



Installing Black Duck using Docker Swarm

Version 2021.4.0



This edition of the *Installing Black Duck using Docker Swarm* refers to version 2021.4.0 of Black Duck.

This document created or updated on Friday, April 23, 2021.

Please send your comments and suggestions to:

Synopsys
800 District Avenue, Suite 201
Burlington, MA 01803-5061 USA

Copyright © 2021 by Synopsys.

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

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

Chapter 1: Overview	1
Black Duck Architecture	1
Components hosted on Black Duck servers	1
Chapter 2: Installation planning	2
Getting started	2
New installations	2
Upgrading from a previous version of Black Duck	2
Hardware requirements	2
Docker requirements	3
Operating systems	4
Software requirements	4
Network requirements	4
Database requirements	5
PostgreSQL versions	6
Proxy server requirements	6
Configuring your NGiNX server to work with Black Duck	6
Amazon services	8
Additional port information	8
Configuring the keepalive setting	9
KnowledgeBase Feedback Service	9
User-agent analytics	10
Disabling the feedback service	10
Chapter 3: Installing Black Duck	11
Installation files	12
Download from the GitHub page	12
Download using the wget command	12
Distribution	12
Installing Black Duck	13
Rapid Scanning in Black Duck	15
Chapter 4: Administrative tasks	16
Using environment files	17
Environment variables and scanning binaries	17
Duplicate BOM Detection	18

Changing the expiration time for a bearer token	18
About the KBMATCH_SENDFPATH parameter	18
Accessing the API documentation through a proxy server	18
Providing access to the REST APIs from a non-Black Duck server	19
Increasing the size of the binary scan file	19
Configuring the Dashboard refresh rate	20
Managing certificates	20
Using custom certificates	21
Getting component migration data from the KnowledgeBase	23
Enabling the recording of migrations	23
Retention of migration data	23
API endpoints	23
Enabling the hierarchical BOM	23
Including ignored components in reports	24
Configuring secure LDAP	24
Obtaining your LDAP information	24
Importing the server certificate	25
LDAP trust store password	27
Accessing log files	28
Obtaining logs	29
Viewing log files for a container	29
Purging logs	29
Changing the default memory limits	30
Changing the default webapp container memory limits	30
Changing the default jobrunner container memory limits	31
Changing the default scan container memory limits	31
Changing the default binaryscanner container memory limits	32
Changing the default bomengine container memory limits	32
Changing hostname for logstash	33
Cleaning up unmapped code locations	34
Schedule a scan purge job by using a cron string	34
Clearing stuck BOM events	35
Using the override file	35
Configuring analytics in Black Duck	35
Configuring an external PostgreSQL instance	36
Modifying the PostgreSQL usernames for an existing external database	40
Configuring proxy settings	41
Proxy password	41
Importing a proxy certificate	42
Configuring the report database password	43
Scaling job runner, scan, bomengine, and binaryscanner containers	43

Scaling bomengine containers	43
Scaling job runner containers	44
Scaling scan containers	44
Scaling binaryscanner containers	44
Configuring SAML for Single Sign-On	44
Uploading source files	47
Backing up the seal key and raw master key	49
Replacing the seal key	49
Additional configuration options	49
Starting or stopping Black Duck	50
Starting up Black Duck	50
Starting up Black Duck when using the override file	51
Shutting down Black Duck	51
Configuring user session timeout	52
Providing your Black Duck system information to Customer Support	53
Understanding the default sysadmin user	54
Configuring Black Duck reporting delay	54
Configuring the containers' time zone	54
Modifying the default usage	54
Customizing user IDs of Black Duck containers	56
Configuring Web server settings	57
Configuring the hostname	57
Configuring the host port	58
Disabling IPv6	58
Chapter 5: Uninstalling Black Duck	59
Chapter 6: Upgrading Black Duck	60
Installation files	60
Download from the GitHub page	61
Download using the wget command	61
Migration script to purge unused rows in the audit event table	61
Upgrading from the AppMgr architecture	62
Migrating your PostgreSQL database	63
Upgrading Black Duck	64
Upgrading from a single-container AppMgr architecture	65
Migrating your PostgreSQL database	65
Upgrading Black Duck	66
Upgrading from an existing Docker architecture	67
Migrating your PostgreSQL databases	67
Upgrading Black Duck	69
Appendix A: Docker containers	71
Authentication container	72

CA container	74
DB container	74
Documentation container	76
Jobrunner container	77
Logstash container	78
Registration container	78
Scan container	79
Uploadcache container	80
Webapp container	81
Webserver container	82
Redis container	83
Rabbitmq container	84
bomengine container	85
Black Duck - Binary Analysis container	86
Binaryscanner container	86

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 Swarm	install_swarm.pdf	Contains information about installing and upgrading Black Duck using Docker Swarm.
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.
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.

The installation methods for installing Black Duck software in a Kubernetes or OpenShift environment are Synopsysctl and Helm. Click the following links to view the documentation.

- [Helm](#) is a package manager for Kubernetes that you can use to install Black Duck.
- [Synopsysctl](#) is a cloud-native administration command-line tool for deploying Black Duck software in Kubernetes and Red Hat [OpenShift](#).

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

Customer support

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

You can contact Synopsys Support in several ways:

- Online: <https://www.synopsys.com/software-integrity/support.html>
- Phone: See the Contact Us section at the bottom of our [support page](#) to find your local phone number.

To open a support case, please log in to the Synopsys Software Integrity Community site at <https://community.synopsys.com/s/contactsupport>.

Another convenient resource available at all times is the [online customer portal](#).

Synopsys Software Integrity Community

The Synopsys Software Integrity Community is our primary online resource for customer support, solutions, and information. The 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 Software Integrity Group (SIG) customers. The many features included in the 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 SIG 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 Synopsys 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 SIG experts and our Knowledgebase.
- Share – Collaborate and connect with Software Integrity Group 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, click [here](#) to get started, or send an email to community.manager@synopsys.com.

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://community.synopsys.com/s/education>.

Synopsys Statement on Inclusivity and Diversity

Synopsys is committed to creating an inclusive environment where every employee, customer, and partner feels welcomed. We are reviewing and removing exclusionary language from our products and supporting customer-facing collateral. Our effort also includes internal initiatives to remove biased language from our engineering and working environment, including terms that are embedded in our software and IPs. At the same time, we are working to ensure that our web content and software applications are usable to people of varying abilities. You may still find examples of non-inclusive language in our software or documentation as our IPs implement industry-standard specifications that are currently under review to remove exclusionary language.

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 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.

Chapter 2: Installation planning

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:

- 6 CPUs
- 26 GB RAM for the minimum Redis configuration; 29 GB for an optimal configuration providing higher availability for Redis-driven caching.
- 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:

- 7 CPUs
- 30 GB RAM for the minimum Redis configuration; 33 GB for an optimal configuration providing higher availability for Redis-driven caching.
- 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 these restrictions when using Black Duck in Docker Swarm:

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

This does *not* apply to installations using an external PostgreSQL instance.

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

This is required so that the blackduck-webapp service can access the logs that need to be downloaded.

- The blackduck-registration service must always run on the same node in the cluster or be backed by an NFS volume or a similar system, so that registration data is not lost.

It does not need to be the same node as used for the blackduck-database service or the blackduck-webapp service.

- The blackduck-upload-cache service must always run on the same node in the cluster or be backed by

an NFS volume or a similar system, so that data is not lost.

It does not need to be the same node as used by other services.

Docker Version

Black Duck installation supports Docker versions 18.09.x, 19.03.x, and 20.10.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 18.04.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 the supported Docker versions.

Note: Docker CE does not support Red Hat Enterprise Linux, Oracle Linux, or SUSE Linux Enterprise Server (SLES). 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 Version 90.0.4430.72 (Official Build) (x86_64)
- Firefox Version 88.0 (64-bit)
- Microsoft Edge Version 90.0.818.41 (Official build) (64-bit)
- Safari Version 14.0.3 (15610.4.3.1.7, 15610)

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
- Port 55456 - Access to restart Black Duck Hub

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)
- <https://auth.docker.io/token?scope=repository/blackducksoftware/blackduckregistration/pull&service=registry.docker.io> (access to Docker Registry)

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

Ensure that the following addresses are on the allow list:

- kb.blackducksoftware.com
- updates.suite.blackducksoftware.com
- hub.docker.com
- registry-1.docker.io
- auth.docker.io
- github.com
- docker.io

To verify connectivity, use the cURL command as shown in the following example.

```
curl -v https://kb.blackducksoftware.com
```

Tip: It's good to check connectivity on the Docker host but it's better to verify the connectivity from within your Docker network.

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 PostgreSQL instance.

Important: As of Black Duck 2020.6.0, Synopsys recommends PostgreSQL 11.7 for new installs that use external PostgreSQL. PostgreSQL 9.6 continues to be fully supported for external PostgreSQL instances.

Important: For users of the internal PostgreSQL container, PostgreSQL 9.6 remains as the supported version for Black Duck 2020.6.0.

For an external PostgreSQL instance, Black Duck supports:

- PostgreSQL 11.x via Amazon Relational Database Service (RDS)
- PostgreSQL 11.x via Google Cloud SQL
- PostgreSQL 11.x (Community Edition)

Note: PostgreSQL 10.x or PostgreSQL 12.x is not supported.

Refer to [Configuring an external PostgreSQL instance](#) for more information.

PostgreSQL versions

For Black Duck 2021.4.0, the currently-supported version of PostgreSQL for the internal PostgreSQL container is 9.6.x, which is the version supplied in Black Duck's PostgreSQL container.

If you choose to run your own external PostgreSQL instance, Synopsys recommends PostgreSQL 11.7 for new installs.

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. Example: X-Forwarded-Host: "10.20.30.40,my.example, 10.1.20.20"
X-Forwarded-Port	Contains a single value representing the port used for the original request. Example: X-Forwarded-Port: "9876"

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

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.

Important: Synopsys recommends that you use PostgreSQL version 11.7 (external PostgreSQL database).
External PostgreSQL instances now support user names that consist of only numeric characters.

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

Configuring the keepalive setting

The `net.ipv4.tcp_keepalive_time` parameter controls how long an application will let an open TCP connection remain idle. By default, this value is 7200 seconds (2 hours).

For optimal Black Duck performance, this parameter should have a value between 600 and 800 seconds.

This setting can be configured before or after Black Duck is installed.

To edit the value

1. Edit the `/etc/sysctl.conf` file. For example:

```
vi /etc/sysctl.conf
```

You can also use the `sysctl` command to modify this file.

