



# Installing Black Duck using Docker Swarm

Version 5.0.0



This edition of the *Installing Black Duck using Docker Swarm* refers to version 5.0.0 of Black Duck.

This document created or updated on Thursday, September 20, 2018.

**Please send your comments and suggestions to:**

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

Copyright © 2018 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.

<b>Chapter 1: Overview</b>	<b>1</b>
Black Duck Architecture	1
Components hosted on Black Duck servers	1
<b>Chapter 2: Installation planning</b>	<b>2</b>
Getting started	2
New installations	2
Upgrading from a previous version of Black Duck	2
Hardware requirements	2
Docker requirements	3
Operating systems	3
Software requirements	4
Network requirements	4
Database requirements	4
Proxy server requirements	5
Configuring your NGiNX server to work with Black Duck	5
Amazon services	7
Additional port information	7
<b>Chapter 3: Installing Black Duck</b>	<b>9</b>
Installation files	10
Download from the GitHub page	10
Download using the wget command	10
Distribution	10
Installing Black Duck	11
<b>Chapter 4: Administrative tasks</b>	<b>13</b>
Understanding the default sysadmin user	14
Configuring Web server settings	14
Configuring the hostname	14
Configuring the host port	14
Disabling IPv6	15
Configuring Proxy settings	15
Proxy password	16
Importing a proxy certificate	17
Configuring an external PostgreSQL instance	17

Managing certificates .....	19
Using custom certificates .....	20
Accessing log files .....	20
Obtaining logs .....	21
Viewing log files for a container .....	21
Purging logs .....	21
Scaling job runner, scan, and binaryscanner containers .....	22
Scaling job runner containers .....	22
Scaling scan containers .....	22
Scaling binaryscanner containers .....	22
Changing the default memory limits .....	23
Changing the default Web App container memory limits .....	23
Changing the default Job Runner container memory limits .....	24
Changing the default Scan container memory limits .....	25
Changing the default binaryscanner container memory limits .....	27
Configuring the report database password .....	27
Accessing the API documentation through a proxy server .....	28
Providing access to the REST APIs from a non-Black Duck server .....	28
Configuring secure LDAP .....	29
Obtaining your LDAP information .....	29
Importing the server certificate .....	30
LDAP trust store password .....	32
Configuring SAML for Single Sign-On .....	33
Providing your Black Duck system information to Customer Support .....	35
Customizing user IDs of Black Duck containers .....	35
Including ignored components in reports .....	37
Enabling the hierarchical BOM .....	37
Increasing the size of the binary scan file .....	37
Starting or stopping Black Duck .....	38
Starting up Black Duck .....	38
Shutting down Black Duck .....	38
<b>Chapter 5: Uninstalling Black Duck .....</b>	<b>39</b>
<b>Chapter 6: Upgrading Black Duck .....</b>	<b>40</b>
Installation files .....	40
Download from the GitHub page .....	40
Download using the wget command .....	40
Upgrading from the AppMgr architecture .....	41
Migrating your PostgreSQL database .....	41
Upgrading Black Duck .....	42
Upgrading from a single-container AppMgr architecture .....	43
Migrating your PostgreSQL database .....	43

Upgrading Black Duck .....	44
Upgrading from an existing Docker architecture .....	44
Migrating your PostgreSQL database .....	45
Upgrading Black Duck .....	46
<b>Appendix A: Docker containers .....</b>	<b>47</b>
Webapp container .....	48
Authentication container .....	50
Scan container .....	51
Job runner container .....	52
Solr container .....	53
Registration container .....	54
DB container .....	55
Documentation container .....	56
WebServer container .....	57
ZooKeeper container .....	59
Logstash container .....	60
CA container .....	60
Black Duck - Binary Analysis containers .....	61
Rabbitmq container .....	61
Uploadcache container .....	62
Binaryscanner container .....	63

## Black Duck documentation

The documentation for Black Duck consists of online help and these documents:

Title	File	Description
Release Notes	release_notes.pdf	Contains information about the new and improved features, resolved issues, and known issues in the current and previous releases.
Installing Black Duck using Docker Compose	install_compose.pdf	Contains information about installing and upgrading Black Duck using Docker Compose.
Installing Black Duck using Docker Swarm	install_swarm.pdf	Contains information about installing and upgrading Black Duck using Docker Swarm.
Installing Black Duck using Kubernetes	install_kubernetes.pdf	Contains information about installing and upgrading Black Duck using Kubernetes.
Installing Black Duck using OpenShift	install_openshift.pdf	Contains information about installing and upgrading Black Duck using OpenShift.
Getting Started	getting_started.pdf	Provides first-time users with information on using Black Duck.
Scanning Best Practices	scanning_best_practices.pdf	Provides best practices for scanning.
Getting Started with the SDK	getting_started_sdk.pdf	Contains overview information and a sample use case.

Title	File	Description
Report Database	report_db.pdf	Contains information on using the report database.
User Guide	user_guide.pdf	Contains information on using Black Duck's UI.

Black Duck integration documentation can be found on [Confluence](#).

## Training

Synopsys Software Integrity, Customer Education (SIG Edu) 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 Synopsys Software Integrity, Customer Education (SIG Edu), 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://education.synopsys.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](mailto:communityfeedback@blackducksoftware.com) or call us at +1 781.891.5100 ext. 5.

To see all the ways you can interact with Black Duck Support, visit:

<https://www.blackducksoftware.com/support/contact-support>.



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

## Black Duck Architecture

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

The Docker architecture brings these significant improvements to Black Duck:

- 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 Black Duck application.

Visit the Docker website: <https://www.docker.com/> for more information on Docker.

To obtain Docker installation information, go to <https://docs.docker.com/engine/installation/>.

## Components hosted on Black Duck servers

The following remote Black Duck services are leveraged by Black Duck:

- Registration server: Used to validate Black Duck's license.
- Black Duck KnowledgeBase server: The Black Duck KnowledgeBase (KB) is the industry's most comprehensive database of open source project, license, and security information. Leveraging the Black Duck KB in the cloud ensures that Black Duck can display the most up-to-date information about open source software (OSS) without requiring regular updates to your Black Duck installation.

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

### Getting started

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

#### New installations

For new installation of Black Duck:

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 administrative tasks.

#### Upgrading from a previous version of Black Duck

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 administrative tasks.

### Hardware requirements

The following is the minimum hardware that is needed to run a single instance of all containers:

- 5 CPUs
- 20 GB RAM
- 250 GB of free disk space for the database and other Black Duck containers
- Commensurate space for database backups

The following is the minimum hardware that is needed to run Black Duck with Black Duck - Binary Analysis:

- 6 CPUs
- 24 GB RAM

- 350 GB of free disk space for the database and other Black Duck containers
- Commensurate space for database backups

**Note:** An additional CPU, 2 GB RAM, and 100 GB of free disk space is needed for every [additional binaryscanner container](#).

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, scan, and binaryscanner containers).

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

**Note:** Installing Black Duck Alert requires 1 GB of additional memory.

## Docker requirements

Docker Swarm, which is the preferred method for installing Black Duck, is a clustering and scheduling tool for Docker containers. With Docker Swarm, you can manage a cluster of Docker nodes as a single virtual system.

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

There are two restrictions when using Black Duck 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 the hub-webapp service can access the logs that need to be downloaded.

## Docker Version

Black Duck installation supports Docker versions 17.06.x, 17.09.x, 17.12.x, and 18.03.x (CE or EE).

## Operating systems

The preferred operating systems for installing Black Duck in a Docker environment are:

- CentOS 7.3
- Red Hat Enterprise Linux server 7.3
- Ubuntu 16.04.x
- SUSE Linux Enterprise server version 12.x (64-bit)
- Oracle Enterprise Linux 7.3

In addition, Black Duck supports other Linux operating systems that support Docker version 17.06.x (CE or EE).

**Note:** Docker CE does not support Red Hat Enterprise Linux. Click [here](#) for more information.

Windows operating system is currently not supported.

## Software requirements

Black Duck 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 Black Duck:

- Chrome 69.0.3497.81 (Official Build) (64-bit)
- Firefox 62.0 (64-bit)
- Internet Explorer 11.1266.15063.0
- Microsoft Edge 40.15063.674.0
- Microsoft EdgeHTML 15.15063
- Safari 11.1.2 (13605.3.8)

Note that Black Duck does not support compatibility mode.

**Note:** These browser versions are the currently-released versions on which Black Duck Software has tested Black Duck. Newer browser versions may be available after Black Duck 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.

## Network requirements

Black Duck requires the following ports to be externally accessible:

- Port 443 – Web server HTTPS port for Black Duck 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 Black Duck installation to Black Duck Software 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)

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

## Database requirements

Black Duck uses the PostgreSQL object-relational database to store data.

Prior to installing Black Duck, determine whether you want to use the database container that is automatically

installed or an external (for example, Amazon Relational Database Service (RDS)) 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 Black Duck.

Currently, Black Duck requires PostgreSQL 9.6.X.

## PostgreSQL versions

For Black Duck version 5.0.0, the currently-supported version of PostgreSQL is 9.6.x, which is the version supplied in Black Duck's PostgreSQL container. If you choose to run your own PostgreSQL instance, you must be at PostgreSQL version 9.6.x for compatibility with Black Duck version 5.0.0.

Refer to [Chapter 6, Upgrading Black Duck](#) for database migration instructions if upgrading from a pre-4.2.0 version of Black Duck.

## Proxy server requirements

Black Duck supports:

- No Authentication
- Digest
- Basic
- NTLM

If you are going to make proxy requests to Black Duck, 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.

## Configuring your NGINX server to work with Black Duck

If you have an NGINX server acting as an HTTPS server/proxy in front of Black Duck, you must modify the NGINX configuration file so that the NGINX server passes the correct headers to Black Duck. Black Duck 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 Black Duck, 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.conf` configuration file:

```
location/prefixPath {

    proxy_set_header X-Forwarded-Prefix "/prefixPath";

}
```

To scan files successfully, you must use the **context** parameter when using the command line or include it in the **Black Duck Server URL** field in the Black Duck 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 Black Duck, 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.  <b>Example:</b> 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.  <b>Example:</b> X-Forwarded-Port: "9876"

X-Forwarded-Proto	<p>Contains a single value representing the protocol scheme used for the original request.</p> <p><b>Example:</b></p> <p>X-Forwarded-Proto: "https"</p>
X-Forwarded-Prefix	<p>Contains a prefix path used for the original request.</p> <p><b>Example:</b></p> <p>X-Forwarded-Prefix: "prefixPath"</p> <p>To successfully scan files, you must use the <b>context</b> parameter</p>

## Amazon services

You can:

- Install Black Duck on Amazon Web Services (AWS)  
Refer to your [AWS documentation](#) and your [AMI documentation](#) for more information on AWS.
- Use Amazon Relational Database Service (RDS) for the PostgreSQL database that is used by Black Duck.  
Refer to your [Amazon Relational Database Service documentation](#) for more information on Amazon RDS.  
Currently Black Duck requires PostgreSQL version 9.6.x.

