate file.
2. Create a file named WEBSERVER_CUSTOM_KEY_FILE with the key file.
3. Mount a directory that contains both files to /run/secrets in the Webserver container by editing the docker-compose.yml or docker-compose.externaldb.yml file.

```
webserver
  image: blackducksoftware/hub-nginx:4.0.0
```

```
ports: ['443:8443']
env_file: hub-webserver.env
links: [webapp, cfssl]
volumes: ['webserver-
volume:/opt/blackduck/hub/webserver/security',
'/directory/where/the/cert-key-files/are:/run/secrets']
```

4. Start the Webserver container.

Docker Run:

1. Create a file named `WEBSERVER_CUSTOM_CERT_FILE` with the certificate file.
2. Create a file named `WEBSERVER_CUSTOM_KEY_FILE` with the key file.
3. Mount a directory that contains both files to `/run/secrets`.

You can mount the volume by editing the `docker-hub.sh` script. Add the following line:

```
docker run -v /directory/where/the/cert-key-files/are:/run/secrets
```


to the Webserver section (`function startWebserver () {}`).

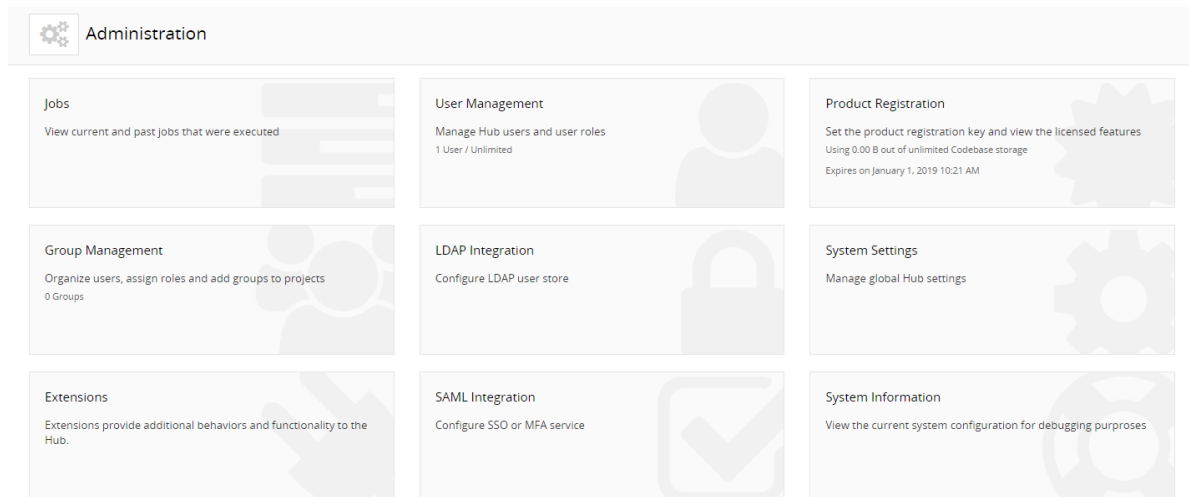
Accessing log files

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.

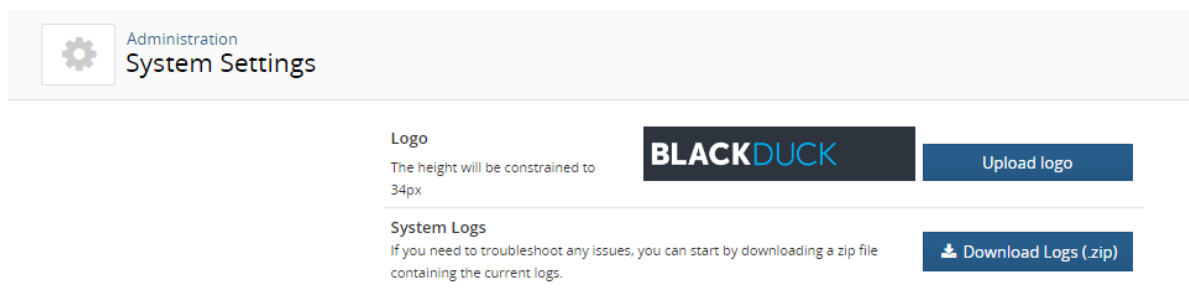
To download the log files from the Hub UI

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



3. Select **System Settings**.

The System Settings page appears.