2. Add the `net.ipv4.tcp_keepalive_time` (if the parameter is not in the file) or edit the existing value (if the parameter is in the file).

```
net.ipv4.tcp_keepalive_time = <value>
```

3. Save and exit the file.
4. Enter the following command to load the new setting:

```
sysctl -p
```

5. If Black Duck is installed, restart it.

KnowledgeBase Feedback Service

The KnowledgeBase feedback is used to enhance Black Duck KnowledgeBase (KB) capabilities.

- Feedback is sent when you make BOM adjustments to the component, version, origin, origin ID, or license of a match made by the KB.
- Feedback is also sent if you identify unmatched files to a component; it is not sent for manually added components that do not have files associated with them.
- Feedback is used to improve the accuracy of future matches. This information also helps Synopsys to

prioritize resources so that components which are important to our customers can be examined in more detail.

Important: No customer-identifiable information is transmitted to the KB.

User-agent analytics

The KnowledgeBase uses originating user-agent analytics to improve the scalability of KnowledgeBase services and improve quality of service for users.

The additional header information increases the header size of outgoing HTTP requests to the KnowledgeBase. It is possible that some intermediate egress proxies (customer managed) may require reconfiguration to support the additional header size, but this scenario is unlikely.

Disabling the feedback service

By default, the KnowledgeBase feedback service is enabled: adjustments that you make to a BOM are sent to the KnowledgeBase.

- You can override the feedback service by using the `BLACKDUCK_KBFEEDBACK_ENABLED` environment variable. A value of `false` overrides the feedback service and BOM adjustments are not sent to the KnowledgeBase.
- To disable the feedback service, add `BLACKDUCK_KBFEEDBACK_ENABLED=false` to the `blackduck-config.env` file.

Note: Set the value to `true` to re-enable the feedback service.

Chapter 3: Installing Black Duck

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

Black Duck Installation Requirements	
Hardware requirements	
<input type="checkbox"/>	You have ensured that your hardware meets the minimum hardware requirements .
Docker requirements	
<input type="checkbox"/>	You have ensured that your system meets the docker requirements .
Software requirements	
<input type="checkbox"/>	You have ensured that your system and potential clients meet the software requirements .
Network requirements	
<input type="checkbox"/>	<p>You have ensured that your network meets the network requirements. Specifically:</p> <ul style="list-style-type: none">• Port 443 and port 55436 are externally accessible.• The server has access to updates.suite.blackducksoftware.com which is used to validate the Black Duck license.
Database requirements	
<input type="checkbox"/>	<p>You have selected your database configuration.</p> <p>Specifically, you have configured database settings if you are using an external PostgreSQL instance.</p>
Proxy requirements	
<input type="checkbox"/>	<p>You have ensured that your network meets the proxy requirements.</p> <p>Configure proxy settings before or after installing Black Duck.</p>
Web server requirements	
<input type="checkbox"/>	Configure web server settings 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-2021.4.0.tar.gz
```

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

```
tar xvf hub-2021.4.0.tar
```

Download using the wget command

1. Run the following command:

```
wget https://github.com/blackducksoftware/hub/archive/v2021.4.0.tar.gz
```

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

```
gunzip v2021.4.0.tar.gz
```

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

```
tar xvf v2021.4.0.tar
```

Distribution

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

- `blackduck-config.env`: Environment file to configure Black Duck settings.
- `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.yml`: Docker Compose file used with an external PostgreSQL database.
- `docker-compose.local-overrides.yml`: Docker Compose file used to override any default settings in the `.yml` file.
- `docker-compose.readonly.yml`: Docker Compose file that declares the file system as read-only

for Swarm services.

- `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-webserver.env`: Environment file to [configure web server settings](#).

In the `bin` directory:

- `bd_get_source_upload_master_key.sh`: Script used to back up the master and seal key when [uploading source files](#).
- `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](#).
- `recover_master_key.sh`: Script to create a new seal key used for [uploading source files](#).
- `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. Use the `.yaml` file(s) located in the `docker-swarm` directory.

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

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

In the following instructions to install Black Duck, you may need to be a user in the `docker` group, a root user, or have `sudo` access. See the next section to install Black Duck as a non-root user.

- To install Black Duck with the PostgreSQL database container:

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

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

- To install Black Duck with a file system as read-only for Swarm services, add the `docker-compose.readonly.yml` file to the previous instructions.

For example, to install Black Duck with the PostgreSQL database container, enter the follow command:

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

Note: There are some versions of Docker where if the images live in a private repository, `docker stack` will not pull them unless the following flag is added to the above commands: `--with-registry-auth`.

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.

Rapid Scanning in Black Duck

Black Duck Rapid scanning is introduced in Black Duck 2021.4.0 with limited customer availability (LCA).

Rapid scanning (dependency scan) provides developers with an efficient method to check for vulnerabilities or policy violations in their code prior to merging, without having to create a BOM in Black Duck.

Important: The Rapid scanning feature is limited availability for the 2021.4.0 release.

If you want to try Rapid scanning, contact your account management team who will connect you with product management to set you up with the necessary files and instructions to start using the Rapid scanning feature.

Chapter 4: Administrative tasks

This chapter describes these administrative tasks:

A

- [About the KBMATCH_SENDFPATH parameter.](#)
- [API documentation, providing access to the API documentation through a proxy server.](#)
- [APIs, providing access to the REST APIs from a non-Black Duck server.](#)

B

- [Binary scan file, increasing the size](#)

C

- [Certificate, replacing the existing self-signed certificate](#)
- [Changing hostname for logstash](#)
- [Cleaning up unmapped code locations](#)
- [Changing the expiration time for a bearer token](#)
- [Clearing stuck BOM events](#)

D

- [Detecting duplicate BOMs](#)

H

- [Hierarchical BOM, enabling](#)

I

- [Ignored components, including ignored components in reports](#)

L

- [LDAP, configuring secure LDAP.](#)
- [Log files, accessing](#)

M

- [Memory limits, changing the default memory limits](#)
- [Migration data retrieval from the KnowledgeBase](#)

O

- [Override file, using](#)

P

- [PostgreSQL, configuring an external PostgreSQL instance.](#)
- [Proxy settings, configuring](#)

R

- [Report database, configuring the report database password.](#)

S

- [Scaling job runner, scan and binaryscanner containers.](#)
- [Single Sign-on \(SSO\), configuring](#)
- [Source files, uploading](#)
- [Starting or stopping Black Duck](#)
- [Support, providing your Black Duck system information to Support.](#)
- [Sysadmin user, understanding the default sysadmin user.](#)

T

- [Time zone, configuring the containers' time zone](#)

U

- [Usage, modifying the default usage](#)
- [User IDs, customizing user IDs of Black Duck containers](#)

W

- [Web server settings, configuring web server settings](#), such as configuring the hostname, host port, or disabling IPv6.

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.

Environment variables and scanning binaries

When you scan binaries with Black Duck - Binary Analysis (BDBA), you must ensure that the `HUB_SCAN_`

`ALLOW_PARTIAL= 'true'` parameter is added to the Job Runner container environment variables to surface components without versions in the BOM. The BDBA scanner, unlike Black Duck scanning, surfaces components without a version when version string information is not discernible in the binary. On the BOM, the component will have a question mark (?) beside the name to signal to the user that this component needs to be reviewed before security vulnerabilities are assigned to the component as Black Duck requires a version to map security vulnerabilities to a component.

Duplicate BOM Detection

To improve scan performance, the duplicate BOM detection feature is enabled by default.

If the feature determines that a scan will produce a BOM identical to the existing one, it skips the BOM computation. You can disable it by using the following setting:

```
SCAN_SERVICE_OPTS=-Dblackduck.scan.disableRedundantScans=true
```

You can change this setting in the `blackduck-config.env` file.

Note: In Black Duck 2021.4.0, this feature only impacts package manager (dependency) scans when the set of dependencies discovered by Detect is identical to the set from the previous scan. This capability will be extended in future releases.

Changing the expiration time for a bearer token

To extend the expiration time of a bearer token used in REST API, use the `docker-compose.local-overrides.yml` file to override the default setting by configuring the `HUB_AUTHENTICATION_ACCESS_TOKEN_EXPIRE` environment variable with the new expiration value in seconds.

The `HUB_AUTHENTICATION_ACCESS_TOKEN_EXPIRE` property is the number of seconds that the access tokens take to expire.

Note: The expiration configuration change only works for API tokens that are created after you change the setting in the `docker-compose.local-overrides.yml` file. The expiration time that you configure isn't updated for existing database records/API tokens when the setting is changed and the service is restarted.

About the KBMATCH_SENDFATH parameter

`KBMATCH_SENDFATH`: This parameter will exclude the file path and file name from being used for matching purposes and accuracy by our KnowledgeBase. Synopsys does not recommend changing this as it will potentially have some impact on your matching results.

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 `blackduck-config.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
<code>BLACKDUCK_HUB_CORS_ENABLED</code>	Required. Defines whether CORS is enabled; "true" indicates CORS is enabled.
<code>BLACKDUCK_CORS_ALLOWED_ORIGINS_PROP_NAME</code>	<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>
<code>BLACKDUCK_CORS_ALLOWED_HEADERS_PROP_NAME</code>	Optional. Headers that can be used to make the requests.
<code>BLACKDUCK_CORS_EXPOSED_HEADERS_PROP_NAME</code>	Optional. Headers that can be accessed by the browser requesting CORS.

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

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
```

Configuring the Dashboard refresh rate

Schedule the Dashboard refresh rate by configuring the `SEARCH_DASHBOARD_REFRESH_JOB_DUTY_CYCLE` environment variable in the `blackduck-config.env` file.

The allowed `SEARCH_DASHBOARD_REFRESH_JOB_DUTY_CYCLE` values are as follows:

- The default value is 20
- The minimum value is 10.
- The maximum value is 50.

Values that do not match the allowed values are reset to the nearest allowable value.

Examples:

`SEARCH_DASHBOARD_REFRESH_JOB_DUTY_CYCLE=100` The configured value is reset to the nearest allowable value (10, 20, or 50), which is 50.

`SEARCH_DASHBOARD_REFRESH_JOB_DUTY_CYCLE=10` The configured value stays at this allowable value, which is 10.

`SEARCH_DASHBOARD_REFRESH_JOB_DUTY_CYCLE=5` The configured value is reset to the nearest allowable value (10, 20, or 50), which is 10.

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 scanner cannot verify the certificate because it is a self-signed and is not issued by a CA.