## Additional port information

The following list of ports cannot be blocked by firewall rules or by your Docker configuration. Examples of how these ports may be blocked include:

- The `iptables` configuration on the host machine.
- A `firewalld` configuration on the host machine.
- External firewall configurations on another router/server on the network.
- Special Docker networking rules applied above and beyond what Docker creates by default, and also what Black Duck creates by default.

The complete list of ports that must remain unblocked is:

- 443
- 8443
- 8000
- 8888
- 8983
- 16543
- 17543

- 16545
- 16544
- 55436



## Chapter 3: Installing Black Duck

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

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

## Installation files

The installation files are available on GitHub.

Download the orchestration files. As part of the install/upgrade process, these orchestration files pull down the necessary Docker images.

Note that although the filename of the `tar.gz` file differs depending on how you access the file, the content is the same.

### Download from the GitHub page

1. Select the link to download the `.tar.gz` file from the GitHub page:

<https://github.com/blackducksoftware/hub>.

2. Uncompress the Black Duck `.gz` file:

```
gunzip hub-5.0.0.tar.gz
```

3. Unpack the Black Duck `.tar` file:

```
tar xvf hub-5.0.0.tar
```

### Download using the wget command

1. Run the following command:

```
wget https://github.com/blackducksoftware/hub/archive/v5.0.0.tar.gz
```

2. Uncompress the Black Duck `.gz` file:

```
gunzip v5.0.0.tar.gz
```

3. Unpack the Black Duck `.tar` file:

```
tar xvf v5.0.0.tar
```

## Distribution

The `docker-swarm` directory consists of following files you need to install or upgrade Black Duck.

- `docker-compose.bdba.yml`: Docker Compose file used when installing Black Duck with Black Duck - Binary Analysis and using the database container provided by Black Duck.
- `docker-compose.dbmigrate.yml`: Docker Compose file used to migrate the PostgreSQL database when using the database container provided by Black Duck.
- `docker-compose.externaldb.bdba.yml`: Docker Compose file used when installing Black Duck with Black Duck - Binary Analysis and using an external PostgreSQL database.
- `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 Black Duck.
- `external-postgres-init.pgsql`: PostgreSQL.sql file used to configure an external PostgreSQL

database.

- `hub-bdba.env`: Environment file that contains additional settings for Black Duck - Binary Analysis. This file should not require any modification.
- `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 Black Duck.
- `hub_db_migrate.sh`: Script used to migrate the PostgreSQL database when using the database container provided by Black Duck.
- `hub_reportdb_changepassword.sh`: Script used to set and [change the report database password](#).
- `system_check.sh`: Script used to [gather your Black Duck system information](#) to send to Customer Support.

## Installing Black Duck

These instructions only apply to installing Black Duck using Docker Swarm.

Prior to installing Black Duck, determine if there are any [settings](#) that need to be configured.

To install Black Duck, 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 Black Duck. Refer to Chapter 6 for more information about [upgrading Black Duck](#).

Separate `.yaml` files are included if you are installing Black Duck with Black Duck - Binary Analysis. The `docker-compose.bdba.yaml` file installs Black Duck with Black Duck - Binary Analysis and with the PostgreSQL database container, The `docker-compose.externaldb.bdba.yaml` file installs Black Duck with Black Duck - Binary Analysis and with an external PostgreSQL instance.

In the instructions below, use the `.yaml` file located in the `docker-swarm` directory.

- To install Black Duck with the PostgreSQL database container:

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

The `docker swarm init` command creates a single-node swarm.

- To install Black Duck with Black Duck - Binary Analysis using the PostgreSQL database container:

```
docker swarm init
docker stack deploy -c docker-compose.bdba.yaml hub
```

The `docker swarm init` command creates a single-node swarm.

- To install Black Duck with an external PostgreSQL instance:

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

The `docker swarm init` command creates a single-node swarm.

- To install Black Duck with Black Duck - Binary Analysis using an external PostgreSQL instance:

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

The `docker swarm init` command creates a single-node swarm.

**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`.

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 Black Duck are up, the web application for Black Duck will be exposed on port 443 to the docker host. Be sure that you have configured the [hostname](#) and then you can access Black Duck by entering the following:

`https://hub.example.com`

The first time you access Black Duck, the Registration & End User License Agreement appears. You must accept the terms and conditions to use Black Duck.

Enter the registration key provided to you to access Black Duck.

**Note:** If you need to reregister, you must accept the terms and conditions of the End User License Agreement again.

## Chapter 4: Administrative tasks

This chapter describes these administrative tasks:

- [Understanding the default sysadmin user.](#)
- [Configuring web server settings](#), such as configuring the hostname, host port, or disabling IPv6.
- [Configuring proxy settings.](#)
- [Configuring an external PostgreSQL instance.](#)
- [Replacing the existing self-signed certificate](#) for the Web Server with a custom certificate.
- [Accessing log files.](#)
- [Scaling job runner, scan and binaryscanner containers.](#)
- [Changing the default memory limits.](#)
- [Configuring the report database password.](#)
- [Providing access to the API documentation through a proxy server.](#)
- [Providing access to the REST APIs from a non-Black Duck server.](#)
- [Configuring secure LDAP.](#)
- [Configuring Single Sign-On \(SSO\).](#)
- [Providing your Black Duck system information to Customer Support.](#)
- [Customizing user IDs of Black Duck containers](#)
- [Including ignored components in reports](#)
- [Enabling the hierarchical BOM](#)
- [Increasing the size of the binary scan file](#)
- [Starting or stopping Black Duck](#)

### Using environment files

Note that some configurations use environment files; for example, configuring web server, proxy, or external PostgreSQL settings. The environment files to configure these settings are located in the `docker-swarm` directory.

To configure settings that use environment files:

- To set configuration settings *before* installing Black Duck, edit the file as described below and save your changes.
- To modify existing settings *after* installing Black Duck, 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.bdba.yml hub` if using the DB container with Black Duck - Binary Analysis
- `docker stack deploy -c docker-compose.externaldb.yml hub` if using an external PostgreSQL instance
- `docker stack deploy -c docker-compose.externaldb.bdba.yml hub` if using an external PostgreSQL instance with Black Duck - Binary Analysis

**Note:** Use the version of the `.yml` file located in the `docker-swarm` directory.

## Understanding the default sysadmin user

When you install Black Duck, 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. To change your password, select **My Profile** from your username/user profile icon in the upper right corner of the Black Duck UI.

## 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 hostname. This value should be changed to the publicly-facing hostname that users will enter in their browser to access Black Duck. 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.

In one of the following files located in the `docker-swarm` directory:

- `docker-compose.yml` (using the DB container)
- `docker-compose.bdba.yml` (using the DB container with Black Duck - Binary Analysis)
- `docker-compose.externaldb.yml` (using an external PostgreSQL instance)
- `docker-compose.externaldb.bdba.yml` (using an external PostgreSQL instance with Black Duck - Binary Analysis)

edit the first value shown in `ports: [ '443:8443' ]` to the new port value.

```
webserver:
  image: blackducksoftware/hub-nginx:5.0.0
  ports: [ '443:8443' ]
```

For example, to change the port to 8443:

```
webserver:
  image: blackducksoftware/hub-nginx:5.0.0
  ports: [ '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.

These are the containers that need access to services hosted by Black Duck Software:

- Authentication
- Registration
- Job runner
- Web app
- Scan

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:

- Authentication
- Web App
- Registration
- Job Runner
- Scan

There are three methods for specifying a proxy password:

- 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.
- Use the docker secret command to create a secret called HUB\_PROXY\_PASSWORD\_FILE as described below:

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 Authentication, Web App, Registration, Job Runner, and Scan services:

```
secrets:
  - HUB_PROXY_PASSWORD_FILE
```

Add text such as the following to the end of the compose file:

```
secrets:
  HUB_PROXY_PASSWORD_FILE:
    external:
      name: "hub_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.

## Importing a proxy certificate

You can import a proxy certificate to work with the proxy.

1. Create a docker secret called '`<stack name>_HUB_PROXY_CERT_FILE`' with the proxy certificate file. For example

```
docker secret create <stack name>_HUB_PROXY_CERT_FILE <certificate file>
```

2. Add the following to the services section of the authentication, jobrunner, scan, registration, and webapp services:

```
...
secrets:
  - HUB_PROXY_CERT_FILE
...
```

## Configuring an external PostgreSQL instance

This section describes how to configure an external PostgreSQL instance.

Black Duck 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 Black Duck.

### 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.

No other specific values are required.

2. Run the `external-postgres-init.pgsql` script, located in the `docker-swarm` directory, to install Black Duck, 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 to specify the database connection parameters:

Parameter	Description
HUB_POSTGRES_ENABLE_SSL	Forces the use of SSL in database connections.  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.

5. Provide the **blackduck** and **blackduck\_user** passwords to Black Duck:
- Create a file named `HUB_POSTGRES_USER_PASSWORD_FILE` with the password for the **blackduck\_user** user.
  - Create a file named `HUB_POSTGRES_ADMIN_PASSWORD_FILE` with the password for the **blackduck** user.
  - Mount a directory that contains both files to `/run/secrets` in the Web App, Job runner, Authentication, and Scan containers by editing the `docker-compose.externaldb.yml` file or the `docker-compose.externaldb.bdba.yml` file (with Black Duck - Binary Analysis) located in the `docker-swarm` directory.

Instead of Steps 5a-c, you can use the `docker secret` command to create a secret called `HUB_POSTGRES_USER_PASSWORD_FILE` and a secret called `HUB_POSTGRES_ADMIN_PASSWORD_FILE`.