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

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

Obtaining logs

To obtain logs from the containers:

```
docker cp <logstash container ID>:/var/lib/logstash/data logs/
```

where 'logs/' is a local directory where the logs will be copied into.

Viewing log files for a container

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

```
docker-compose logs
```

Scaling job runners

The Job Runner container can be scaled.

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
```

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 -f docker-compose.yml -p hub up --scale jobrunner=2 -d
```

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 -f docker-compose.yml -p hub up --scale jobrunner=1 -d
```

Configuring secure LDAP

If you see certificate issues when connecting your secure LDAP server to the Hub, the most likely reason is that the Hub server has not set up a trust connection to the secure LDAP sever. This usually occurs if you are using a self-signed certificate.

Configuring secure LDAP consists of:

1. Importing your certificate to the Hub keystore.
2. Testing the configuration.

Importing your certificate into the Hub

1. Export the appropriate (self-signed) certificate that can be used to set up a trusted client-server connection between the Hub and the secured LDAP server.
2. Find the ID of the Hub Web App container. For example:

```
docker ps -a | grep hub-webapp
```

3. Copy the certificate from the Docker host to the Hub Web App container:

```
docker cp ldap_certificate_name.crt Web App Container
ID:/opt/blackduck/hub/hub-webapp/security/ldap_certificate_name.crt
```

For example, for an LDAP certificate named `ldap_certificate` that needs to be copied to the Web App container with the ID of `7ca35ca6bc27`:

```
docker cp ldap_certificate.crt 7ca35ca6bc27:/opt/blackduck/hub/hub-
webapp/security/ldap_certificate.crt
```

4. From inside the container, use the Java keytool application to import the certificate into the `hub-webapp.truststore` truststore. The `hub-webapp.truststore` trust store is located at `/opt/blackduck/hub/hub-webapp/security/hub-webapp.truststore`.

The default password for the `hub-webapp.truststore` trust store is 'changeit'.

For example:

```
keytool -import -trustcacerts -keystore hub-webapp.truststore -storepass
changeit -alias ldaps -file ldaps.cer
```


Note: If you later change the truststore password, use the Docker secrets file (located at `/run/secrets/LDAP_TRUST_STORE_PASSWORD_FILE`) or use an environment variable: `LDAP_TRUST_STORE_PASSWORD` so that the Hub knows the new password.

5. Information about the certificate appears on the screen. When prompted, enter **yes** to trust this certificate.

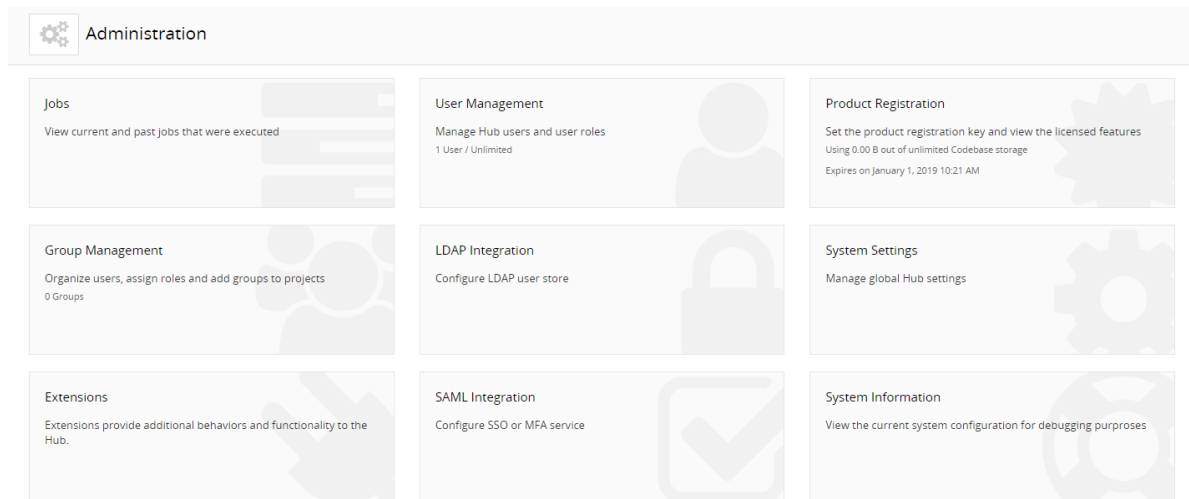
```
Trust this certificate? [no]:
```

6. Restart the Web App container.


Testing the configuration

1. Log in to the Hub as a system administrator.
2. Click the expanding menu icon () and select **Administration**.

The Administration page appears.



3. Select LDAP integration to display the LDAP Integration page.

 Administration
LDAP Integration

LDAP Server Details

Enable LDAP

☒ Enable LDAP

Server URL *

ldap://mamba-vm.blackducksoftware.com:389

Authentication Type *

Simple

Manager DN

cn=BlackDuck Manager,ou=blackduck,ou=People,dc=blackdi

Manager Password

Password already set, click to change it.

LDAP User Attributes

User Search Base *

ou=People,dc=blackducksoftware,dc=com

User Search Filter *

cn={0}

User DN Pattern

LDAP Attribute Mappings

First Name

givenName

Last Name

initials

Email

mail

LDAP Groups

Synchronize LDAP groups

☒ Synchronize LDAP groups

Group search base

ou=groups,dc=blackducksoftware,dc=com

Group filter

uniqueMember={0}

Group name attribute

cn

Save


Test Connection, User Authentication and Field Mapping

Tests ability to connect. Also tests ability to authenticate test-user and shows result of mapping test-user's meta-data. Note: test-user credentials are not saved.

Test Username *

Test Password *

Test Connection

 Test Connection

- Configure the LDAP server settings as described in the online help. In the **Server URL** field, ensure that you have configured the secure LDAP server: the protocol scheme is ldaps://.

5. Verify the connection to the secure directory server by entering the user credentials in the **Test Connection, User Authentication and Field Mapping** section and clicking **Test Connection**.

Chapter 5: Uninstalling the Hub

To uninstall the Hub, do one of the following:

- Stop and remove the containers and remove the volumes.

Docker Swarm:

```
docker stack rm hub
```

Docker Compose:

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

Docker Run:

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

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

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.

Docker Compose

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

Docker Run:

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

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: 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 your previous version of the Hub:

- AppMgr architecture
- Single-container AppMgr architecture
- Multi-container Docker

Upgrading from the AppMgr architecture

To upgrade from a previous version of the Hub based on the AppMgr architecture to the multi-container 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.

Migrating your PostgreSQL database

To use your existing PostgreSQL data you must migrate the database data which consists of:

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

⚙️ To back up the original PostgreSQL database

1. Log in to the Hub server as the **blackduck** 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 restore the PostgreSQL data

The process to restore the PostgreSQL data depends on whether you are using Docker Swarm, Docker Compose, or Docker Run to install 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 upgrade to the multi-image Docker version of the Hub.

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>
```

After stopping the containers, you can upgrade to 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 4.0.0 -m <path/to/dump/file>
```

You can now upgrade to 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

1. If you are installing the Hub on the same server that had the AppMgr version of the Hub installed on it:

- a. Run the `uninstall.sh` script to remove old files:

```
/opt/blackduck/hub/appmgr/bin/uninstall.sh
```

- b. As a root user or with `sudo` access, remove the autostart file. The `uninstall.sh` script states the location of the file at the end of the script run. For example:

```
rm -rf /etc/init.d/bds-hub-controller
```

2. Run one of the following commands using the files in the newer version of the Hub. The command depends on whether you are using Docker Swarm, Docker Compose, or Docker Run and whether you are using the DB container or an [external PostgreSQL instance](#):

Docker Swarm:

- Using the DB container: `docker stack deploy -c docker-compose.yml hub`
- Using an external PostgreSQL database: `docker stack deploy -c docker-compose.externaldb.yml hub`

Docker Compose:

- Using the DB container: `docker-compose -f docker-compose.yml -p hub up -d`
- Using an external PostgreSQL instance: `docker-compose -f docker-compose.externaldb.yml -p hub up -d`

Docker Run:

- Using the DB container: `docker-hub.sh -r 4.0.0 -u`
- Using an external PostgreSQL instance: `docker-hub.sh -r 4.0.0 -u -e`

Upgrading from a single-container AppMgr Hub

To upgrade from a previous version of the Hub based on the single-container AppMgr architecture to the multi-container 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.

Migrating your PostgreSQL database

To use your existing PostgreSQL data you must migrate the database data which consists of:

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

⚙ To back up the PostgreSQL database

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 restore the PostgreSQL data

The process to restore the PostgreSQL data depends on whether you are using Docker Swarm, Docker Compose, or Docker Run to install 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 upgrade to the multi-image Docker version of the Hub.

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>
```