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 webserver 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>
```

2. Add the secret to the webserver service in the docker-compose.local-overrides.yml file:

```
secrets:
  - WEBSERVER_CUSTOM_CERT_FILE
  - WEBSERVER_CUSTOM_KEY_FILE
```

3. Remove the comment character (#) from the secrets section located at the end of the docker-compose.local-overrides.yml file located in the docker-swarm directory:

```
secrets:
```

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

4. The healthcheck property in the webserver service the `docker-compose.local-overrides.yml` file must point to the new certificate from the secret:

```
webserver:
  healthcheck:
    test: [CMD, /usr/local/bin/docker-healthcheck.sh,
    'https://localhost:8443/health-
    checks/liveness', /run/secrets/WEBSERVER_CUSTOM_CERT_FILE]
```

5. Redeploy the stack by running the following command:

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

Using a custom certificate authority for certificate authentication

You can use your own certificate authority for certificate authentication.

⚙ To use a custom certificate authority

1. Add a docker secret called `AUTH_CUSTOM_CA`, the custom certificate authority certificate file, to the webserver and authentication services in the `docker-compose.local-overrides.yml` file located in the `docker-swarm` directory:

```
webserver:
  secrets:
    - AUTH_CUSTOM_CA

authentication:
  secrets:
    - AUTH_CUSTOM_CA
```

2. Add the following text to the end of the `docker-compose.local-overrides.yml` file located in the `docker-swarm` directory:

```
secrets:
  AUTH_CUSTOM_CA:
    file: {file path on host machine}
```

3. Start the webserver container and the authentication service.
4. Once the Black Duck services are up, make an API request which will return the Json Web Token

(JWT) with the certificate key pair that was signed with the trusted certificate authority. For example:

```
curl https://localhost:443/jwt/token --cert user.crt --key user.key
```

Note: The user name of the certificate used for authentication must exist in the Black Duck system as its Common Name (CN).

Getting component migration data from the KnowledgeBase

Use the migration APIs to get new or changed component IDs from the Black Duck KnowledgeBase. These APIs return the raw data and new IDs for migrated components in the KnowledgeBase.

You can use the API to submit an old component ID, for example, component ID: bf368a1d-ef4f-422c-baca-a138737595e7 to get the new component ID from the KnowledgeBase.

The migration tracking APIs can get migration information for a specific component or component version, or get details of migrations that occurred on specific dates.

Enabling the recording of migrations

To enable the recording of migrations, set the following property in the `blackduck-config.env` file.

`RECORD_MIGRATIONS = true` (Default is false)

This enables records to be written when migrations are detected.

Retention of migration data

To set the number of days for which records are returned, configure the following property in the `blackduck-config.env` file:

`MIGRATED_OBJECT_RETENTION_DAYS = <number_of_days>` (Default is 30 days)

API endpoints

Go to the API documentation to start using the following APIs.

For migrations that occurred after a specific date:

```
GET /api/component-migrations
```

For migrations for a specific component or component version:

```
GET /api/component-migrations/{componentOrVersionId}
```

Refer to the Black Duck API documentation at https://<blackduck_server>/api-doc/public.html#_component_component_version_migration_endpoints for more information.

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. Set the value to "true", for example, `HUB_HIERARCHICAL_BOM=true`. Resetting the value to "false" disables the feature.

You can also edit the webapp service in the `docker-compose.local-overrides.yml` file located in the `docker-swarm` directory:

```
webapp:
  environment: {HUB_HIERARCHICAL_BOM: "true"}
```

Note that if you have more than one environment variable, separate them with a comma (,). For example:

```
environment: {HUB_HIERARCHICAL_BOM: "true", HUB_MAX_MEMORY: 8192m}
```

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 `blackduck-config.env` file in the `docker-swarm` directory to "true".

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

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.

Note: All hosted customers should secure access to their Black Duck application by leveraging our out-of-the-box support for single sign on (SSO) via SAML or LDAP. Information on how to enable and configure these security features can be found in the installation guides. In addition, we encourage customers that are using a SAML SSO provider that offers two-factor authorization to also enable and leverage that technology to further secure access to their Black Duck application.

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 **Admin** icon (Admin).

The Administration page appears.

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

LDAP Server Details

☒ Enable LDAP

Server URL *

Authentication Type *

Manager DN

Manager Password

LDAP User Attributes

☒ Create user accounts automatically in Black Duck

User Search Base *

User Search Filter *

User DN Pattern

LDAP Attribute Mappings

First Name

Last Name

Email

LDAP Groups

☒ Synchronize LDAP groups

Group search base

Group filter

Group name attribute

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** section. In the **Server URL** field, ensure that you have configured the secure LDAP server: the protocol scheme is `ldaps://`.

5. Complete the information in the **LDAP User Attributes** section as described above.

Note: Optionally, deselect the **Create user accounts automatically in Black Duck** check box to turn off the automatic creation of users when they authenticate with LDAP. This check box is selected by default so users that don't exist in Black Duck are created automatically when they log into Black Duck using LDAP, which applies to new installs, and upgrades.

6. Enter the user credentials in the **Test Connection, User Authentication and Field Mapping** section and click **Test Connection**.
7. 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 * flast

Test Password *

Test Connection Test Connection ✓ Connection Test Succeeded

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

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

Certificate Problem

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

Certificate Details	
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

Cancel Save

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.

Use these 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 webapp service in the `docker-compose.local-overrides.yml` file located in the `docker-swarm` directory:

```
secrets:
  - LDAP_TRUST_STORE_PASSWORD_FILE
```

Add text, such as the following, to the `secrets` section located at the end of the `docker-compose.local-overrides.yml` file:

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

- Mount a directory that contains a file called `LDAP_TRUST_STORE_PASSWORD_FILE` to `/run/secrets` by adding a `volumes` section for the webapp service in the `docker-compose.local-overrides.yml` file located in the `docker-swarm` directory.

```
webapp:
  volumes: ['/directory/where/file/is:/run/secrets']
```

Note: You only need to mount a directory that contains the `LDAP_trust_store_password_file` if the trust store is fully replaced and it is protected by a different password.

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 **Admin** icon (Admin).

The Administration page appears.

3. Select **System Settings**.

The System Settings page appears.

Administration
System Settings

Logo
The dimensions will be constrained to a height of 34px and a maximum width of 150px.

SYNOPSYS Upload logo

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

Download Logs (.zip)

Legal Tab Visibility
Enables the Legal tab in Project Versions so project team members can check off license terms as fulfilled as part of their workflow. Note: this also requires license administrators to indicate which terms require fulfillment. See the documentation on license terms for more information.

Enable

Security Risk Configuration Ranking
Drag and drop the security risk configuration priority order
Warning: Changing the order of the security risk configuration will result in revised security risk calculations for all project version BOMs and may result in new policy violations. These calculations may take a considerable amount of time to complete.

Save

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 14 days. To modify this value:

1. Stop the containers.
2. Edit the `docker-compose.local-overrides.yml` file located in the `docker-swarm` directory:
 - a. Add the logstash service.
 - b. Add the `DAYS_TO_KEEP_LOGS` environment variable with the new value. This example purges log files after 10 days:

```
logstash:
  environment: {DAYS_TO_KEEP_LOGS: 10}
```

3. Restart the containers.

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:

- webapp
- jobrunner
- scan
- binaryscanner
- bomengine

Changing the default webapp container memory limits

There are three memory settings for the webapp 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 webapp 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.

Use the `webapp` section in the `docker-compose.local-overrides.yml` file and if necessary, remove the comment characters (`#`) and enter new values.

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.

Original values:

```
#webapp:
#environment: {HUB_MAX_MEMORY: REPLACE_WITH_NEW_MAX_MEMORYm}
#deploy:
#limits: {MEMORY: REPLACE_WITH_NEW_MEM_LIMITm}
#reservations: {MEMORY: REPLACE_WITH_NEW_VALUEm}
```

Updated values:

```
webapp:
environment: {HUB_MAX_MEMORY: 8192m}
```

```

deploy:
  limits: {MEMORY: 9216m}
  reservations: {MEMORY: 9216m}

```

Changing the default jobrunner container memory limits

There are three memory settings for the jobrunner 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 jobrunner 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.

Use the `jobrunner` section in the `docker-compose.local-overrides.yml` file and if necessary, remove the comment characters (`#`) and enter new values.

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

Original values:

```

#jobrunner:
#environment: {HUB_MAX_MEMORY: REPLACE_WITH_NEW_MAX_MEMORYm}
#deploy:
#  limits: {MEMORY: REPLACE_WITH_NEW_MEM_LIMITm}
#  reservations: {MEMORY: REPLACE_WITH_NEW_VALUEm}

```

Updated values:

```

jobrunner:
  environment: {HUB_MAX_MEMORY: 8192m}
  deploy:
    limits: {MEMORY: 9216m}
    reservations: {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.

Use the `scan` section in the `docker-compose.local-overrides.yml` file and if necessary, remove the comment characters (`#`) and enter new values.

The following example increases the maximum Java heap size for the scan container to 4GB and the value for the `limits memory` and `reservations memory` settings to 5GB each.

Original values:

```
#scan:
#environment: {HUB_MAX_MEMORY: REPLACE_WITH_NEW_MAX_MEMORYm}
#deploy:
#limits: {MEMORY: REPLACE_WITH_NEW_MEM_LIMITm}
#reservations: {MEMORY: REPLACE_WITH_NEW_VALUEm}
```

Updated values:

```
scan:
environment: {HUB_MAX_MEMORY: 4096m}
deploy:
  limits: {MEMORY: 5210m}
  reservations: {MEMORY: 5210m}
```

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.

Add the `binaryscanner` section to the `docker-compose.local-overrides.yml` file.

The following example changes the container memory limits to 4GB.

Updated values:

```
binaryscanner:
  limits: {MEMORY: 4096M}
```

Changing the default bomengine container memory limits

There are three memory settings for the bomengine 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 bomengine 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: These settings apply to all bomengine containers, including scaled bomengine containers.

Use the `bomengine` section in the `docker-compose.local-overrides.yml` file and if necessary, remove the comment characters (`#`) and then enter new values.

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

Original values:

```
#bomengine:
#environment: {HUB_MAX_MEMORY: REPLACE_WITH_NEW_MAX_MEMORYm}
#deploy:
#limits: {MEMORY: REPLACE_WITH_NEW_MEM_LIMITm}
#reservations: {MEMORY: REPLACE_WITH_NEW_VALUEm}
```

Updated values:

```
bomengine:
environment: {HUB_MAX_MEMORY: 8192m}
deploy:
limits: {MEMORY: 9216m}
reservations: {MEMORY: 9216m}
```

Important: Synopsys recommends allocating `HUB_MAX_MEMORY` and `limits memory` for the bomengine container using the same values as the jobrunner container.