- 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>
```

- Add the password secret to the services section of the Web App, Job Runner, Authentication, and Scan services:

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

Add text such as the following to the end of the compose file:

```
secrets:
  HUB_POSTGRES_USER_PASSWORD_FILE:
    external:
      name: "hub_HUB_POSTGRES_USER_PASSWORD_FILE"
  HUB_POSTGRES_ADMIN_PASSWORD_FILE:
    external:
      name: "hub_HUB_POSTGRES_ADMIN_PASSWORD_FILE"
```

6. [Install](#) or [upgrade](#) Black Duck.

## Managing certificates

By default, Black Duck uses an HTTPS connection. The default certificate used to run HTTPS is a self-signed certificate which means that it was created locally and was not signed by a recognized Certificate Authority (CA).

If you use this default certificate, you will need to make a security exception to log in to Black Duck's UI, as your browser does not recognize the issuer of the certificate, so it is not accepted by default.

You will also receive a message regarding the certificate when connecting to the Black Duck server when scanning as the Black Duck Scanner cannot verify the certificate because it is a self-signed and is not issued by a CA. Note that the Black Duck Scanner Desktop does provide an option that allows you to connect to a Black Duck instance with a self-signed certificate.

You can obtain a signed SSL certificate from a Certificate Authority of your choice. To obtain a signed SSL certificate, create a Certificate Signing Request (CSR), which the CA then uses to create a certificate that will identify the server running your Black Duck instance as "secure". After you receive your signed SSL certificate from the CA, you can replace the self-signed certificate.

### ⚙️ To create an SSL certificate keystore

1. At the command line, to generate your SSL key and a CSR, type:

```
openssl genrsa -out <keyfile> <keystrength>
openssl req -new -key <keyfile> -out <CSRfile>
```

where:

- **<keyfile>** is <your company's server name>.key
- **<keystrength>** is the size of your site's public encryption key
- **<CSRfile>** is <your company's server name>.csr

**Note:** It is important that the name entered for your company's server be the full hostname that your SSL server will reside on, and that the organization name be identical to what is in the 'whois' record for the domain.

For example:

```
openssl genrsa -out server.company.com.key 1024
openssl req -new -key server.company.com.key -out server.company.com.csr
```

This example creates a CSR for server.company.com to get a certificate from the CA.

2. Send the CSR to the CA by their preferred method (usually through a web portal).
3. Indicate that you need a certificate for an Apache web server.
4. Provide any requested information about your company to the CA. This information must match your domain registry information.
5. Once you receive your certificate from the CA, use the instructions in the next section to upload the certificate into a Black Duck instance.

## 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.

1. Use the docker secret command to tell Docker Swarm the certificate and key by using WEBSERVER\_CUSTOM\_CERT\_FILE and 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_WEBSERVER_CUSTOM_CERT_FILE <certificate file>
docker secret create hub_WEBSERVER_CUSTOM_KEY_FILE <key file>
```

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

```
secrets: [WEBSERVER_CUSTOM_CERT_FILE, WEBSERVER_CUSTOM_KEY_
FILE]
```

Add text such as the following to the end of the compose file:

```
secrets:
  WEBSERVER_CUSTOM_CERT_FILE:
    external:
      name: "hub_WEBSERVER_CUSTOM_CERT_FILE"
  WEBSERVER_CUSTOM_KEY_FILE:
    external:
      name: "hub_WEBSERVER_CUSTOM_KEY_FILE"
```

2. The healthcheck property in the webserver service must point to the new certificate from the secret:

```
healthcheck:
  test: [CMD, /usr/local/bin/docker-healthcheck.sh,
'https://localhost:8443/health-checks/liveness',
/run/secrets/WEBSERVER_CUSTOM_CERT_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 Black Duck UI

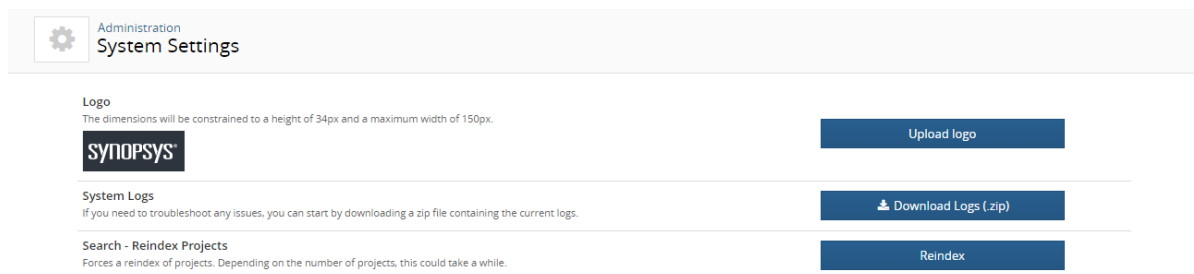
1. Log in to Black Duck with the System Administrator role.

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
```

For more information on Docker commands, visit the Docker documentation website: <https://docs.docker.com/>

### Purging logs

By default, log files are automatically purged after 30 days. To modify this value:

1. Stop the containers.
2. Edit the existing value of `DAYS_TO_KEEP_LOGS` in one of the following files located in the `docker-`

swarm directory:

- `docker-compose.yml` (Using the DB container)
- `docker-compose.bdba.yml` (Using the DB container with Black Duck - Binary Analysis)
- `docker-compose.externaldb.yml` (Using an external PostgreSQL instance)
- `docker-compose.externaldb.bdba.yml` (Using an external PostgreSQL instance with Black Duck - Binary Analysis)

3. Restart the containers.

## Scaling job runner, scan, and binaryscanner containers

The job runner, scan, and binaryscanner containers can be scaled.

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

### Scaling job runner containers

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 runner containers. The following example scales back the job runner container to a single container:

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

### Scaling scan containers

This example adds a second Scan container:

```
docker service scale hub_scan=2
```

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

```
docker service scale hub_scan=1
```

### Scaling binaryscanner containers

Binaryscanner containers are used with Black Duck - Binary Analysis.

This example adds a second binaryscanner container:

```
docker service scale appcheck-worker=2
```

**Note:** An additional CPU, 2 GB RAM, and 100 GB of free disk space is needed for every additional binaryscanner container.

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

```
docker service scale appcheck-worker=1
```

## Changing the default memory limits

There are some containers that may require higher than default memory limits depending on the load placed on Black Duck.

**Note:** The default memory limits should never be decreased as this will cause Black Duck to function incorrectly.

You can change the default memory limits for these containers:

- web app
- job runner
- scan
- binaryscanner

## Changing the default Web App container memory limits

There are three memory settings for the Web App container:

- The `HUB_MAX_MEMORY` environment variable controls the maximum Java heap size.
- The `limits memory` and `reservations memory` settings control the limit that Docker uses to schedule and limit the overall memory of the Web App container.
  - The `limits memory` setting is the amount of memory a container can use.
  - Docker uses the `reservations memory` setting to determine if a container can be deployed (scheduled) to a machine. Using this value, Docker ensures that all containers deployed to a machine have enough memory instead of all containers competing for the same memory.

Note that the value for each of these settings must be higher than the maximum Java heap size. If updating the Java heap size, Black Duck Software recommends setting the `limits memory` and `reservations memory` values to at least 1GB higher each than the maximum Java heap size.

The following example changes the maximum Java heap size for the Web App container to 8GB and the value for the `limit memory` and `reservations memory` settings to 9GB each.

In the `docker-swarm` directory, use this file:

- `docker-compose.yml` (if using the DB container)
- `docker-compose.bdba.yml` (if using the DB container with Black Duck - Binary Analysis)
- `docker-compose.externaldb.yml` (if using an external PostgreSQL instance),
- `docker-compose.externaldb.bdba.yml` (if using external PostgreSQL instance with Black Duck - Binary Analysis)

and edit these lines under the webapp services description:

**Original values:**

```
services:
```

```
...
webapp:
...
environment: {HUB_MAX_MEMORY: 4096m}
deploy:
  mode: replicated
  restart_policy: {condition: on-failure, delay: 5s, window: 60s}
  resources:
    limits: {cpus: '1', memory: 4608M}
    reservations: {cpus: '1', memory: 4608M}
...
```

### Updated values:

```
services:
...
webapp:
...
environment: {HUB_MAX_MEMORY: 8192m}
deploy:
  mode: replicated
  restart_policy: {condition: on-failure, delay: 5s, window: 60s}
  resources:
    limits: {cpus: '1', memory: 9216M}
    reservations: {cpus: '1', memory: 9216M}
...
```

## Changing the default Job Runner container memory limits

There are three memory settings for the Job Runner container:

- The `HUB_MAX_MEMORY` environment variable controls the maximum Java heap size.
- The `limits memory` and `reservations memory` settings control the limit that Docker uses to schedule and limit the overall memory of the Job Runner container.
  - The `limits memory` setting is the amount of memory a container can use.
  - Docker uses the `reservations memory` setting to determine if a container can be deployed (scheduled) to a machine. Using this value, Docker ensures that all containers deployed to a machine have enough memory instead of all containers competing for the same memory.

Note that the value for each of these settings must be higher than the maximum Java heap size. If updating the Java heap size, Black Duck Software recommends setting the `limits memory` and `reservations memory` values to at least 1GB higher each than the maximum Java heap size.

**Note:** These settings apply to all Job Runner containers, including scaled Job Runner containers.

The following example changes the maximum Java heap size for the Job Runner container to 8GB and the value for the `limit memory` and `reservations memory` settings to 9GB each. In the `docker-swarm`



directory, use this file:

- `docker-compose.yml` (if using the DB container)
- `docker-compose.bdba.yml` (if using the DB container with Black Duck - Binary Analysis )
- `docker-compose.externaldb.yml` (if using an external PostgreSQL instance),
- `docker-compose.externaldb.bdba.yml` (if using an external PostgreSQL instance with Black Duck - Binary Analysis)

and edit these lines under the jobrunner services description:

#### Original values:

```
services:
...
  jobrunner:
...
    environment: {HUB_MAX_MEMORY: 4096m}
    deploy:
      mode: replicated
      restart_policy: {condition: on-failure, delay: 5s, window: 60s}
      resources:
        limits: {cpus: '1', memory: 4608M}
        reservations: {cpus: '1', memory: 4608M}
        ...
```

#### Updated values:

```
services:
...
  jobrunner:
...
    environment: {HUB_MAX_MEMORY: 8192m}
    deploy:
      mode: replicated
      restart_policy: {condition: on-failure, delay: 5s, window: 60s}
      resources:
        limits: {cpus: '1', memory: 9216M}
        reservations: {cpus: '1', memory: 9216M}
        ...
```

### Changing the default Scan container memory limits

There are three memory settings for the Scan container:

- The `HUB_MAX_MEMORY` environment variable controls the maximum Java heap size.
- The `limits memory` and `reservations memory` settings control the limit that Docker uses to schedule and limit the overall memory of the Scan container.