After stopping the containers, you can upgrade to 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 4.0.0 -m <path/to/dump/file>
```

You can now upgrade to 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

1. Run one of the following commands using the files in the newer version of the Hub. The command depends on whether you are using Docker Swarm, Docker Compose, or Docker Run and whether you are using the DB container or an [external PostgreSQL instance](#):

Docker Swarm:

- Using the DB container: `docker stack deploy -c docker-compose.yml hub`
- Using an external PostgreSQL database: `docker stack deploy -c docker-compose.externaldb.yml hub`

Docker Compose:

- Using the DB container: `docker-compose -f docker-compose.yml -p hub up -d`
- Using an external PostgreSQL instance: `docker-compose -f docker-compose.externaldb.yml -p hub up -d`

Docker Run:

- Using the DB container: `docker-hub.sh -r 4.0.0 -u`
- Using an external PostgreSQL instance: `docker-hub.sh -r 4.0.0 -u -e`

Upgrading from an existing Docker architecture

To upgrade from a previous version of the Hub:

1. Back up your PostgreSQL database.

This is an optional step.

2. Upgrade the Hub.

Note: The method to configure custom SSL certificates for NGiNX has changed in 4.0.0. If you configured custom SSL certificates for NGiNX in the previous Docker release, you will need to [reconfigure them](#).

Backing up your PostgreSQL database

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:

```
./hub_create_data_dump.sh /destination/path
```

Upgrading the Hub

⚙ To upgrade the Hub from a previous version of the Hub:

1. Stop the existing containers.

Docker Compose:

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

Docker Run:

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

2. Run the following command using the files included in the newer version of the Hub. The command depends on whether you are using Docker Swarm, Docker Compose, or Docker Run and whether you are using the DB container or an [external PostgreSQL instance](#):

Docker Swarm:

- Using the DB container: `docker stack deploy -c docker-compose.yml hub`

- Using an external PostgreSQL instance: `docker stack deploy -c docker-compose.externaldb.yml hub`

Docker Compose:

- Using the DB container: `docker-compose -f docker-compose.yml -p hub up -d`
- Using an external PostgreSQL instance: `docker-compose -f docker-compose.externaldb.yml -p hub up -d`

Docker Run:

- Using the DB container: `docker-hub.sh -r 4.0.0 -u`
- Using an external PostgreSQL instance: `docker-hub.sh -r 4.0.0 -u -e`

Appendix A: Configuration settings

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

- `hub-webserver.env`
- `hub-proxy.env`
- `hub-postgres.env`

To configure the settings:

- 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, do the following:
 - For Docker Swarm, modify the settings and then redeploy the services in the stack by entering:
 - `docker stack deploy -c docker-compose.yml hub` if using the DB container
 - `docker stack deploy -c docker-compose.externaldb.yml hub` if using an external PostgreSQL instance
 - 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.
 3. Reconnect the containers.

Docker Compose:

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


Docker Run:

```
docker-run.sh
```

Configuring Web server settings

Edit the hub-webserver.env file to:

- Configure the hostname.
- Configure the host port.
- Disable IPv6.

Configuring the hostname

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. This value should be changed to the publically-facing hostname that users will enter in their browser in order to access Hub. For example:

```
PUBLIC_HUB_WEBSERVER_HOST=blackduck-docker01.dc1.lan
```

Configuring the host port

You can configure a different value for the host port which, by default, is 443.

To configure the host port

1. Modify the host port value defined in the following files.

For Docker Swarm or Docker Compose:

In the docker-compose.yml or docker-compose.externaldb.yml file, edit the first value shown in `ports: ['443:8443']` to the new port value.