Changing hostname for logstash

When you make changes to the hostname and service name for logstash, they are not pushed to the internal PostgreSQL container. For example, you might want to change the logstash hostname to `bdlogstash` because you're using logstash for another purpose and you now want to write PostgreSQL logs to `bdlogstash`; make the following changes:

1. In `blackduck-config.env` change `logstash` to `bdlogstash`.
`HUB_LOGSTASH_HOST=bdlogstash`
2. Edit `docker-compose.yml` and change `logstash` to `bdlogstash`.
`bdlogstash:`

```
image: blackducksoftware/blackduck-logstash:1.0.9
```

```
volumes: ['log-volume:/var/lib/logstash/data']
```

```
env_file: [blackduck-config.env]
```

3. Add `env_file: [blackduck-config.env]` to the postgres container definition in `docker-compose.yml` so that it reads the hostname change.
`env_file: [blackduck-config.env]`

Cleaning up unmapped code locations

Unmapped code locations are code locations that are not mapped to a project version. Internally in Black Duck, a code location is represented by one or more scans.

- Sometimes code scanning creates code locations without mapping them to a project version, which results in unmapped code locations.
- Code locations/scan data can be orphaned when users delete project versions.

Users who don't want to purge unmapped code location data might want to disable this feature in the `blackduck-config.env` file, which, by default, is set to true and to purge every 365 days.

Users who scan regularly and want to discard the data frequently might want to set the retention period to a low number of days, such as 14 days.

To schedule a cleanup of unmapped code locations, configure the following properties in the `blackduck-config.env` file:

- `BLACKDUCK_HUB_UNMAPPED_CODE_LOCATION_CLEANUP=true`
The default setting is true.
- `BLACKDUCK_HUB_UNMAPPED_CODE_LOCATION_RETENTION_DAYS=<number of days between 1-365, for example, 14>`
The default setting is 365 days.

Note: When you configure the cleanup of unmapped code locations and restart the system, the scan purge starts to remove unmapped code locations that meet the retention criteria (older than the configured number for retention days). By default, the scan purge job runs every 15 minutes.

Schedule a scan purge job by using a cron string

You can configure a scan purge by setting a variable in the `blackduck-config.env` file for docker swarm implementations.

The scan purge job can be scheduled by setting a variable in `blackduck-config.env` by using either of the following:

- using a cron job: `blackduck.scan.processor.scanpurge.cronstring`

Use the following cron format: `second minute hour dayOfMonth month daysOfWeek`

- using fixed delay: `blackduck.scan.processor.scanpurge.fixeddelay` (configured as the frequency to run `scanpurgejob` in milliseconds, which defaults to 15 minutes)

Note: If the `blackduck.scan.processor.scanpurge.cronstring` is provided, then the `blackduck.scan.processor.scanpurge.fixeddelay` setting is ignored because the cron string is used instead.

Clearing stuck BOM events

The BOM event cleanup job clears BOM events which might be stuck because of processing errors. By default, the job is run every day at midnight and removes stuck events prior to last 24 hours. Users can change the cron schedule depending on their system's quiet hours if needed, and also, change the last how many hours to keep them for, between 1 - 48.

- Set the retention periods for last how many hours to keep BOM events. This is useful to purge BOM events which may have been stuck due to processing errors or topology changes. Default is 24 hours. Valid range is 1 - 48 hours. For example, if you want to remove all stuck events prior to last 12 hours.

```
BLACKDUCK_BOM_EVENT_CLEANUP_BEFORE_HOURS=12
```

- Set the schedule via Cron expression when should job run which clears BOM events. It is recommended to run it around system's quiet hours. By default it runs at midnight and value is `0 0 * * * * ?`. For example, if BOM event clean up job should run at 2:00am every day

```
BLACKDUCK_BOM_EVENT_CLEANUP_JOB_CRON_TRIGGER="0 2 * * * ?"
```

The `VersionBomEventCleanupJob` is enabled by default and you can't disable this job.

Using the override file

You may want to override some of the default settings used by Black Duck. Instead of directly editing the `.yaml` file, use the `docker-compose.local-overrides.yaml`, located in the `docker-swarm` directory.

By using this file to modify default settings, your changes are preserved when you upgrade: you no longer need to modify the `.yaml` file after each Black Duck upgrade. T

Note in the `docker-compose` command, the `docker-compose.local-overrides.yaml` file *must* be the last `.yaml` file used. For example, the following command starts Black Duck using an external database:

```
docker stack deploy -c docker-compose.externaldb.yaml -c docker-compose.local-overrides.yaml hub
```

Configuring analytics in Black Duck

In Black Duck you can disable phone home globally for Synopsys Detect by turning off analytics in the `blackduck-config.env` file.

1. In the `blackduck-config.env` file, configure `ANALYTICS=false`.
2. Restart Black Duck.

Configuring an external PostgreSQL instance

Black Duck supports using an external PostgreSQL instance.

Synopsys recommends PostgreSQL 11.7 for new installs that use external PostgreSQL. Full support is retained for PostgreSQL 9.6 external databases. PostgreSQL 11.x is supported for external database users.

- Community PostgreSQL users must follow PostgreSQL 11 [instructions](#) for PostgreSQL upgrades.
- Users of third-party PostgreSQL, for example, RDS, must follow instructions from their providers.
- Users of the internal PostgreSQL container that uses PostgreSQL 9.6 are unaffected and don't need to make any changes. PostgreSQL 11.7 is not yet supported for the internal PostgreSQL container.
- Black Duck users on Black Duck versions prior to 4.0.0 must upgrade to 2020.4 before upgrading to 2020.6.0 or later.
- Numeric user names that consist of numbers only can be used for PostgreSQL user names in external PostgreSQL instances.

The updated `external-postgres-init.pgsql` script uses double quotation marks around the user names, so that numeric PostgreSQL user names can run successfully in the script. In the `external-postgres-init.pgsql` script, you search and replace the default name values for `HUB_POSTGRES_USER` and `POSTGRES_USER` with the numeric user names.

Migrating to Black Duck version 2020.6.x does not require migration to PostgreSQL 11.7 but you must note that PostgreSQL 11.7 is now the recommended option for external databases.

Important: PostgreSQL 10.x or PostgreSQL 12.x is not supported.

To configure an external PostgreSQL database:

1. Initialize the external PostgreSQL cluster with the UTF8 encoding. The method to accomplish this might depend on your external PostgreSQL provider. For example, when using the PostgreSQL `initdb` tool, run the following command:

```
initdb --encoding=UTF8 -D /path/to/data
```

2. Create and configure database usernames and passwords. There are three users for the PostgreSQL database: an administrator (by default, **blackduck** is the username), a user (by default, **blackduck_user** is the username), and a user for the Black Duck reporting database (by default, **blackduck_reporter** is the username). You can:
 - [Create accounts with the default usernames.](#)
 - [Create accounts with custom usernames.](#)
3. [Configure the PostgreSQL instance.](#)

⚙️ Creating and configuring accounts using default usernames

Use these instructions to use the default **blackduck**, **blackduck_user**, and **blackduck_reporter** usernames.

After completing these steps, go to [Configuring the 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. Replace the following in the `external-postgres-init.pgsql` file in the `docker-swarm` directory.
Replace `POSTGRES_USER` with `blackduck`
Replace `HUB_POSTGRES_USER` with `blackduck_user`
Replace `BLACKDUCK_USER_PASSWORD` with the password that you use for `blackduck_user`

Important: This step is mandatory.

3. Run the `external-postgres-init.pgsql` script, located in the `docker-swarm` directory, to create users, databases, and other necessary items. For example:

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

4. 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.

5. Go to [Configuring the PostgreSQL instance](#).

Note: Black Duck makes use of the `pgcrypto` extension and expects it to be available. CloudSQL, RDS, and Azure provide the `pgcrypto` extension by default, therefore, no further configuration is required. Users of community PostgreSQL will need to install the `postgresql-contrib` package if it is not already installed. Note that the package name varies depending upon distribution.

⚙️ Creating and configuring accounts using custom usernames and passwords

Use these instructions to create custom database usernames.

In these instructions:

- **DBAdminName** is the new custom administrator's username.
- **DBUserName** is the new custom database user's username.
- **DBReporterName** is the new custom database reporter's username.

After completing these steps, go to the next section, [Configuring the PostgreSQL instance](#).

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

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

No other specific values are required.

2. Edit the `external-postgres-init.pgsql` script, located in the `docker-swarm` directory with the

account names you wish to use for **DBAdminName**, **DBUserName**, and **DBReporterName**.

3. Run the edited `external-postgres-init.psql` script, located in the `docker-swarm` directory, to create users, databases, and other necessary items. For example:

```
psql -U DBAdminName -h <hostname> -p <port> -f external-postgres-init.psql postgres
```

4. Using your preferred PostgreSQL administration tool, configure passwords for the **DBAdminName**, **DBUserName**, and **DBReporterName** database users.

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

5. Edit the `hub-postgres.env` environment file. The file lists the default usernames for HUB_POSTGRES_USER and HUB_POSTGRES_ADMIN. Replace these default values with your custom usernames for the database user and administrator.
6. Go to the next section, [Configuring the PostgreSQL instance](#).

⚙️ Configuring the PostgreSQL instance

After creating users and configuring passwords, complete these steps:

1. Edit the `hub-postgres.env` environment file, located in the `docker-swarm` directory, to specify the database connection parameters. You can select to:

- Enable SSL in database connections.

For authentication, you can select to use certificate or username and password or both.

- Disable SSL in database connections.

If SSL is disabled, you must use username and password authentication.

Parameter	Description
HUB_POSTGRES_ENABLE_SSL	Defines the use of SSL in database connections. <ul style="list-style-type: none"> • Set the value to "false" to disable using SSL in database connections. This is the default value. • Set the value to "true" to enable using SSL in database connections.
HUB_POSTGRES_ENABLE_SSL_CERT	Defines whether a certificate is required for authentication. <ul style="list-style-type: none"> • Set the value to "false" to disable client certificate authentication. This is the default value. • Set the value to "true" to require client certificate authentication when using SSL in database connections.
HUB_POSTGRES_HOST	Hostname of the server with the PostgreSQL instance.
HUB_POSTGRES_PORT	Database port to connect to for the PostgreSQL instance.