- The `limits memory` setting is the amount of memory a container can use.
- Docker uses the `reservations memory` setting to determine if a container can be deployed (scheduled) to a machine. Using this value, Docker ensures that all containers deployed to a machine have enough memory instead of all containers competing for the same memory.

Note that the value for each of these settings must be higher than the maximum Java heap size. If updating the Java heap size, Black Duck Software recommends setting the `limits memory` and `reservations memory` values to at least 1GB higher each than the maximum Java heap size.

**Note:** These settings apply to all Scan containers, including scaled Scan containers.

The following example changes the maximum Java heap size for the scan container from 2GB to 4GB and the value for the `limit memory` and `reservations memory` settings to 5GB each. In the `docker-swarm` directory, use this file:

- `docker-compose.yml` (if using the DB container)
- `docker-compose.bdba.yml` (if using the DB container with Black Duck - Binary Analysis)
- `docker-compose.externaldb.yml` (if using an external PostgreSQL instance),
- `docker-compose.externaldb.bdba.yml` (if using an external PostgreSQL instance with Black Duck - Binary Analysis)

and edit these lines under the scan services description:

#### Original values:

```
services:
...
  scan:
...
  environment: {HUB_MAX_MEMORY: 2048m}
  deploy:
    mode: replicated
    restart_policy: {condition: on-failure, delay: 5s, window: 60s}
  resources:
    limits: {cpus: '1', memory: 2560M}
    reservations: {cpus: '1', memory: 2560M}
...

```

#### Updated values:

```
services:
...
  scan:
...
  environment: {HUB_MAX_MEMORY: 4096m}
  deploy:
    mode: replicated
    restart_policy: {condition: on-failure, delay: 5s, window: 60s}
  resources:

```

```
limits: {cpus: '1', memory: 5120M}
reservations: {cpus: '1', memory: 5120M}
...
```

## Changing the default binaryscanner container memory limits

The only default memory size for the binaryscanner container is the actual memory limit for the container.

**Note:** These settings apply to all binaryscanner containers, including scaled binaryscanner containers.

+The following configuration example will update the container memory limits from 2GB to 4GB. These configuration values can be changed

+in the 'docker-compose.yml' or whichever 'docker compose' file you are using under the 'binaryscanner' service section:

The following example changes the container memory limits from 2GB to 4GB. In the `docker-swarm` directory, use this file:

- `docker-compose.bdba.yml` (with Black Duck - Binary Analysis and using the DB container)
- `docker-compose.externaldb.bdba.yml` (with Black Duck - Binary Analysis and using an external PostgreSQL instance)

and edit these lines under the binaryscanner services section:

### Original values:

```
...
restart: always
mem_limit: 2048M
...
```

### Updated values:

```
...
restart: always
mem_limit: 4096M
...
```

## Configuring the report database password

This section provides instructions on configuring the report database password.

Use the `hub_reportdb_changepassword.sh` script, located in the `docker-swarm/bin` directory to set or change the report database password.

**Note:** This script sets or changes the report database password when using the database container that is automatically installed by Black Duck. If you are using an external PostgreSQL database, use your preferred PostgreSQL administration tool to configure the password.

Note that to run the script to set or change the password:

- You may need to be a user in the docker group, a root user, or have `sudo` access.
- You must be on the Docker host that is running the PostgreSQL database container.

In the following example, the report database password is set to 'blackduck':

```
./bin/hub_reportdb_changepassword.sh blackduck
```

## Accessing the API documentation through a proxy server

If you are using a reverse proxy and that reverse proxy has Black Duck under a subpath, configure the `BLACKDUCK_SWAGGER_PROXY_PREFIX` property so that you can access the API documentation. The value of `BLACKDUCK_SWAGGER_PROXY_PREFIX` is the Black Duck path. For example, if you have Black Duck being accessed under 'https://customer.companyname.com/hub' then the value of `BLACKDUCK_SWAGGER_PROXY_PREFIX` would be 'hub'.

To configure this property, edit the `hub-proxy.env` file located in the `docker-swarm` directory.

## Providing access to the REST APIs from a non-Black Duck server

You may wish to access Black Duck REST APIs from a web page that was served from a non-Black Duck server. To enable access to the REST APIs from a non-Black Duck server, Cross Origin Resource Sharing (CORS) must be enabled.

The properties used to enable and configure CORS for Black Duck installations are:

Property	Description
BLACKDUCK_HUB_CORS_ENABLED	Required. Defines whether CORS is enabled; "true" indicates CORS is enabled.
BLACKDUCK_CORS_ALLOWED_ORIGINS_PROP_NAME	<p>Required. Allowed origins for CORS.</p> <p>The browser sends an origin header when it makes a cross-origin request. This is the origin that must be listed in the <code>blackduck.hub.cors.allowedOrigins/BLACKDUCK_CORS_ALLOWED_ORIGINS_PROP_NAME</code> property.</p> <p>For example, if you are running a server that serves a page from <code>http:///123.34.5.67:8080</code>, then the browser should set this as the origin, and this value should be added to the property.</p> <p>Note that the protocol, host, and port must match. Use a comma-separated list to specify more than one base origin URL.</p>
BLACKDUCK_CORS_ALLOWED_HEADERS_PROP_NAME	Optional. Headers that can be used to make the requests.
BLACKDUCK_CORS_EXPOSED_HEADERS_PROP_NAME	Optional. Headers that can be accessed by the browser requesting CORS.

To configure these properties, edit the `hub-proxy.env` file, located in the `docker-swarm` directory.

## Configuring secure LDAP

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

To set up a trust connection to the secure LDAP server, import the server certificate into the local Black Duck LDAP truststore by:

1. Obtaining your LDAP information.
2. Using the Black Duck UI to import the server certificate.

### Obtaining your LDAP information

Contact your LDAP administrator and gather the following information:

#### LDAP Server Details

This is the information that Black Duck uses to connect to the directory server.

- (required) The host name or IP address of the directory server, including the protocol scheme and port, on which the instance is listening.

**Example:** `ldaps://<server_name>.<domain_name>.com:339`

- (optional) If your organization does not use anonymous authentication, and requires credentials for LDAP access, the password and either the LDAP name or the absolute LDAP distinguished name (DN) of a user that has permission to read the directory server.

**Example of an absolute LDAP DN:** `uid=ldapmanager,ou=employees,dc=company,dc=com`

**Example of an LDAP name:** `jdoe`

- (optional) If credentials are required for LDAP access, the authentication type to use: simple or digest-MD5.

### LDAP Users Attributes

This is the information that Black Duck uses to locate users in the directory server:

- (required) The absolute base DN under which users can be located.

**Example:** `dc=example,dc=com`

- (required) The attribute used to match a specific, unique user. The value of this attribute personalizes the user profile icon with the name of the user.

**Example:** `uid={0}`

### Test Username and Password

- (required) The user credentials to test the connection to the directory server.

## Importing the server certificate

### ⚙ To import the server certificate

1. Log in to Black Duck 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

Authentication Type

Manager DN

Manager Password

**LDAP User Attributes**

User Search Base

User Search Filter

User DN Pattern

**LDAP Attribute Mappings**

First Name

Last Name

Email

**LDAP Groups**

☐ Synchronize LDAP groups ☒ Synchronize LDAP groups

Group search base

Group filter

Group name attribute

**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

4. Select the **Enable LDAP** option and complete the information in the **LDAP Server Details** and **LDAP User Attributes** sections, as described above. In the **Server URL** field, ensure that you have configured the secure LDAP server: the protocol scheme is ldaps://.
5. Enter the user credentials in the **Test Connection, User Authentication and Field Mapping** section and click **Test Connection**.
6. If there are no issues with the certificate, it is automatically imported and the "Connection Test Succeeded" message appears:

**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  ✓ Connection Test Succeeded

✓ First Name First  
 ✓ Last Name Last  
 ✓ Email flast@company.com

7. If there is an issue with the certificate, a dialog box listing details about the certificate appears:

**Certificate Problem** [X]

Details about the certificates are below. If you'd like to accept this certificate, press "Save".

<b>Certificate Details</b>	
Issuer	CN=www.blackducksoftware.com, OU=Engineering, O="Black Duck Software, Inc.", L=Burlington, ST=Massachusetts, C=US
Subject	CN=www.blackducksoftware.com, OU=Engineering, O="Black Duck Software, Inc.", L=Burlington, ST=Massachusetts, C=US
Alt Subjects	blackducksoftware.com, ldap.blackducksoftware.com, sknb, *.updates.blackducksoftware.com
Begins On	Jun 19, 2017
Expires On	Jun 19, 2019
Algorithm	SHA1withRSA

Do one of the following:

- Click **Cancel** to fix the certificate issues.

Once fixed, retest the connection to verify that the certificate issues have been fixed and the certificate has been imported. If successful, the "Connection Test Succeeded" message appears.

- Click **Save** to import this certificate.

Verify that the certificate has been imported by clicking **Test Connection**. If successful, the "Connection Test Succeeded" message appears.

## LDAP trust store password

If you add a custom Black Duck web application trust store, use these methods for specifying an LDAP trust store password.

There are three methods for specifying an LDAP trust store password when using Docker Swarm.

- Use the docker secret command to tell Docker Swarm the password by using LDAP\_TRUST\_STORE\_PASSWORD\_FILE. The name of the secret must include the stack name. 'HUB' is the stack name in this example:

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

Add the password secret to the services section of the webapp service:



```
secrets:
  - LDAP_TRUST_STORE_PASSWORD_FILE
```

Add text such as the following to the end of the compose file:

```
secrets:
  LDAP_TRUST_STORE_PASSWORD_FILE:
    external:
      name: "HUB_LDAP_TRUST_STORE_PASSWORD_FILE"
```

- Mount a directory that contains a file called `LDAP_TRUST_STORE_PASSWORD_FILE` to `/run/secrets` by editing the volumes section for webapp services in the `docker-compose.yml` or `docker-compose.externaldb.yml` file located in the `docker-swarm` directory.

```
volumes: ['log-volume:/opt/blackduck/hub/logs', 'webapp-volume:/opt/blackduck/hub/hub-webapp/security', '/directory/where/files/are:/run/secrets']
```

- Specify an environment variable called `LDAP_TRUST_STORE_PASSWORD` that contains the password.