```
webserver:
  image: blackducksoftware/hub-nginx:4.0.0
  ports: ['443:8443']
```

For example, to change the port to 8443:

```
webserver:
  image: blackducksoftware/hub-nginx:4.0.0
  ports: ['8443:8443']
```

For Docker Run:

In the docker-hub.sh or docker-run.sh file, edit the first value shown in `443:8443` found in the following line:

```
docker run -it -d --name webserver --link webapp --link cfssl -p 443:8443
```

\

For example, to change the port to 8443:

```
docker run -it -d --name webserver --link webapp --link cfssl -p 8443:8443
\
```

2. Edit the `PUBLIC_HUB_WEBSERVER_PORT` value in the `hub-webserver.env` file to the new port value. For example:

```
PUBLIC_HUB_WEBSERVER_PORT=8443
```

Disabling IPv6

By default, NGiNX listens on IPv4 and IPv6. If IPv6 is disabled on a host machine, change the value of the `IPV4_ONLY` environment variable to 1.

Configuring 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.

Proxy password

The following services require the proxy password:

- Web App
- Registration
- Job Runner

There are two methods for specifying a proxy password:

Docker Compose:

- Mount a directory that contains a text 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.

Docker Swarm:

For Docker Swarm, you can use the methods described for Docker Compose 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:

```
secrets:
- HUB_PROXY_PASSWORD_FILE
```

Docker Run:

- Mount a directory that contains a text file called `HUB_PROXY_PASSWORD_FILE` to `/run/secrets`. This is the most secure option.

Mount the volume by editing the `docker-hub.sh` script. Add the following line:

```
docker run -v /directory/where/file/is:/run/secrets
```

to the Web App (`function startWebapp () {}`), Registration (`function Registration () {}`), and Job runner (`function Jobrunner () {}`) sections in the `docker-hub.sh` script.

- Specify an environment variable called `HUB_PROXY_PASSWORD` that contains the proxy password. Add the following line

```
docker run -e HUB_PROXY_PASSWORD=password
```

to the `function startWebapp () {}`, `function Registration () {}`, and `function Jobrunner () {}` sections in the `docker-hub.sh` script.

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.

Configuring an external PostgreSQL instance

The Hub supports using an external PostgreSQL instance managed by Amazon Relational Database Service (RDS). Be sure that you have configured the instance as described below prior to installing or

upgrading the Hub.

⚙️ To configure an external PostgreSQL instance

1. Create a database user named **blackduck** with administrator privileges.

For Amazon RDS, set the "Master User" to **blackduck** when creating the database instance.

2. Run the `external-postgres-init.pgsql` script, located in the orchestration directory you used to install the Hub, to create users, databases, and other necessary items. For example:

```
psql -U blackduck -h <hostname> -p <port> -f external_postgres_init.pgsql postgres
```

3. Using your preferred PostgreSQL administration tool, configure passwords for the **blackduck**, **blackduck_user**, and **blackduck_reporter** database users.

These users were created by the `external-postgres-init.pgsql` script in the previous step.

4. Edit the `hub-postgres.env` environment file, as described [here](#), to specify the database connection parameters:

Parameter	Description
HUB_POSTGRES_ENABLE_SSL	Forces the use of SSL in database connections. As Amazon RDS automatically uses SSL, this must be set to "false".
HUB_POSTGRES_HOST	Hostname of the server with the PostgreSQL instance.
HUB_POSTGRES_PORT	Database port to connect to for the PostgreSQL instance.
HUB_POSTGRES_USER	Database username. By default, this is set to blackduck_user .
HUB_POSTGRES_ADMIN	Database administrator. By default, this is set to blackduck .

5. Provide the **blackduck** and **blackduck_user** passwords to the Hub:

Docker Compose:

- a. Create a file named `HUB_POSTGRES_USER_PASSWORD_FILE` with the password for the **blackduck_user** user.
- b. Create a file named `HUB_POSTGRES_ADMIN_PASSWORD_FILE` with the password for the **blackduck** user.
- c. Mount a directory that contains both files to `/run/secrets` in both the Web App and Job runner containers by editing the `docker-compose.externaldb.yml` file.

Docker Swarm:

You can use the method described for Docker Compose or use the `docker secret` command to create a secret called `HUB_POSTGRES_USER_PASSWORD_FILE` and a secret called `HUB_POSTGRES_`

ADMIN_PASSWORD_FILE.

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

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

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

- b. Add the password secret to the services section of the Web App and Job Runner services:

```
secrets:  
  - HUB_POSTGRES_USER_PASSWORD_FILE  
  - HUB_POSTGRES_ADMIN_PASSWORD_FILE
```

Docker Run:

- Create a file name HUB_POSTGRES_USER_PASSWORD_FILE with the password for the **blackduck** user.
- Create a file named HUB_POSTGRES_ADMIN_PASSWORD_FILE with the password for the **blackduck_user** user.

6. [Install](#) or [upgrade](#) the Hub.

Appendix B: 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, `4.0.0`.

To start the Hub:

```
docker-run.sh 4.0.0
```

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.
-e, --externaldb	Use an external PostgreSQL instance rather than the default Docker container. The --externaldb parameter cannot be run in the same command as the --migrate parameter.
-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 4.0.0 -u
```

Start the Hub using an external PostgreSQL instance:

```
docker-hub.sh -r 4.0.0 -u -e
```

Stop the Hub:

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

Stop the Hub with an external PostgreSQL instance:

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

Uninstalling the Hub but leaving volumes in place:

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

Uninstalling the Hub with an external PostgreSQL instance but leaving volumes in place:

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

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

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

Uninstalling the Hub with an external PostgreSQL instance and removing volumes. This removes *all* Docker-managed data; the external PostgreSQL instance is not affected.

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

Appendix C: Docker containers

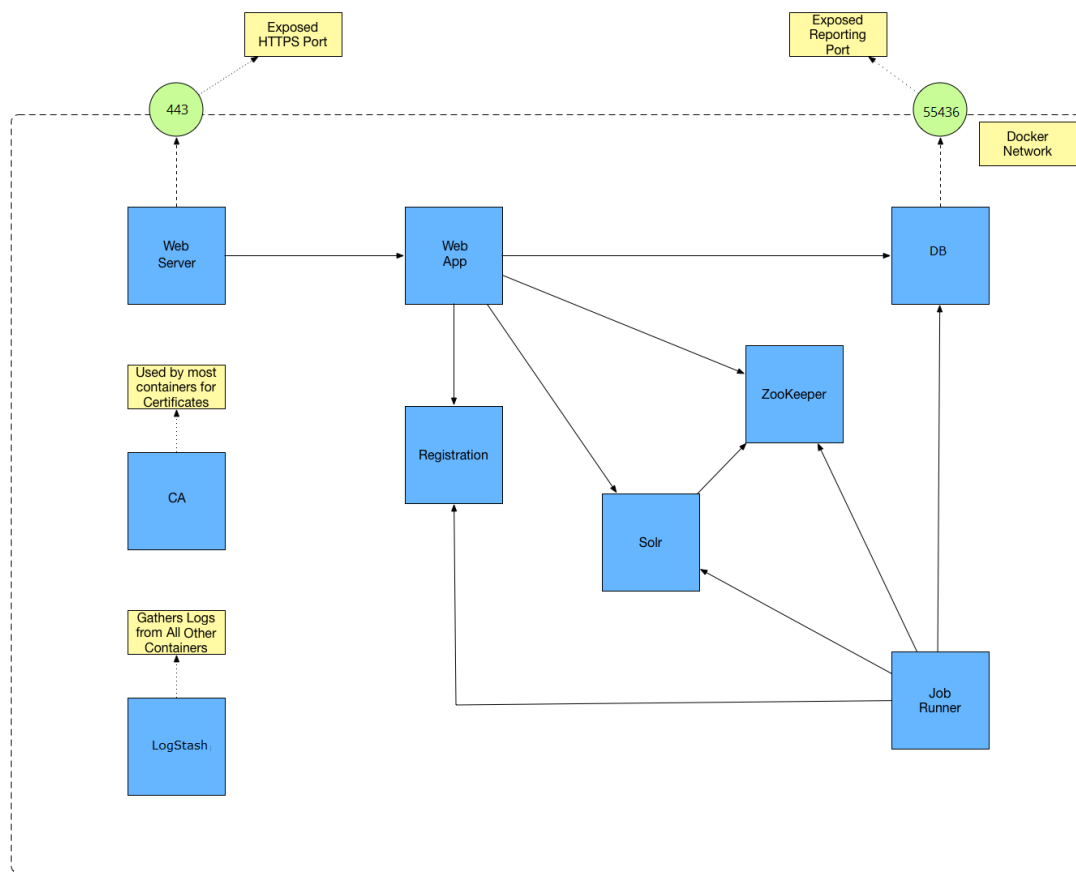
These are the containers within the Docker network that comprise the Hub application:

1. Web App
2. Job Runner App
3. Solr
4. Registration
5. DB

Note: This container is not included in the Hub application if you use an [external Postgres instance](#).