2. If you are using username and password authentication, provide the PostgreSQL administrator and

user passwords to Black Duck :

- a. Create a file named `HUB_POSTGRES_USER_PASSWORD_FILE` with the password for the database user. This is the **blackduck_user** username if you are using the default username, or **DBUserName** in the previous example.
- b. Create a file named `HUB_POSTGRES_ADMIN_PASSWORD_FILE` with the password for the database administrator user. This is the **blackduck** username, if using the default username or **DBAdminName** in the previous example.
- c. Mount a directory that contains both files to `/run/secrets`. Use the `docker-compose.local-overrides.yml` file located in the `docker-swarm` directory. For each service (webapp, jobrunner, authentication, bomengine, and scan), do the following:
 - i. If necessary, remove the comment character (`#`) before the name of the service.
 - ii. Add the volume mount to the service.

This example adds the volume to the webapp service:

```
webapp:
  volumes: ['directory/of/password/files:/run/secrets']
```

You would also need to add this text to the authentication, jobrunner, bomengine, and scan services.

Instead of Steps 2a-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`.

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

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

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

- b. Add the password secret to the webapp, jobrunner, authentication, bomengine, and scan services in the `docker-compose.local-overrides.yml` file. This example is for the webapp service:

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

If necessary, remove the comment characters (`#`).

Remove the comment characters and if necessary, change the stack name to the text at the end of the `docker-compose.local-overrides.yml` file:


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

3. If you are using certificate authentication, mount a directory that contains all certificate files (HUB_POSTGRES_CA (server CA file), HUB_POSTGRES_CRT (client certificate file), HUB_POSTGRES_KEY (client key file)) to `/run/secrets` in the webapp, jobrunner, authentication, and scan services by editing the `docker-compose.local-overrides.yml` file located in the `docker-swarm` directory. See the example in Step 2c.
4. To be able to use SSL with certificate *and/or* username/password authentication, set HUB_POSTGRES_ENABLE_SSL_CERT to "true" and complete steps 2 and 3.
5. [Install](#) or [upgrade](#) Black Duck.

Modifying the PostgreSQL usernames for an existing external database

By default, the username of the PostgreSQL database user is **blackduck_user** and the username of the PostgreSQL administrator is **blackduck**.

If you are using an external PostgreSQL database, you can change these usernames.

These instructions are for an existing Black Duck instance in which the external database currently uses the **blackduck** and **blackduck_user** user names. To change the user names for a new configuration of an external database, follow the instructions in the previous section.

Important: For Black Duck database users who don't have administrator privileges, which is common with hosted providers such as GCP and RDS, connect to the `bds_hub` database and run `GRANT blackduck_user TO blackduck;`

⚙️ To modify the existing PostgreSQL account names

1. [Stop Black Duck](#).
2. Rename the users and reset the passwords in the `bds_hub` database.

```
alter user blackduck_user rename to NewName1 ;
alter user blackduck rename to NewName2 ;
alter user NewName1 password 'NewName1Password' ;
alter user NewName2 password 'NewName2Password' ;
```

3. In the `hub-postgres.env` file, located in the `docker-swarm` directory, edit the values for HUB_POSTGRES_USER AND HUB_POSTGRES_ADMIN. The value for HUB_POSTGRES_USER is the new username for **blackduck_user**. The value for HUB_POSTGRES_ADMIN is the new username for **blackduck**. For example:

```
HUB_POSTGRES_USER=NewName1
HUB_POSTGRES_ADMIN=NewName2
```

4. Restart Black Duck.

Note: In the 2020.4.0 release, the BDIO database was removed.

Configuring proxy settings

Edit the `blackduck-config.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
- Bomengine

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
- Bomengine
- 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. In the `docker-compose.local-overrides.yml` file, located in the `docker-swarm` directory, for each service (authentication, webapp, registration, jobrunner, and scan), provide access to the secret. This example is for the scan service:

```
scan:
  secrets:
    - HUB_PROXY_PASSWORD_FILE
```

If necessary, remove the comment characters (#).

3. In the `secrets` section at the end of the `docker-compose.local-overrides.yml` file, add the following:

```
secrets:
  HUB_PROXY_PASSWORD_FILE:
    external: true
    name: "hub_HUB_PROXY_PASSWORD_FILE"
```

If necessary, remove the comment characters (#).

You can use the `blackduck-config.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.

Note: If KB calls fail when the proxy password is provided by using the `docker secret HUB_PROXY_PASSWORD_FILE` in a Docker Swarm deployment, provide the password in the `blackduck-config.env` file to resolve the issue.

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. In the `docker-compose.local-overrides.yml` file, located in the `docker-swarm` directory,

provide access to the secret to these services: authentication, webapp, registration, jobrunner, and scan. This example is for the scan service:

```
scan:
  secrets:
    - HUB_PROXY_CERT_FILE
```

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
```

Scaling job runner, scan, bomengine, and binaryscanner containers

The job runner, scan, bomengine, 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 bomengine containers

This example adds a second bomengine container:

```
docker service scale hub_bomengine=2
```

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

```
docker service scale hub_bomengine=1
```

Note: Synopsys recommends that you scale bomengine containers to the same level as the jobrunner containers.

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 containers 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 containers 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 bdba-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 containers to a single container:

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

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.

Note: All hosted customers should secure access to their Black Duck application by leveraging our out-of-the-box support for single sign on (SSO) via SAML or LDAP. Information on how to enable and configure these security features can be found in the installation guides. In addition, we encourage customers that are using a SAML SSO provider that offers two-factor authorization to also enable and leverage that technology to further secure access to their Black Duck application.

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 can synchronize and obtain an external user's information such as firstname, lastname and email if the information is provided in attribute statements.

Note: The first name, last name, and email values are case-insensitive.

- You can configure the IdP to send groups in attribute statements with the attribute name of Groups. Black Duck can synchronize an external user's group information if you enable group synchronization in Black Duck.

Note: The Groups attribute is only synchronized when the **Enable Group Synchronization** check box is selected.

- The SAML assertion is expected to have a unique subject **NameID** value, which identifies the subject of a SAML assertion; typically the user who is being authenticated.
- 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.

For improved security, Black Duck now requires that you provide an assertion signature when you configure your Single Sign-On (SSO) Identity Provider (IdP). Although Synopsys does not recommend it, if your IdP is unable to provide this signature, you can disable this added security measure in the `blackduck-config.env` file.

⚙️ To disable SAML signature assertion do the following:

1. Open the `blackduck-config.env` file.
2. Find the `SAML_ASSERTION_SIGNATURE_VERIFICATION` property and set the value to `false`.
`SAML_ASSERTION_SIGNATURE_VERIFICATION=false`
3. Save the file and restart the system.

⚙️ To enable Single Sign-On using SAML



1. Click the **Admin** icon (Admin).

The Administration page appears.

2. Click the **SAML Integration** tab to display the SAML Integration page.

SAML Configuration Details

☐ Enable SAML

☐ Enable Group Synchronization

☒ Enable Local Logout Support ⓘ

☒ Create user accounts automatically in Black Duck

Service Provider Entity ID

Identity Provider Metadata ☒ URL ☐ XML File

External Black Duck Url

3. In the **SAML Configuration Details** settings, complete the following:
 - a. Select the **Enable SAML** check box to enable SAML.
 - 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. In the **Service Provider Entity ID** field, type the information for the Black Duck server in your environment using the format **https://host** where *host* is your Black Duck server.
- e. Optionally, deselect the **Create user accounts automatically in Black Duck** check box to turn off the automatic creation of users when they authenticate using SAML. This check box is selected by default so users that don't exist in Black Duck are created automatically when they log into Black Duck using SAML, which applies to new installs, and upgrades.
- f. For **Identity Provider Metadata** select one of the following:
 - **URL** Type the URL for your identity provider.
 - **XML File** You either drop the file or click in the area shown to open a dialog box from which you can select the XML file.
- g. In the **External Black Duck Url** field, type the URL of the public URL for the Black Duck server.


For example: `https://blackduck-docker01.dc1.lan`

4. Click **Save**.

After clicking **Save**, the **BlackDuck Metadata URL** field appears. You can copy the link or directly download the SAML XML configuration information.

To disable Single Sign-On using SAML



1. Click the **Admin** icon ().
The Administration page appears.
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**.

Uploading source files

BOM reviewers need to be able to easily confirm the results of a scan by confirming matches and investigating false negatives. When reviewing snippet matches, seeing a side-by-side comparison of the source file to the match can help in the evaluation and review of the match.

Black Duck provides the ability for you to upload your source files so that BOM reviewers can see the file contents from within the Black Duck UI.

When you enable deep license data detection or copyright text search during scanning, uploading source files enables BOM reviewers to view discovered license or copyright text from within the Black Duck UI. When files are uploaded, Black Duck provides a list of embedded licenses and copyright text and displays the highlighted text in the file.

For a BOM reviewer to view file content from within the Black Duck UI:

1. Administrators must enable the upload of source files.
 - a. The administrator enables the feature using an environment variable and provides a Docker-related secret consisting of a "user key" known as a seal key.
 - b. Black Duck retrieves the seal key and generates a "master key" which is used for encrypting and decrypting the source files.

The "master key" is stored encrypted via the "user key" (AES-GCM-256, key length of 32 bytes) and stored on a mounted volume.

2. Users with the Global or Project Code Scanner role must use the Signature Scanner and enable the **--upload-source** parameter. See the online help or User Guide for more information.

The scan client sends the source file contents to the Black Duck instance via SSL/TLS-secured endpoint(s) and with the proper authorization token.

The "master key" encrypts the files. Uploaded files are stored using their associated scan identifier and file signature and not by their file name.

In the Black Duck UI, the source file is transmitted via HTTPS over the network.

Note the following:

- Ensure that you have enough disk space for file uploads.
- The maximum total source size that you can upload at one time is 4 GB (4000 MB). This value is configurable.
- Uploaded files are deleted after 180 days. This value is configurable.
- Files are deleted when the upload service reaches 95% of the maximum disk setting.

The service deletes the oldest files until the disk space is equal to 90% of the maximum disk setting.

To enable file upload

1. Create the "user key" by creating a file named SEAL_KEY on the host system. This file should be stored in a safe location. It must be 32 bytes or 32 characters in length. For example:

```
vi opt/secrets/SEAL_KEY
```

2. Enable this feature by setting the ENABLE_SOURCE_UPLOADS environment variable in the `blackduck-config.env` file located in the `docker-swarm` directory, to true.