## Configuring SAML for Single Sign-On

Security Assertion Markup Language (SAML) is an XML-based, open-standard data format for exchanging authentication and authorization data between parties. For example, between an identity provider and a service provider. Black Duck's SAML implementation provides single sign-on (SSO) functionality, enabling Black Duck users to be automatically signed-in to Black Duck when SAML is enabled. Enabling SAML applies to all your Black Duck users, and cannot be selectively applied to individual users.

To enable or disable SAML functionality, you must be a user with the system administrator role.

For additional SAML information:

- Assertion Consumer Service (ACS): <https://host/saml/SSO>
- Recommended Service Provider Entity ID: **https://host** where *host* is your Black Duck server location.

Note the following:

- Black Duck is able to synchronize and obtain an external user's information (Name, FirstName, LastName and Email) if the information is provided in attribute statements. Note that the first and last name values are case-sensitive.

Black Duck is also able to synchronize an external user's group information if you enable group synchronization in Black Duck.

- When logging in with SAML enabled, you are re-directed to your identity provider's login page, not Black Duck's login page.
- When SSO users log out of Black Duck, a logout page now appears notifying them that they successfully logged out of Black Duck. This logout page includes a link to log back into Black Duck; users may not need to provide their credentials to successfully log back in to Black Duck.

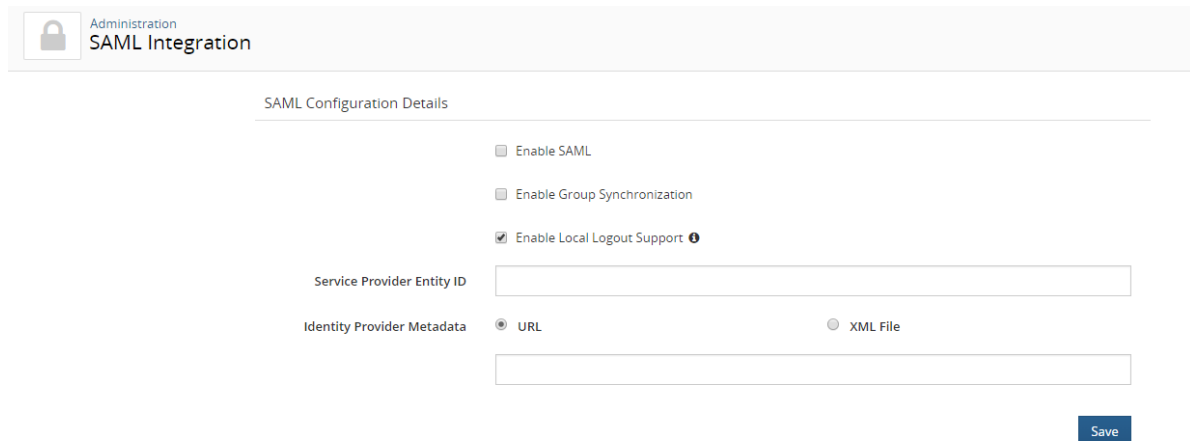
- If there are issues with the SSO system and you need to disable the SSO configuration, you can enter the following URL: *Black Duck servername/sso/login* to log in to Black Duck.

### ⚙️ To enable single sign-on using SAML

1. Click the expanding menu icon () and select **Administration**.

The Administration page appears.

2. Select **SAML Integration** to display the SAML Integration page.



3. In the **SAML Configuration Details** settings, complete the following:
  - a. Select the **Enable SAML** check box.
  - b. Optionally, select the **Enable Group Synchronization** check box. If this option is enabled, upon login, groups from the Identity Provider (IDP) are created in Black Duck and users will be assigned to those groups. Note that you must configure IDP to send groups in attribute statements with the attribute name of 'Groups'.
  - c. Optionally, select the **Enable Local Logout Support** check box. If this option is enabled, after logging out of Black Duck, the IDP's login page would appear.

**Note:** When local logout support is enabled, SAML requests are sent with ForceAuthn="true". Check with the IDP to confirm that this is supported.


- d. **Service Provider Entity ID** field. Enter the information for the Black Duck server in your environment in the format **https://host** where *host* is your Black Duck server.
- e. **Identity Provider Metadata**. Select one of the following:
  - **URL** and enter the URL for your identity provider.
  - **XML File** and either drop the file or click in the area shown to open a dialog box from which you can select the XML file.

4. Click **Save**.
5. Add the `HUB_SAML_EXTERNAL_URL` to your `hub-proxy.env` file (for Docker Swarm or Docker Compose) or the `3-hub.yml` file (for Kubernetes or OpenShift). The value is the public URL of the Black Duck server. For example:

```
HUB_SAML_EXTERNAL_URL=https://blackduck-docker01.dcl.lan
```

**Note:** You must restart Black Duck for your configuration changes to take effect.

#### ⚙ To disable single sign-on using SAML

1. Click the expanding menu icon () and select **Administration**.
2. Select **SAML Integration** to display the SAML Integration page.
3. In the **SAML Configuration Details** settings, clear the **Enable SAML** check box.
4. Click **Save**.

**Note:** You must restart Black Duck for your configuration changes to take effect.

## Providing your Black Duck system information to Customer Support

Customer Support may ask you to provide them with information regarding your Black Duck installation, such as system statistics and environmental or network information. To make it easier for you to quickly obtain this information, Black Duck provides a script, `system_check.sh`, which you can use to collect this information. The script outputs this information to a file, `system_check.txt`, located in your working directory, which you can then send to Customer Support.

The `system_check.sh` script is located in the `docker-swarm/bin` directory:

```
./bin/system_check.sh
```

Note that to run this script, you may need to be a user in the `docker` group, a root user, or have `sudo` access.

## Customizing user IDs of Black Duck containers

You may need to change the user ID (UID) under which a container runs.

The current UID for each container is:

- Authentication (hub-authentication): 100
- Binaryscanner (appcheck-worker): 0
- CA (hub-cfssl): 100
- DB (hub-postgres): 70
- Documentation (hub-documentation): 8080

- Job Runner (hub-jobrunner): 100
- Logstash (hub-logstash): 100
- RabbitMQ (rabbitmq): 100
- Registration (hub-registration): 8080
- Scan (hub-scan): 8080
- Solr (hub-solr): 8983
- Web App (hub-webapp): 8080
- webserver (hub-nginx): 100
- Uploadcache (hub-uploadcache): 100
- Zookeeper (hub-zookeeper): 1000

Changing the UID consists of editing the existing value for a container that is specified in the `.yaml` file. In the `docker-swarm` directory, use this file:

- `docker-compose.yaml` (if using the DB container)
- `docker-compose.bdba.yaml` (if using the DB container with Black Duck - Binary Analysis)
- `docker-compose.externaldb.yaml` (if using an external PostgreSQL instance),
- `docker-compose.externaldb.bdba.yaml` (if using an external PostgreSQL instance with Black Duck - Binary Analysis)

and edit the `user:` line in the container's section

The following example changes the UID for the Web App container from 8080 to 1001:

**Original value:**

```
services:
...
  webapp:
    ...
    user: tomcat:root
    ...
```

**Updated value:**

```
services:
...
  webapp:
    ...
    user: 1001:root
    ...
```

Note the following:

- The UID for the postgres container and binaryscanner *cannot* be changed.
  - The UID for the postgres container must equal 70.

- The UID for the binaryscanner container must equal 0 (root).
- If you change a value, you will need to edit these values again after you upgrade Black Duck.
- Although some containers have the same UID value (for example, the Documentation, Registration, Scan, and Web App container each has a UID of 8080), changing the UID value of one container does *not* change the UID value for the containers that have the same UID value. For example, changing the value of the Web App container from 8080 to 1001 does not change the value of the Documentation, Scan, or Registration containers – the UID value for these containers remains 8080.
- The containers expect that whichever user the container runs as, the user must still be specified as being in the root group.

#### ⚙ To customize the UID

1. [Bring down](#) Black Duck.
2. Edit the value as described above.
3. [Bring up](#) Black Duck.

## Including ignored components in reports

By default, ignored components and vulnerabilities associated with those ignored components are excluded from the Vulnerability Status report, Vulnerability Update report, Vulnerability Remediation report and the Project Version report. To include ignored components, set the value of the `BLACKDUCK_REPORT_IGNORED_COMPONENTS` environment variable in the `hub-proxy.env` file in the `docker-swarm` directory to "true".

Resetting the value of the `BLACKDUCK_REPORT_IGNORED_COMPONENTS` to "false" excludes ignored components.

## Enabling the hierarchical BOM

By default, the hierarchical BOM is disabled. To enable this feature, add the `HUB_HIERARCHICAL_BOM` environment variable to an `.env` file or to the webapp service in one of the following files located in the `docker-swarm` directory:

- `docker-compose.yml` (using the DB container)
- `docker-compose.bdba.yml` (using the DB container with Black Duck - Binary Analysis)
- `docker-compose.externaldb.yml` (using an external PostgreSQL instance)
- `docker-compose.externaldb.bdba.yml` (using an external PostgreSQL instance with Black Duck - Binary Analysis)

Set the value to "true", for example, `HUB_HIERARCHICAL_BOM=true`.

Resetting the value to "false" disables the feature.

## Increasing the size of the binary scan file

When using Black Duck - Binary Analysis, the maximum size of the binary that can be scanned is 6 GB. You can increase this limit by adding the environment variable `BINARY_UPLOAD_MAX_SIZE` to the `hub_webserver.env` file in the `docker-swarm` directory and specifying a value in megabytes.

For example, to increase the maximum binary scan to 7 GB, add the following:

```
BINARY_UPLOAD_MAX_SIZE=7168m
```

## Starting or stopping Black Duck

Use these commands to start up or shut down Black Duck.