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. It includes a PostgreSQL database container.



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:4.0.0
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	<p>The Web App container needs to connect to these containers/services:</p> <ul style="list-style-type: none"> • postgres • solr • zookeeper • registration • logstash • cfssl <p>The container needs to expose port 8080 to other containers that will link to it.</p>
Alternate Host Name Environment Variables	<p>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:</p> <ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • Default max Java heap size: 4GB • Container memory: 4GB • Container CPU: 1 CPU

Container Name: Web App	
Volumes	log-volume:/opt/blackduck/hub/logs webapp-volume:/opt/blackduck/hub/hub-webapp/security
Environment File	hub-proxy.env

Job runner container

Container Name: Job Runner	
Image Name	blackducksoftware/hub-jobrunner:4.0.0
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: <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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:4.0.0
Description	<p>Solr is an open source enterprise search platform. The Hub uses Solr as its search server for project data.</p> <p>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.</p>
Scalability	This container should not be scaled.
Links/Ports	<p>The Solr container needs to connect to these containers/services:</p> <ul style="list-style-type: none"> • zookeeper • logstash <p>The container needs to expose port 8080 to other containers that will link to it.</p>
Alternate Host Name Environment Variables	<p>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:</p> <ul style="list-style-type: none"> • zookeeper: \$HUB_ZOOKEEPER_HOST • logstash: \$HUB_LOGSTASH_HOST
Constraints	<ul style="list-style-type: none"> • 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:4.0.0
Description	<p>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.</p>
Scalability	The container should not be scaled.

Container Name: Registration	
Links/Ports	<p>The Registration container needs to connect to this containers/services:</p> <ul style="list-style-type: none"> • logstash <p>The container needs to expose port 8080 to other containers that link to it.</p>
Alternate Host Name Environment Variables	<p>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:</p> <ul style="list-style-type: none"> • logstash: \$HUB_LOGSTASH_HOST
Constraints	<ul style="list-style-type: none"> • 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

Note: This container is included in the Hub application if you use an [external Postgres instance](#).

Container Name: DB	
Image Name	blackducksoftware/hub-postgres:4.0.0
Description	<p>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.</p> <p>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 <code>hub_reportdb_changepassword.sh</code> script. This port and user can be used for reporting and data extraction.</p> <p>Refer to the <i>Report Database</i> guide for more information on the report database.</p>
Scalability	There should only be a single instance of this container. It should not be scaled.

Container Name: DB	
Links/Ports	<p>The DB container needs to connect to these containers/services:</p> <ul style="list-style-type: none"> • logstash • cfssl <p>The container needs to expose port 5432 to other containers that will link to it within the Docker network.</p> <p>This container exposes port 55436 outside of the Docker network for database reporting.</p>
Alternate Host Name Environment Variables	<p>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:</p> <ul style="list-style-type: none"> • logstash: \$HUB_LOGSTASH_HOST • cfssl: \$HUB_CFSSL_HOST
Constraints	<ul style="list-style-type: none"> • 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:4.0.0
Description	<p>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.</p>
Scalability	The container should not be scaled.
Links/Ports	<p>The Web App container needs to connect to these containers/services:</p> <ul style="list-style-type: none"> • webapp • cfssl <p>This container exposes port 443 outside of the Docker network.</p>

Container Name: WebServer	
Alternate Host Name Environment Variables	<p>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:</p> <ul style="list-style-type: none"> • webapp: \$HUB_WEBAPP_HOST • cfssl: \$HUB_CFSSL_HOST
Constraints	<ul style="list-style-type: none"> • 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:4.0.0
Description	<p>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.</p>
Scalability	This container should not be scaled.
Links/Ports	<p>The Zookeeper container needs to connect to this container/service:</p> <ul style="list-style-type: none"> • logstash <p>The container needs to expose port 2181 within the Docker network to other containers that will link to it.</p>
Alternate Host Name Environment Variables	<p>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:</p> <ul style="list-style-type: none"> • logstash: \$HUB_LOGSTASH_HOST
Constraints	<ul style="list-style-type: none"> • 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:4.0.0
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	<ul style="list-style-type: none"> • 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:4.0.0
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	<ul style="list-style-type: none"> • Default max Java heap size: N/A • Container memory: 512MB • Container CPU: Unspecified
Volumes	cert-volume:/etc/cfssl
Environment File	N/A