```
ENABLE_SOURCE_UPLOADS=true
```

3. Use the Docker secret command to tell Docker Swarm the key by using SEAL_KEY. The name of the secret must include the stack name. In the following example, the stack name is 'hub':

```
docker secret create hub_SEAL_KEY <location of key file>
```

4. Provide the upload-cache service access to the secret by removing the comment characters (#) from

the `uploadcache` section to the `docker-compose.local-overrides.yml` file located in the `docker-swarm` directory:

```
uploadcache:
```

```
secrets:
  - SEAL_KEY
```

5. Remove the comment characters (`#`) from the following text to the end of the `docker-compose.local-overrides.yml` file located in the `docker-swarm` directory:

```
secrets:
  SEAL_KEY:
    external: true
    name: "hub_SEAL_KEY"
```

6. [Restart the containers.](#)

You can now use the Signature Scanner to upload the source files.

Backing up the seal key and raw master key

Back up the seal key and raw master key. If these keys are not backed up, you may lose data if the seal key is lost.

```
./bd_get_source_upload_master_key.sh <local destination of raw master key>
<path to SEAL_KEY>
```

Replacing the seal key

If you lose your seal key, you can replace it if you backed up the existing master key, as described in the previous section.

To replace the seal key, use the following script:

```
./recover_master_key.sh <new seal key file> <local destination of master key>
```

Additional configuration options

Black Duck provides these additional configurations:

- Data retention. By default file data is retained for 180 days.

Use the `DATA_RETENTION_IN_DAYS` environment variable to change this setting.

- Total source size. By default, the maximum total source size is 4 GB (4000 MB).

Use the `MAX_TOTAL_SOURCE_SIZE_MB` environment variable to change this setting.

To modify these settings, add the environment variable to the upload cache service in the `docker-compose.local-overrides.yml` file located in the `docker-swarm` directory and define the new setting. For example:

```
uploadcache:
  environment: [MAX_TOTAL_SOURCE_SIZE_MB: 8000]
```

You can also limit the size of the uploaded file. By default, the maximum size is 5 MB. To modify this setting, at the command line enter:

```
export SCAN_CLI_OPTS="-Dblackduck.scan.cli.file.upload.limitMB=#"
```

where # is the new value in MB. For example, to increase the maximum size to 6 MB, enter:

```
export SCAN_CLI_OPTS="-Dblackduck.scan.cli.file.upload.limitMB=6"
```

Starting or stopping Black Duck

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

Starting up Black Duck

Use these commands if you have not used the override file to modify the default configuration settings.

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

- If you are running Black Duck with a read-only file system for Swarm services, add the `docker-compose.readonly.yml` file to the previous instructions.

For example, to install Black Duck with the PostgreSQL database container, enter the follow command:

```
docker swarm init
```

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

Starting up Black Duck when using the override file

Use these commands if you have used the override file to modify the default configuration settings.

Note: The `docker-compose.local-overrides.yml` file *must* be the last `.yml` file used in the `docker-compose` command.

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

```
docker swarm init
```

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

- If you are running Black Duck with a read-only file system for Swarm services, add the `docker-compose.readonly.yml` file to the previous instructions.

For example, to install Black Duck with the PostgreSQL database container, enter the follow command:

```
docker swarm init
```

```
docker stack deploy -c docker-compose.yml -c docker-compose.readonly.yml -c docker-compose.local-overrides.yml hub
```

Shutting down Black Duck

- Run the following command to shut down Black Duck:

```
docker stack rm hub
```

Configuring user session timeout

Configure the user session timeout value to automatically log out users from the Black Duck server, and align with your corporate security policy.

1. To view the current timeout value, make the following GET request:

```
GET https://<Black-Duck-server>/api/system-oauth-client
```

Note: Users must have read permission for the OAuth Client to use the GET method.

2. To change the current timeout value, make the following PUT request with the PUT request body.

```
PUT https://<Black-Duck-server>/api/system-oauth-client
```

```
{
  "accessTokenValiditySeconds": <time value in seconds>
}
```

Note: Users must have permission to update the OAuth Client to use the PUT method for this task. The system administrator role includes the required permissions.

The value that you type in the PUT request body is the new timeout value.

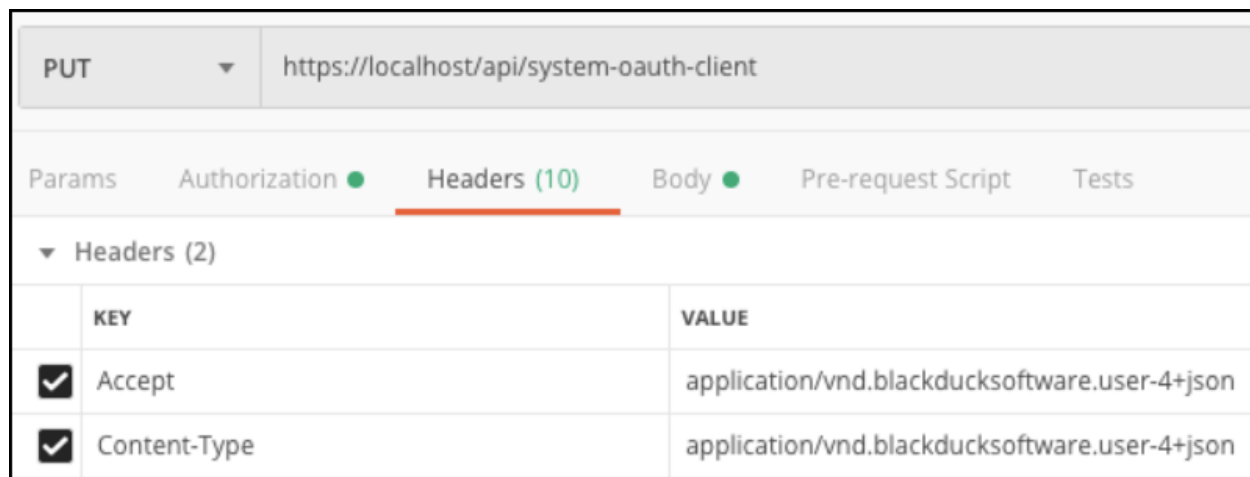
Timeout values between 30 minutes (1800 seconds) and 24 hours (86400) are accepted.

The following media types are accepted:

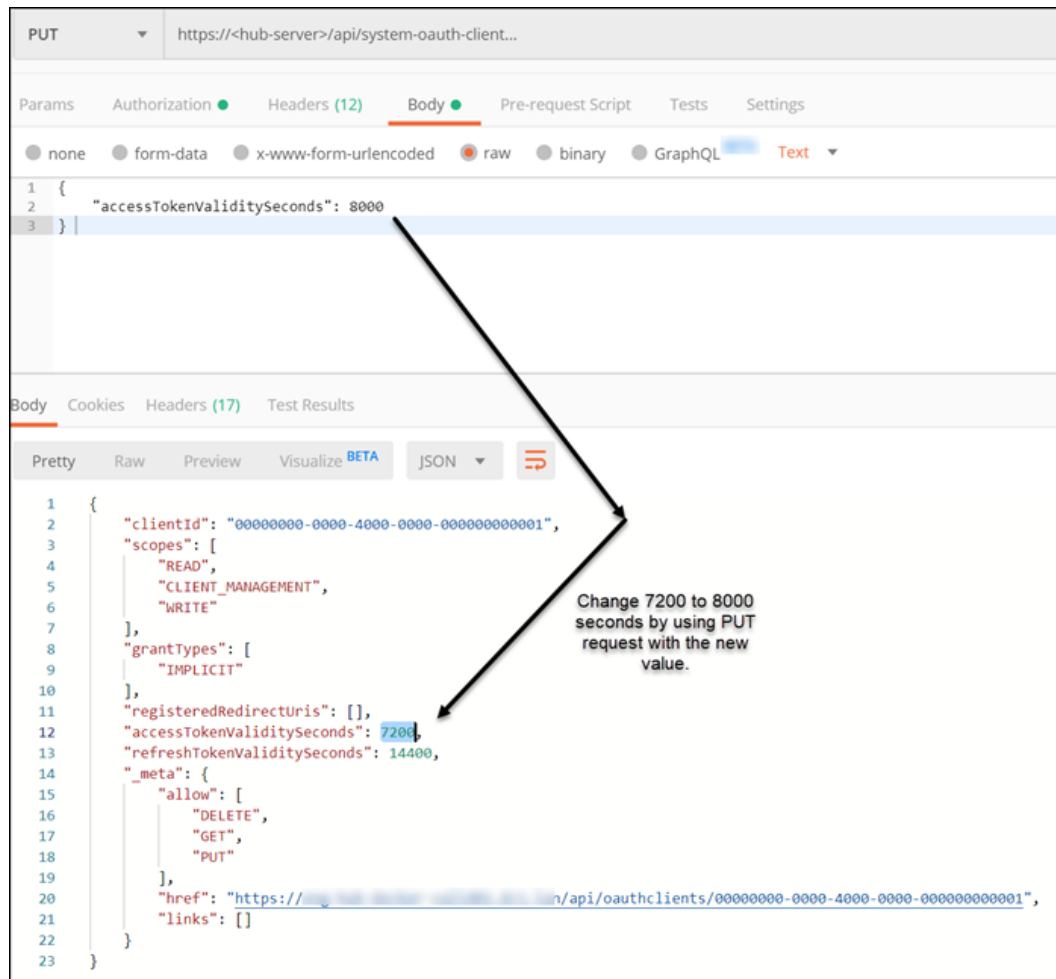
application/vnd.blackducksoftware.user-4+json

application/json

Here's an example in Postman:



In the following example, you change the timeout value from 7200 to 8000 seconds.



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.

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.

To edit the default email address that is associated with the sysadmin user, go to the **User Management** page on the Black Duck UI, select the **sysadmin** user name, change the email address and save. To see the change, you must log out and log in again.

To access the **User Management** page, the user account that you use must have the **Super User** role, which is assigned, by default, to the sysadmin account. The main purpose of the email address is as a contact reference for the user account.

Configuring Black Duck reporting delay

In Black Duck 2021.4.0 the reporting database job process runs every 480 minutes, which is configurable.

⚙ To configure a different reporting delay

1. Edit the `blackduck-config.env` file in the `docker-swarm` directory and configure `BLACKDUCK_REPORTING_DELAY_MINUTES=<value in minutes>`

For example, `BLACKDUCK_REPORTING_DELAY_MINUTES=360`

2. Restart the containers.

Configuring the containers' time zone

By default, the time zone for Black Duck containers is UTC. For monitoring purposes, you may want to change this value so that the timestamps shown in logs reflect the local time zone.

⚙ To configure a different time zone

1. Set the value of the TZ environment variable in the `blackduck-config.env` file in the `docker-swarm` directory to the new time zone. Use the values shown in Wikipedia, as shown [here](#).

For example, to change the timezone to that used in Denver, Colorado, enter:

```
TZ=America/Denver
```

2. Restart the containers.

Modifying the default usage

Usage indicates how a component is intended to be included in the project when this version is released.

Possible usage values are:

- **Statically Linked.** A tightly-integrated component that is statically linked in and distributed with your project.
- **Dynamically Linked.** A moderately-integrated component that is dynamically linked in, such as with DLLs or .jar files.
- **Source Code.** Source code such as .java or .cpp files.
- **Dev Tool / Excluded.** Component will not be included in the released project. For example, a component that is used internally for building, development, or testing. Examples are unit tests, IDE files, or a compiler.
- **Separate Work.** Intended for loosely-integrated components. Your work is not derived from the component. To be considered a separate work, your application has its own executables, with no linking between the component and your application. An example is including the free Acrobat PDF Viewer with your distribution media.
- **Implementation of Standard.** Intended for cases where you implemented according to a standard. For example, a Java spec request that ships with your project.
- **Merely Aggregated.** Intended for components that your project does not use or depend upon in any way, although they may be on the same media. For example, a sample version of an unrelated product included with your distribution.
- **Prerequisite.** Intended for components that are required but not provided by your distribution.
- **Unspecified.** The usage for this component has not yet been determined.

In Black Duck version 2020.10.0, Synopsys introduced the UNSPECIFIED usage value, which you can use as a default option instead of existing defaults to eliminate confusion over whether the component is assigned a true usage value or a default usage value. For example, if you use DYNAMICALLY_LINKED as the default usage value, you might not be able to distinguish between components that have a true usage value of DYNAMICALLY_LINKED and those components that are assigned the usage value by default. By using UNSPECIFIED as the usage default, you know that you should take action to assign a valid usage value when you see UNSPECIFIED assigned as the usage value.

The default usage is determined by match type: Snippets have a usage of Source Code while all other match types are Dynamically Linked.

Black Duck uses the following variables so that you can change the default usage for similar match types:

- **BLACKDUCK_HUB_FILE_USAGE_DEFAULT.** Defining a usage for this variable sets the default value for the following match types:
 - Binary
 - Exact Directory
 - Exact File
 - Files Added/Deleted
 - Files Modified
 - Partial
- **BLACKDUCK_HUB_DEPENDENCY_USAGE_DEFAULT.** Defining a usage for this variable sets the default value for the following match types:
 - File Dependency

- Direct Dependency
- Transitive Dependency
- `BLACKDUCK_HUB_SOURCE_USAGE_DEFAULT`. Defining a usage for this variable sets the default value for the following match types:
 - Snippet
- `BLACKDUCK_HUB_MANUAL_USAGE_DEFAULT`. Defining a usage for this variable sets the default value for the following match types:
 - Manually Added
 - Manually Identified

⚙️ To configure different usage values

1. Edit the `blackduck-config.env` file located in the `docker-swarm` directory to the new usage values by removing the comment icon (`#`) and entering a value. Use one of the usage values as shown in the file: `SOURCE_CODE`, `STATICALLY_LINKED`, `DYNAMICALLY_LINKED`, `SEPARATE_WORK`, `IMPLEMENTATION_OF_STANDARD`, `DEV_TOOL_EXCLUDED`, `MERELY_AGGREGATED`, `PREREQUISITE`, `UNSPECIFIED`

For example, to change default usage for files to Unspecified:

```
BLACKDUCK_HUB_FILE_USAGE_DEFAULT=UNSPECIFIED
```

Note: If you enter the incorrect usage text, the original default value will still apply. A warning message will appear in the log files of the jobrunner container.

The modified usage values apply to any new scans or rescans.

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 (`blackduck-authentication`): 100
- Binaryscanner (`bdba-worker`): 0
- CA (`blackduck-cfssl`): 100
- DB (`blackduck-postgres`): 70
- Documentation (`blackduck-documentation`): 8080
- Job Runner (`blackduck-jobrunner`): 100
- Logstash (`blackduck-logstash`): 100
- RabbitMQ (`rabbitmq`): 100
- Registration (`blackduck-registration`): 8080
- Scan (`blackduck-scan`): 8080
- Web App (`blackduck-webapp`): 8080

- webserver (blackduck-nginx): 100
- Uploadcache (blackduck-uploadcache): 100

Changing the UID consists of adding the new value for a container to the `docker-compose.local-overrides.yml` located in the `docker-swarm` directory. Add the

`user: UID_NewValue:root` line in the container's section.

The following example changes the UID for the webapp container to 1001:

```
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).
- 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.

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. Add the new host port value to the `docker-compose.local-overrides.yml` file located in the `docker-swarm` directory.

Use the `webserver` section to add the port information using the following format: `ports:`

`['NewValue:8443']` For example to change the port to 8443:

```
webserver:
  ports: ['8443:8443']
```

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

```
PUBLIC_HUB_WEBSERVER_PORT=8443
```

3. Comment out the following part in the `docker-compose.yml` file to disable access to port 443 on the host, otherwise, users can still access the host using port 443.

```
webserver:
  #ports: ['443:8443']
```

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.

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](#).

Chapter 6: Upgrading Black Duck

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

Important: Black Duck users on versions prior to 4.0.0 must upgrade to 2020.4 before upgrading to 2020.6 or later.

The upgrade instructions depend on your previous version of Black Duck:

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

Note: For customers upgrading from a version prior to 2019.8.0, two jobs, the VulnerabilityReprioritizationJob and the VulnerabilitySummaryFetchJob run at start up to synchronize vulnerability data. These jobs may take some time to run and the overall vulnerability score for existing BOMs will not be available until these jobs complete. Users with the System Administrator role can use the Black Duck Jobs page to monitor these jobs.

Note: When upgrading from a version prior to 2018.12.0, you will experience a longer than usual upgrade time due to a data migration that is needed to support new features in this release. Upgrade times will depend on the size of the Black Duck database. If you would like to monitor the upgrade process, please contact Synopsys Customer Support for instructions.

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-2021.4.0.tar.gz
```

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

```
tar xvf hub-2021.4.0.tar
```

Download using the wget command

1. Run the following command:

```
wget https://github.com/blackducksoftware/hub/archive/v2021.4.0.tar.gz
```

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

```
gunzip v2021.4.0.tar.gz
```

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

```
tar xvf v2021.4.0.tar
```

Migration script to purge unused rows in the audit event table

During an upgrade, a migration script is run to purge rows that are no longer used in the `audit_event` table because of changes to the reporting database. This script might take a long time to run, depending on the size of the `audit_event` table. For example, the migration script takes approximately 20 minutes to run against a 350 GB `audit_event` table.


Important: Any upgrade from a pre 2019.12.0 Black Duck version to 2019.12.0 or later versions requires that the migration script is run for one upgrade only because of changes to the reporting database.

To determine the size of the audit event table, do one of the following tasks:

- From the `bds_hub` database, run the following command:

```
SELECT pg_size_pretty( pg_total_relation_size('st.audit_event') );
```
- Log in to the Black Duck UI as system administrator and do the following steps:



1. Click the expanding menu icon () and select **Administration**.
2. On the Administration page, select **System Information**.
The System Information page appears.
3. Select **db** in the left column of the page.

- Find the `total_tbl_size` value for the `audit_event` tablename in the Table Sizes table.

Table Sizes (100 biggest sorted by size)					
schemaname	tablename	total_tbl_size_pretty	tbl_size_pretty	total_tbl_size	tbl_size
st	scan_composite_leaf	6508 MB	4232 MB	6823731200	4437254144
st	audit_event	2027 MB	1577 MB	2125168640	1653915648

When the migration script is finished running, Synopsys strongly recommends that you run the `VACUUM` command on the `audit_event` table to optimize PostgreSQL performance.

- Depending on your system usage, running the `VACUUM` command can reclaim a significant amount of disk space no longer in use by Black Duck.
- By running this command, querying performance will be improved.

Note: If you don't run the `VACUUM` command, there may be a degradation of performance.

Important: You must ensure you have enough space to run the `VACUUM` command, otherwise, it will fail by running out of disk space and possibly corrupt the entire database. The `VACUUM` command requires twice the amount of disk space that is currently being used by the `audit_event` table.

To run the `VACUUM` command with containerized PostgreSQL database deployments, do the following steps:

- Get the size of the `audit_event` table and ensure that you have enough space to run the `VACUUM` command.
- Run the `docker ps` command to get the ID of the PostgreSQL container.
- Run the following command to access the PostgreSQL container.
`docker exec -it <container_ID> psql bds_hub`
- Run the following `VACUUM` command to reclaim space that is no longer used.

```
VACUUM FULL ANALYZE st.audit_event;
```

If you have an external PostgreSQL database deployment, you must determine the size of your `audit_event` table, execute the `VACUUM` command, and when it's finished, you restart the deployment.

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>
```

To migrate a specific database, use the following syntax:

```
hub_db_migrate.sh <db name> <path to dump file>
```

The acceptable values for <db name> are as follows:

- bds_hub
- bds_hub_report

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.yml -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.yml -c docker-compose.bdba.yml hub` (Using an external database with Black Duck - Binary Analysis)

If you wish to install Black Duck with a read-only file system for Swarm services, add the following `docker-compose.readonly.yml` file to the instructions listed in step 3.