### Starting up Black Duck

- Run the following command to start up Black Duck with the PostgreSQL database container:

```
docker swarm init
```

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

- Run the following command to start up Black Duck with Black Duck - Binary Analysis and using the PostgreSQL database container:

```
docker swarm init
```

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

- Run the following command to start up Black Duck with an external database:

```
docker swarm init
```

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

- Run the following command to start up Black Duck with Black Duck - Binary Analysis using an external database:

```
docker swarm init
```

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

### Shutting down Black Duck

- Run the following command to shut down Black Duck:

```
docker stack rm hub
```

## Chapter 5: Uninstalling Black Duck

Follow these instructions to uninstall Black Duck.

Use either of these methods to uninstall Black Duck:

- Stop and remove the containers and remove the volumes.

```
docker stack rm hub
```

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

```
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](#).

Black Duck 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 Black Duck:

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

### Installation files

The installation files are available on GitHub.

Download the orchestration files. As part of the install/upgrade process, these orchestration files pull down the necessary Docker images.

Note that although the filename of the `tar.gz` file differs depending on how you access the file, the content is the same.

#### Download from the GitHub page

1. Select the link to download the `.tar.gz` file from the GitHub page:  
<https://github.com/blackducksoftware/hub>.

2. Uncompress the Black Duck `.gz` file:

```
gunzip hub-5.0.0.tar.gz
```

3. Unpack the Black Duck `.tar` file:

```
tar xvf hub-5.0.0.tar
```

#### Download using the `wget` command

1. Run the following command:

```
wget https://github.com/blackducksoftware/hub/archive/v5.0.0.tar.gz
```

2. Uncompress the Black Duck `.gz` file:

```
gunzip v5.0.0.tar.gz
```



3. Unpack the Black Duck `.tar` file:

```
tar xvf v5.0.0.tar
```

## Upgrading from the AppMgr architecture

This section describes how to upgrade from a previous version of Black Duck based on the AppMgr architecture to the multi-container Docker architecture.

**Note:** These instructions also apply when upgrading from an AppMgr Amazon Web Services (AWS) AMI.

Upgrading to the multi-container Docker architecture consists of:

1. Migrating your PostgreSQL database.

This is an optional step if you want to retain your existing database data.

2. Upgrading Black Duck.

### 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 Black Duck server as the **blackduck** user.

**Note:** This is the user that owns the Black Duck 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

1. Use the `docker-compose.dbmigrate.yml` file located in the `docker-swarm` directory. 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 located in the `docker-swarm` directory. 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 Black Duck.

### Error messages

When the dump file is restored from the an AppMgr installation of Black Duck, 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 Black Duck

1. If you ran the `setup-autostart.sh` script in your previous AppMgr version of Black Duck, you will need to remove the 'iptables' entries that were created by that script. As a root user, `cd` to the directory where you installed Black Duck, for example, `/opt/blackduck/hub/appmgr/bin` and run the `iptables-redirect.sh` script with the `delete` parameter:

```
./iptables-redirect.sh delete
```

Note that you can safely run this script If you are unsure if autostart was configured as this script makes no changes if the previous AppMgr version of Black Duck was not configured for autostart.

2. If you are installing Black Duck on the same server that had the AppMgr version of Black Duck 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
```

3. Run one of the following commands, using the files located in the `docker-swarm` directory in the newer version of Black Duck:

- `docker stack deploy -c docker-compose.yml hub` (using the DB container)
- `docker stack deploy -c docker-compose.bdba.yml hub` (using the DB container with Black Duck - Binary Analysis)
- `docker stack deploy -c docker-compose.externaldb.yml hub` (using an external PostgreSQL database)
- `docker stack deploy -c docker-compose.externaldb.bdba.yml hub` (using an external PostgreSQL database with Black Duck - Binary Analysis)

## Upgrading from a single-container AppMgr architecture

This section describes how to upgrade from a previous version of Black Duck based on the single-container AppMgr architecture to the multi-container Docker architecture.

Upgrading to the multi-container Docker architecture consists of:

1. Migrating your PostgreSQL database.

This is an optional step if you want to retain your existing database data.

2. Upgrading Black Duck.

### 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

1. Use the `docker-compose.dbmigrate.yml` file located in the `docker-swarm` directory. 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 located in the `docker-swarm` directory. 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 Black Duck.

## Error messages

When the dump file is restored from the an AppMgr installation of Black Duck, 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 Black Duck

1. Run one of the following commands, using the files located in the `docker-compose` directory in the newer version of Black Duck:
  - `docker stack deploy -c docker-compose.yml hub` (using the DB container)
  - `docker stack deploy -c docker-compose.bdba.yml hub` (using the DB container with Black Duck - Binary Analysis)
  - `docker stack deploy -c docker-compose.externaldb.yml hub` (using an external PostgreSQL database)
  - `docker stack deploy -c docker-compose.externaldb.bdba.yml hub` (using an external PostgreSQL database with Black Duck - Binary Analysis)

## Upgrading from an existing Docker architecture

To upgrade from a previous version of Black Duck:

1. Migrate your PostgreSQL database.

The PostgreSQL database version was upgraded to version 9.6.x in 4.2.0. If you are upgrading from a version prior to 4.2.0, you must migrate your database prior to upgrading Black Duck.

The data migration will temporarily require an additional free disk space at approximately 2.5 times your original database volume size to hold the database dump and the new 4.2 database volume. As a rule-of-thumb, if the volume upon which your database resides is at least 60% free, there should be enough disk

space.

If you are upgrading from version 4.2.0, then migrating your database is optional.

## 2. Upgrade Black Duck.

**Note:** The method to configure custom SSL certificates for NGiNX changed in 4.1.0. If you are upgrading from version 4.0.0 or 4.0.1 and you had configured custom SSL certificates for NGiNX, you will need to [reconfigure them](#).

## 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. Bringing down Black Duck containers.
3. Restoring the data.

**Note:** If your Black Duck: instance was configured to use an external database (like Amazon RDS), the recommended approach is to migrate your data to a 9.6 instance of PostgreSQL and configure your system to point to that instance. If an administrator attempts to perform an upgrade on a system that is connected to a non-9.6 PostgreSQL database, the application will fail to start, however the data remains safe.

### ⚙ To back up the PostgreSQL database that is automatically installed with Black Duck

Run the following script which creates a PostgreSQL dump file in the hub-postgres container and then copies the dump file from the container to the local PostgreSQL dump file.

```
./bin/hub_create_data_dump.sh <path to local PostgreSQL dump file>
```

**Important:** You must run the `hub_create_data_dump.sh` script *before* upgrading Black Duck using the version of the script located in the pre-upgrade directory.

### ⚙ To bring down Black Duck containers

1. Run the following command to bring down Black Duck containers which removes the current stack that has the previous version of Black Duck running:

```
docker stack rm hub
```

### ⚙ To restore the PostgreSQL data

1. Use the `docker-compose.dbmigrate.yml` file located in the `docker-swarm` directory. 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 located in the `docker-swarm` directory. This script restores the data from the existing database dump file.

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

You can now upgrade Black Duck.

## Upgrading Black Duck

### ⚙ To upgrade Black Duck:

1. Run one of the following commands, using the files located in the `docker-swarm` directory in the newer version of Black Duck::
  - `docker stack deploy -c docker-compose.yml hub` (using the DB container)
  - `docker stack deploy -c docker-compose.bdba.yml hub` (using the DB container with Black Duck - Binary Analysis)
  - `docker stack deploy -c docker-compose.externaldb.yml hub` (using an external PostgreSQL instance)
  - `docker stack deploy -c docker-compose.externaldb.bdba.yml hub` (using an external PostgreSQL instance with Black Duck - Binary Analysis)

## Appendix A: Docker containers

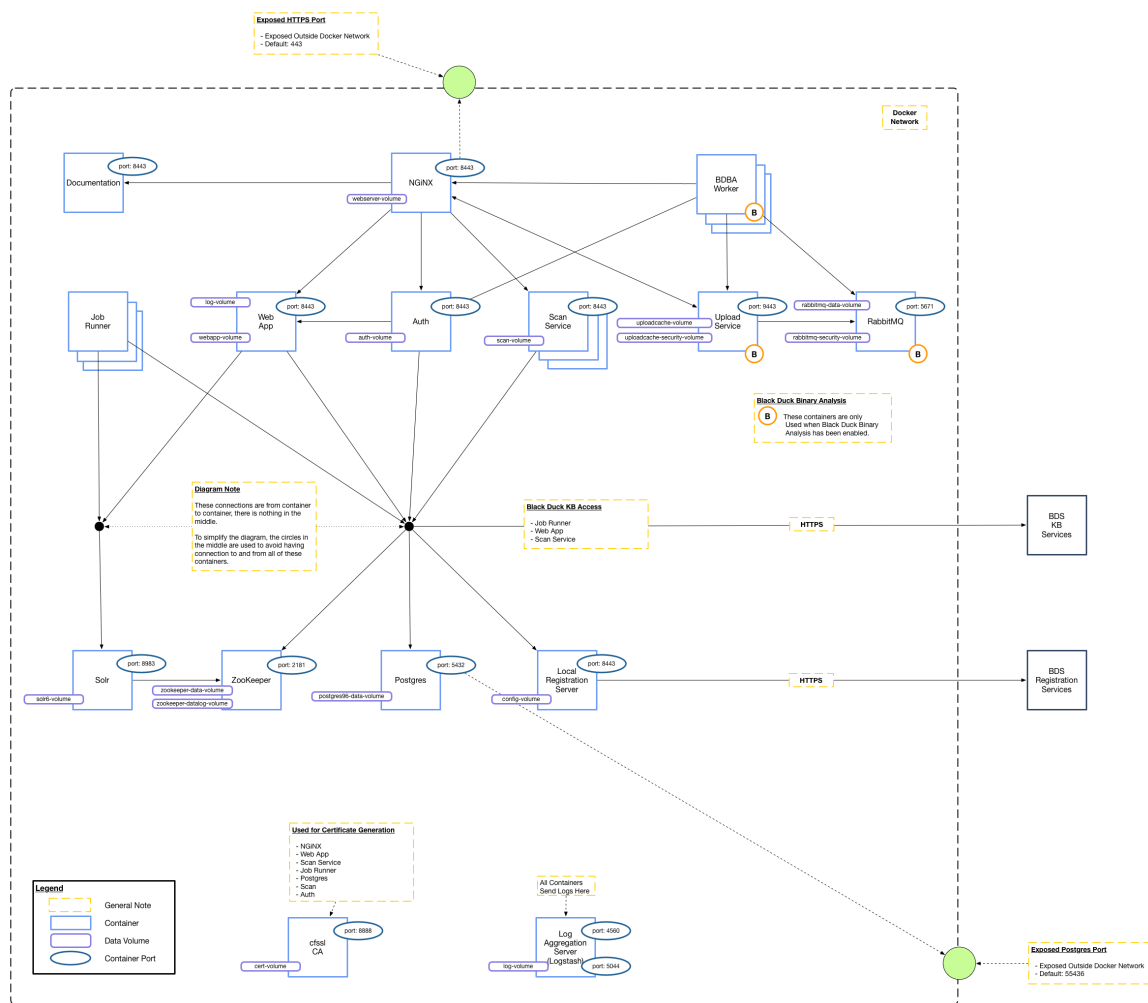
These are the containers within the Docker network that comprise the Black Duck application:

1. Webapp
2. Authentication
3. Scan
4. Job Runner
5. Solr
6. Registration
7. DB

**Note:** This container is not included in the Black Duck application if you use an external Postgres instance.

8. Documentation
9. WebServer
10. Zookeeper
11. LogStash
12. CA
13. RabbitMQ (only if Black Duck - Binary Analysis is enabled)
14. Uploadcache (only if Black Duck - Binary Analysis is enabled)
15. Binaryscanner (only if Black Duck - Binary Analysis is enabled)

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



The following tables provide more information on each container.

## Webapp container

Container Name: Webapp	
Image Name	blackducksoftware/hub-webapp:5.0.0
Description	The webapp 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 webapp 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.



Container Name: Webapp	
Links/Ports	<p>The webapp container needs to connect to these containers/services:</p> <ul style="list-style-type: none"> <li>• postgres</li> <li>• solr</li> <li>• zookeeper</li> <li>• registration</li> <li>• logstash</li> <li>• cfssl</li> </ul> <p>The container needs to expose port 8443 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"> <li>• postgres: \$HUB_POSTGRES_HOST</li> <li>• solr: This should be taken care of by ZooKeeper.</li> <li>• zookeeper: \$HUB_ZOOKEEPER_HOST</li> <li>• registration: \$HUB_REGISTRATION_HOST</li> <li>• logstash: \$HUB_LOGSTASH_HOST</li> <li>• cfssl: \$HUB_CFSSL_HOST</li> </ul>
Resources/Constraints	<ul style="list-style-type: none"> <li>• Default max Java heap size: 2GB</li> <li>• Container memory: 2.5GB</li> <li>• Container CPU: 1 CPU</li> </ul>
Users/Groups	<p>This container runs as UID 8080. If the container is started as UID 0 (root) then the user will be switched to UID 8080:root before executing its main process.</p> <p>This container is also able to be started as a random UID as long as it is also started within the root group (GID/fsGroup 0).</p>
Volumes	<p>log-volume:/opt/blackduck/hub/logs</p> <p>webapp-volume:/opt/blackduck/hub/hub-webapp/security</p>
Environment File	hub-proxy.env

## Authentication container

Container Name: hub-authentication	
Image Name	blackducksoftware/hub-authentication:5.0.0
Description	The authentication service is the container that all authentication-related requests are made against.
Scalability	There should only be a single instance of this container. It currently cannot be scaled.
Links/Ports	<p>Nothing external (8443 internally). This container will need to connect to these other containers/services:</p> <ul style="list-style-type: none"> <li>• postgres</li> <li>• cfssl</li> <li>• logstash</li> <li>• registration</li> <li>• zookeeper</li> <li>• webapp</li> </ul> <p>The container needs to expose 8443 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"> <li>• postgres - \$HUB_POSTGRES_HOST</li> <li>• cfssl - \$HUB_CFSSL_HOST</li> <li>• logstash - \$HUB_LOGSTASH_HOST</li> <li>• registration - \$HUB_REGISTRATION_HOST</li> <li>• zookeeper - \$HUB_ZOOKEEPER_HOST</li> <li>• webapp - \$HUB_WEBAPP_HOST</li> </ul>
Resources/Constraints	<ul style="list-style-type: none"> <li>• Default max Java heap size: 512MB</li> <li>• Container memory: 1GB</li> <li>• Container CPU: 1 CPU</li> </ul>
Users/Groups	<p>This container runs as UID 100. If the container is started as UID 0 (root) then the user will be switched to UID 100:root before executing its main process.</p> <p>This container is also able to be started as a random UID as long as it is also started within the root group (GID/fsGroup 0).</p>

Container Name: hub-authentication	
Volumes	authentication-volume: /opt/blackduck/hub/hub-authentication/security
Environment File	hub-proxy.env

## Scan container

Container Name: Scan	
Image Name	blackducksoftware/hub-scan:5.0.0
Description	The scan service is the container that all scan data requests are made against.
Scalability	This container can be scaled.
Links/Ports	<p>This container needs to connect to these containers/services:</p> <ul style="list-style-type: none"> <li>• postgres</li> <li>• zookeeper</li> <li>• registration</li> <li>• logstash</li> <li>• cfssl</li> </ul> <p>The container needs to expose port 8443 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"> <li>• postgres: \$HUB_POSTGRES_HOST</li> <li>• zookeeper: \$HUB_ZOOKEEPER_HOST</li> <li>• registration: \$HUB_REGISTRATION_HOST</li> <li>• logstash: \$HUB_LOGSTASH_HOST</li> <li>• cfssl: \$HUB_CFSSL_HOST</li> </ul>
Resources/Constraints	<ul style="list-style-type: none"> <li>• Default max Java heap size: 2GB</li> <li>• Container memory: 2.5GB</li> <li>• Container CPU: 1 CPU</li> </ul>
Users/Groups	<p>This container runs as UID 8080. If the container is started as UID 0 (root) then the user will be switched to UID 8080:root before executing its main process.</p> <p>This container is also able to be started as a random UID</p>

Container Name: Scan	
	as long as it is also started within the root group (GID/fsGroup 0).
Volumes	log-volume:/opt/blackduck/hub/logs scan-volume:/opt/blackduck/hub/hub-scan/security
Environment File	hub-proxy.env

## Job runner container

Container Name: Job Runner	
Image Name	blackducksoftware/hub-jobrunner:5.0.0
Description	The Job Runner container is the container that is responsible for running all of the application's 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"> <li>• postgres</li> <li>• solr</li> <li>• zookeeper</li> <li>• registration</li> <li>• logstash</li> <li>• cfssl</li> </ul>
Alternate Host Name Environment Variables	There are times when running in other types of orchestrations that any individual service name may be different. For example, you may have an external PostgreSQL endpoint which is resolved through a different service name. To support such use cases, these environment variables can be set to override the default host names: <ul style="list-style-type: none"> <li>• postgres: \$HUB_POSTGRES_HOST</li> <li>• solr: This should be taken care of by ZooKeeper.</li> <li>• zookeeper: \$HUB_ZOOKEEPER_HOST</li> <li>• registration: \$HUB_REGISTRATION_HOST</li> <li>• logstash: \$HUB_LOGSTASH_HOST</li> <li>• cfssl: \$HUB_CFSSL_HOST</li> </ul>

Container Name: Job Runner	
Resources/Constraints	<ul style="list-style-type: none"> <li>• Default max Java heap size: 4GB</li> <li>• Container memory: 4.5GB</li> <li>• Container CPU: 1 CPU</li> </ul>
Users/Groups	<p>This container runs as UID 100. If the container is started as UID 0 (root) then the user will be switched to UID 100:root before executing its main process.</p> <p>This container is also able to be started as a random UID as long as it is also started within the root group (GID/fsGroup 0).</p>
Volumes	N/A
Environment File	hub-proxy.env

## Solr container

Container Name: Solr	
Image Name	blackducksoftware/hub-solr:5.0.0
Description	<p>Solr is an open source enterprise search platform. Black Duck 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"> <li>• zookeeper</li> <li>• logstash</li> </ul> <p>The container needs to expose port 8983 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"> <li>• zookeeper: \$HUB_ZOOKEEPER_HOST</li> <li>• logstash: \$HUB_LOGSTASH_HOST</li> </ul>

Container Name: Solr	
Resources/Constraints	<ul style="list-style-type: none"> <li>• Default max Java heap size: 512MB</li> <li>• Container memory: 640MB</li> <li>• Container CPU: Unspecified</li> </ul>
Users/Groups	<p>This container runs as UID 8983. If the container is started as UID 0 (root) then the user will be switched to UID 8983:root before executing its main process.</p> <p>This container is also able to be started as a random UID as long as it is also started within the root group (GID/fsGroup 0).</p>
Volumes	solr6-volume:/opt/blackduck/hub/solr/cores.data
Environment File	N/A

## Registration container

Container Name: Registration	
Image Name	blackducksoftware/hub-registration:5.0.0
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.
Links/Ports	<p>The Registration container needs to connect to this containers/services:</p> <ul style="list-style-type: none"> <li>• logstash</li> <li>• cfssl</li> </ul> <p>The container needs to expose port 8443 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"> <li>• logstash: \$HUB_LOGSTASH_HOST</li> <li>• cfssl: \$HUB_CFSSL_HOST</li> </ul>
Resources/Constraints	<ul style="list-style-type: none"> <li>• Default max Java heap size: 512MB</li> <li>• Container memory: 640MB</li> <li>• Container CPU: Unspecified</li> </ul>

Container Name: Registration	
Users/Groups	This container runs as UID 8080. If the container is started as UID 0 (root) then the user will be switched to UID 8080:root before executing its main process. This container is also able to be started as a random UID as long as it is also started within the root group (GID/fsGroup 0).
Volumes	config-volume:/opt/blackduck/hub/hub-registration/config
Environment File	hub-proxy.env

## DB container

**Note:** This container is not included in the Black Duck application if you use an external Postgres instance.

Container Name: DB	
Image Name	blackducksoftware/hub-postgres:5.0.0
Description	<p>The DB container holds the PostgreSQL database which is an open source object-relational database system. The application uses the PostgreSQL database to store data.</p> <p>There is a single instance of this container. This is where all of the application's data is stored. There are two sets of ports for Postgres. One port will be exposed to containers within the Docker network. This is the connection that the application will 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"> <li>• logstash</li> <li>• cfssl</li> </ul> <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"> <li>• logstash: \$HUB_LOGSTASH_HOST</li> <li>• cfssl: \$HUB_CFSSL_HOST</li> </ul>
Resource/Constraints	<ul style="list-style-type: none"> <li>• Default max Java heap size: N/A</li> <li>• Container memory: 3GB</li> <li>• Container CPU: 1 CPU</li> </ul>
Users/Groups	<p>This container runs as UID 70. If the container is started as UID 0 (root) then the user will be switched to UID 70:root before executing its main process.</p> <p>This container is not able to start with any other user id.</p>
Volumes	postgres96-data-volume:/var/lib/postgresql/data
Environment File	N/A

## Documentation container

Container Name: Documentation	
Image Name	blackducksoftware/hub-documentation:5.0.0
Description	The Documentation container supplies documentation for the application.
Scalability	There is a single instance of this container. It should not be scaled.



Container Name: Documentation	
Links/Ports	<p>This container must connect to these other containers/services:</p> <ul style="list-style-type: none"> <li>• logstash</li> <li>• cfssl</li> </ul> <p>The documentation container must expose port 8443 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"> <li>• logstash: \$HUB_LOGSTASH_HOST</li> <li>• cfssl: \$HUB_CFSSL_HOST</li> </ul>
Resources/Constraints	<ul style="list-style-type: none"> <li>■ Default Max Java Heap Size: 512MB</li> <li>■ Container Memory: 512MB</li> <li>■ Container CPU: unspecified</li> </ul>
Users/Groups	<p>This container runs as UID 8080. If the container is started as UID 0 (root) then the user will be switched to UID 8080:root before executing its main process.</p> <p>This container is also able to be started as a random UID as long as it is also started within the root group (GID/fsGroup 0).</p>
Volumes	N/A
Environment File	N/A

## WebServer container

Container Name: WebServer	
Image Name	blackducksoftware/hub-nginx:5.0.0
Description	<p>The WebServer container is a reverse proxy for containers with the application. It has a port exposed outside of the Docker network. This is the container configured for HTTPS. There are config volumes here to allow for the configuration of HTTPS.</p>
Scalability	The container should not be scaled.

Container Name: WebServer	
Links/Ports	<p>The Web App container needs to connect to these containers/services:</p> <ul style="list-style-type: none"> <li>• webapp</li> <li>• cfssl</li> <li>• documentation</li> <li>• scan</li> <li>• authentication</li> <li>• upload cache (if Black Duck - Binary Analysis is enabled)</li> </ul> <p>This container exposes port 443 outside of the Docker network.</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"> <li>• webapp: \$HUB_WEBAPP_HOST</li> <li>• cfssl: \$HUB_CFSSL_HOST</li> <li>• scan: \$HUB_SCAN_HOST</li> <li>• documentation: \$HUB_DOC_HOST</li> <li>• authentication: \$HUB_AUTHENTICATION_HOST</li> <li>• upload cache: \$HUB_UPLOAD_CACHE_HOST</li> </ul>
Resources/Constraints	<ul style="list-style-type: none"> <li>• Default max Java heap size: N/A</li> <li>• Container memory: 512MB</li> <li>• Container CPU: Unspecified</li> </ul>
Users/Groups	<p>This container runs as UID 100. If the container is started as UID 0 (root) then the user will be switched to UID 100:root before executing its main process.</p> <p>This container is also able to be started as a random UID as long as it is also started within the root group (GID/fsGroup 0).</p>
Volumes	webserver-volume:/opt/blackduck/hub/webserver/security
Environment File	hub-webserver.env, hub-bdba.env

## ZooKeeper container

Container Name: Zookeeper	
Image Name	blackducksoftware/hub-zookeeper:5.0.0
Description	This container stores data for the other containers. It exposes ports within the Docker network, but not outside the Docker network.
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"> <li>• logstash</li> </ul> <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"> <li>• logstash: \$HUB_LOGSTASH_HOST</li> </ul>
Constraints	<ul style="list-style-type: none"> <li>• Default max Java heap size: 256MB</li> <li>• Container memory: 384MB</li> <li>• Container CPU: Unspecified</li> </ul>
Users/Groups	<p>This container runs as UID 1000. If the container is started as UID 0 (root) then the user will be switched to UID 1000:root before executing its main process.</p> <p>This container is also able to be started as a random UID as long as it is also started within the root group (GID/fsGroup 0).</p>
Volumes	<p>zookeeper-data-volume:/opt/blackduck/zookeeper/data</p> <p>zookeeper-datalog-volume:/opt/blackduck/zookeeper/datalog</p>
Environment File	N/A

## Logstash container

Container Name: LogStash	
Image Name	blackducksoftware/hub-logstash:5.0.0
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.
Resources/Constraints	<ul style="list-style-type: none"> <li>• Default max Java heap size: 1GB</li> <li>• Container memory: 1GB</li> <li>• Container CPU: Unspecified</li> </ul>
Users/Groups	<p>This container runs as UID 100. If the container is started as UID 0 (root) then the user will be switched to UID 100:root before executing its main process.</p> <p>This container is also able to be started as a random UID as long as it is also started within the root group (GID/fsGroup 0).</p>
Volumes	log-volume:/var/lib/logstash/data
Environment File	N/A

## CA container

Container Name: CA	
Image Name	blackducksoftware/hub-cfssl:5.0.0
Description	This container uses CFSSL which is used for certificate generation for PostgreSQL, NGiNX, and clients that need to authenticate to Postgres. This container is also used to generate TLS certificates for the internal containers that make up the application.
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.
Resources/Constraints	<ul style="list-style-type: none"> <li>• Default max Java heap size: N/A</li> <li>• Container memory: 512MB</li> <li>• Container CPU: Unspecified</li> </ul>

Container Name: CA	
Users/Groups	<p>This container runs as UID 100. If the container is started as UID 0 (root) then the user will be switched to UID 100:root before executing its main process.</p> <p>This container is also able to be started as a random UID as long as it is also started within the root group (GID/fsGroup 0).</p>
Volumes	cert-volume:/etc/cfssl
Environment File	N/A

## Black Duck - Binary Analysis containers

The following containers will only be installed if you have Black Duck - Binary Analysis

### Rabbitmq container

Container Name: rabbitmq	
Image Name	blackducksoftware/rabbitmq:1.0.0
Description	<p>This container facilitates upload information to the binary analysis worker. It exposes ports within the Docker network, but not outside the Docker network.</p> <p>This container is currently only used if Black Duck - Binary Analysis is enabled.</p>
Scalability	There should only be a single instance of this container. It should not be scaled.
Links/Ports	<p>This container needs to connect to these other containers/services:</p> <ul style="list-style-type: none"> <li>cfssl</li> </ul> <p>The container needs to expose port 5671 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"> <li>cfssl: \$HUB_CFSSL_HOST</li> </ul>
Constraints	<ul style="list-style-type: none"> <li>Default max Java heap size: N/A</li> <li>Container memory: 1GB</li> <li>Container CPU: Unspecified</li> </ul>
Users/Groups	This container runs as UID 100. If the container is started

Container Name: rabbitmq	
	<p>as UID 0 (root) then the user will be switched to UID 100:root before executing it's main process.</p> <p>This container is also able to be started as a random UID as long as it is also started within the root group (GID/fsGroup 0).</p>
Volumes	<p>rabbitmq-security-volume:/opt/blackduck/rabbitmq/security</p> <p>rabbitmq-data-volume:/var/lib/rabbitmq</p>
Environment File	hub-bdba.env

## Uploadcache container

Container Name: Uploadcache	
Image Name	blackducksoftware/hub-uploadcache:1.0.0
Description	<p>This container will be used to temporarily store uploads for binary analysis. It exposes ports within the Docker network, but not outside the Docker network.</p> <p>This container is currently only used if Black Duck - Binary Analysis is enabled.</p>
Scalability	There should only be a single instance of this container. It should not be scaled.
Links/Ports	<p>This container needs to connect to these containers/services:</p> <ul style="list-style-type: none"> <li>• cfssl</li> <li>• rabbitmq</li> <li>• logstash</li> </ul> <p>The container exposes ports 9443 and 9444 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"> <li>• cfssl: \$HUB_CFSSL_HOST</li> <li>• logstash: \$HUB_LOGSTASH_HOST</li> <li>• rabbitmq: \$RABBIT_MQ_HOST</li> </ul>

Container Name: Uploadcache	
Constraints	<ul style="list-style-type: none"> <li>• Default max Java heap size: N/A</li> <li>• Container memory: 512MB</li> <li>• Container CPU: Unspecified</li> </ul>
Users/Groups	This container runs as UID 100. If the container is started as UID 0 (root) then the user will be switched to UID 100:root before executing its main process. This container is also able to be started as a random UID as long as it is also started within the root group (GID/fsGroup 0).
Volumes	<p>uploadcache-security-volume/opt/blackduck/hub/hub-upload-cache/security</p> <p>uploadcache-volume:/opt/blackduck/hub/hub-upload-cache/uploads</p>
Environment File	hub-bdba.env

## Binaryscanner container

Container Name: binaryscanner	
Image Name	defensics-store.internal.synopsys.com:5004/appcheck-worker:latest
Description	<p>This container analyzes binary files.</p> <p>This container is currently only used if Black Duck - Binary Analysis is enabled.</p>
Scalability	This container can be scaled.
Links/Ports	<p>This container needs to connect to these containers/services:</p> <ul style="list-style-type: none"> <li>• cfssl</li> <li>• logstash</li> <li>• rabbitmq</li> <li>• webserver</li> </ul> <p>The container will need to expose port 5671 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"> <li>• cfssl: \$HUB_CFSSL_HOST</li> <li>• logstash: \$HUB_LOGSTASH_HOST</li> </ul>

Container Name: binaryscanner	
	<ul style="list-style-type: none"><li>• rabbitmq: \$RABBIT_MQ_HOST</li><li>• webserver: \$HUB_WEBSERVER_HOST</li></ul>
Resources/Constraints	<ul style="list-style-type: none"><li>• Default max Java heap size: N/A</li><li>• Container memory: 2GB</li><li>• Container CPU: 1 CPU</li></ul>
Users/Groups	This container runs as UID 0.
Volumes	N/A
Environment File	hub-bdba.env