For example, to install Black Duck with the PostgreSQL database container, enter the follow command:

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

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>
```

To migrate a specific database, use the following syntax:

```
hub_db_migrate.sh <db name> <path to dump file>
```

The acceptable values for `<db name>` are as follows:

- `bds_hub`
- `bds_hub_report`

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-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.yml -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.yml -c docker-`

```
compose.bdba.yml hub (Using an external database with Black Duck - Binary Analysis)
```

If you wish to install Black Duck with a read-only file system for Swarm services, add the following `docker-compose.readonly.yml` file to the instructions listed above.

For example, to install Black Duck with the PostgreSQL database container, enter the follow command:

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

Upgrading from an existing Docker architecture

To upgrade from a previous version of Black Duck:

1. Migrate your PostgreSQL databases.

PostgreSQL 11.7 is recommended and supported for external databases, whilst support is retained for PostgreSQL 9.6 external databases.

For users of the internal PostgreSQL container, PostgreSQL 9.6 is the supported version for 2020.8.x. PostgreSQL 11.7 is not yet supported for the internal PostgreSQL container.

Important: Migrating to Black Duck 2020.8.x does not require migration to PostgreSQL 11.7 for external databases but users must note that PostgreSQL 11.7 is recommended, whilst full support for PostgreSQL 9.6 is retained. For users of the internal PostgreSQL container, PostgreSQL 9.6 remains as the supported version for 2020.8.x.

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 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.

Note: 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.

If you are upgrading from version 4.2.0 or later, then migrating your databases 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 databases

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

1. Backing up the original PostgreSQL databases: the Black Duck database (`bds_hub`), the reporting database (`bds_hub_report`).
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), you can migrate your data from PostgreSQL 9.6 to a 11.7 instance of PostgreSQL and configure your system to point to that instance. The pre-upgraded system will only work with PostgreSQL 9.6, and the post-upgraded system will work with PostgreSQL 9.6 or 11.7.

Note: As of the Black Duck 2019.10.0 release, the `bds_hub_report` database is no longer in use. Synopsys recommends that VulnDB users or former VulnDB users back up the `bds_hub_report` database. For all other users, data that was in the `bds_hub_report` database is now in the `bds_hub` database.

⚙️ To back up the PostgreSQL databases

1. Run the `hub_create_data_dump.sh` script which creates PostgreSQL data files in the `blackduck-postgres` container and then copies the files from the container to a local directory.

Important: You must run the Black Duck version <version you're backing up/dumping from> of the `hub_create_data_dump.sh` script located in the `docker-swarm/bin` directory. For example, if you're dumping a 2019.10.0 Docker-swarm database use the dump script in 2019.10.0 `docker-swarm/bin` directory.

```
./hub_create_data_dump.sh <local directory to store PostgreSQL data files>
```

This script will create a number of data backup files (`globals.sql`, `bds_hub.dump`, `bds_hub_report.dump`).

⚙️ 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 which restores the data from the existing database data files.

Important: You must run the Black Duck version <version you're restoring from> of the `hub_db_migrate.sh` script located in the `docker-swarm/bin` directory. For example, to restore a 2019.10.0 Docker-swarm database, you must restore it by using the 2019.10.0 `hub_db_migrate.sh` script into a 2019.10.0 Black Duck stack.

```
./hub_db_migrate.sh <local directory to load PostgreSQL data files>
```

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>
```

To migrate a specific database, use the following syntax:

```
hub_db_migrate.sh <db name> <path to dump file>
```

The acceptable values for <db name> are as follows:

- `bds_hub`
- `bds_hub_report`

You can now upgrade Black Duck.

Upgrading Black Duck

⚙️ To upgrade Black Duck:

1. Do one of the following:
 - If you did not use the `docker-compose.local-overrides.yml` to modify the `.yml` file, 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.externaldb.yml hub` (using an external PostgreSQL instance)
 - If you are upgrading Black Duck - Binary Analysis, 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 -c docker-compose.bdba.yml hub` (using the DB container with Black Duck - Binary Analysis)
 - `docker stack deploy -c docker-compose.externaldb.yml -c docker-compose.bdba.yml hub` (using an external database with Black Duck - Binary Analysis)
 - If you used the `docker-compose.local-overrides.yml` to modify the `.yml` file, run one of the following commands. Use the version of the version of the `docker-compose.local-overrides.yml` file that contains your modifications; for all other files, use the files located in

the `docker-swarm` directory in the newer version of Black Duck:

- `docker stack deploy -c docker-compose.yml -c docker-compose.local-overrides.yml hub` (using the DB container)
- `docker stack deploy -c docker-compose.externaldb.yml -c docker-compose.local-overrides.yml hub` (using an external PostgreSQL instance)
- `docker stack deploy -c docker-compose.yml -c docker-compose.bdba.yml -c docker-compose.local-overrides.yml hub` (using the DB container with Black Duck - Binary Analysis)
- `docker stack deploy -c docker-compose.externaldb.yml -c docker-compose.bdba.yml -c docker-compose.local-overrides.yml hub` (using an external database with Black Duck - Binary Analysis)
- To upgrade Black Duck with a file system as read-only for Swarm services, add the `docker-compose.readonly.yml` file to the previous examples.

For example, If you used the `docker-compose.local-overrides.yml` to modify the `.yml` file and wish to upgrade a Black Duck installation that uses the DB container, run the following command: `docker stack deploy -c docker-compose.yml -c docker-compose.readonly.yml -c docker-compose.local-overrides.yml hub`

Note: If you previously edited the `hub-proxy.env` file, those edits must be copied over to the `blackduck-config.env` file.

Appendix A: Docker containers

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

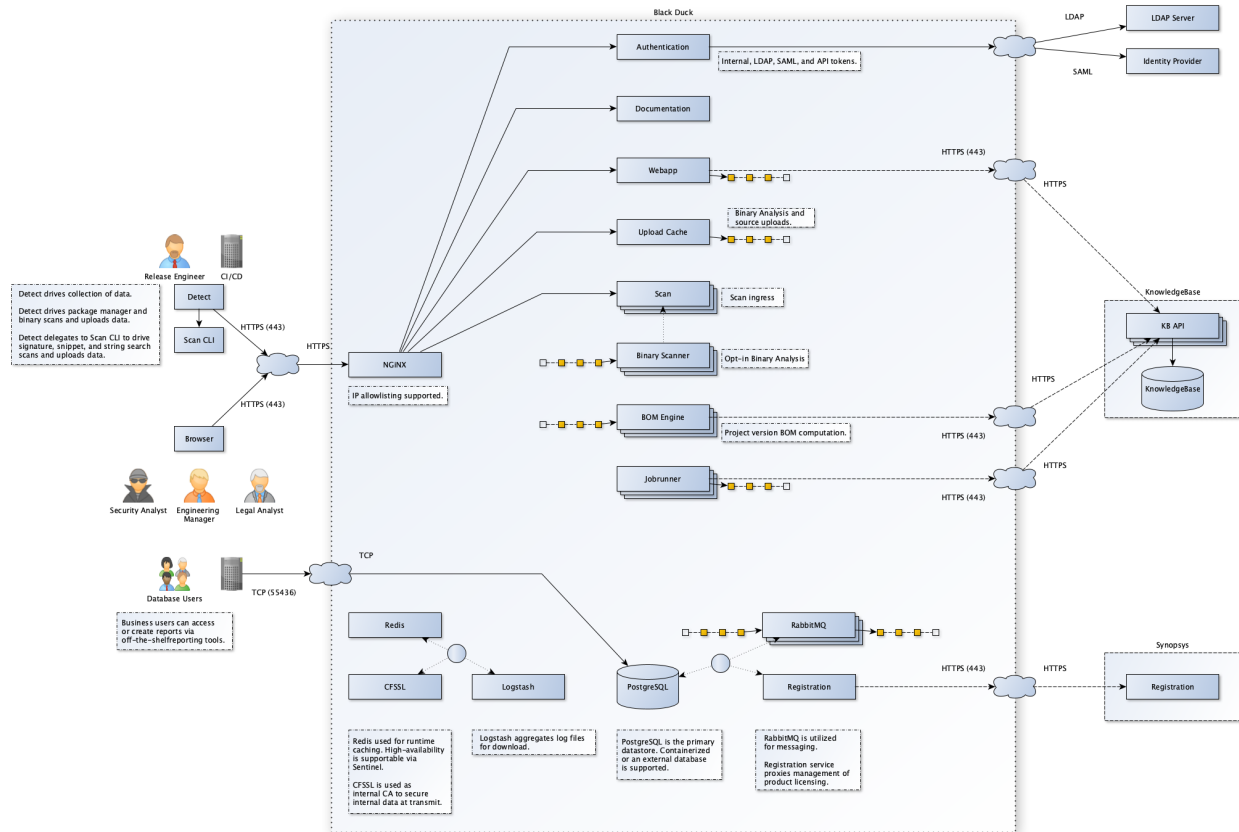
1. Authentication
2. CA
3. DB

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

4. Documentation
5. Jobrunner
6. Logstash
7. Registration
8. Scan
9. Uploadcache - This container is also used for Black Duck - Binary Analysis.
10. Webapp
11. Webserver
12. Redis
13. Rabbitmq
14. Bomengine

If Black Duck - Binary Analysis is enabled, the Binaryscanner container is required.

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



The Zookeeper container was removed in Black Duck version 2020.4.0.

You can manually remove the following zookeeper volumes because they are no longer used:

- zookeeper-data-volume
- zookeeper-datalog-volume

The following tables provide more information on each container.

Authentication container

Container Name: blackduck-authentication	
Image Name	blackduckssoftware/blackduck-authentication:2021.4.0
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.

Container Name: blackduck-authentication	
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"> • postgres • cfssl • logstash • registration • webapp <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 Swarm uses. These environment variables can be set to override the default host names:</p> <ul style="list-style-type: none"> • postgres - \$HUB_POSTGRES_HOST • cfssl - \$HUB_CFSSL_HOST • logstash - \$HUB_LOGSTASH_HOST • registration - \$HUB_REGISTRATION_HOST • webapp - \$HUB_WEBAPP_HOST
Resources/Constraints	<ul style="list-style-type: none"> • Default max Java heap size: 512MB • Container memory: 1GB • Container CPU: 1 CPU
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>
Environment File	<code>blackduck-config.env</code>

CA container

Container Name: blackduck-cfssl	
Image Name	blackducksoftware/blackduck-cfssl:1.0.1
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">• Default max Java heap size: N/A• Container memory: 512MB• Container CPU: Unspecified
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>
Environment File	blackduck-config.env

DB container

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

Container Name: blackduck-postgres	
Image Name	blackducksoftware/blackduck-postgres:1.0.16
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.
Links/Ports	<p>The DB container needs to connect to these containers/services:</p> <ul style="list-style-type: none"> • logstash • cfssl <p>The container needs to expose port 5432 to other containers that will link to it within the Docker network.</p> <p>This container exposes port 55436 outside of the Docker network for database reporting.</p>
Alternate Host Name Environment Variables	<p>There are times when running in other types of orchestrations that it is useful to have host names set for these containers that are not the default that Docker Swarm uses. These environment variables can be set to override the default host names:</p> <ul style="list-style-type: none"> • logstash: <code>\$HUB_LOGSTASH_HOST</code> • cfssl: <code>\$HUB_CFSSL_HOST</code>
Resource/Constraints	<ul style="list-style-type: none"> • Default max Java heap size: N/A • Container memory: 3GB • Container CPU: 1 CPU
Users/Groups	This container runs as UID 70. If the container is started

Container Name: blackduck-postgres	
	<p>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>
Environment File	N/A

Documentation container

Container Name: blackduck-documentation	
Image Name	blackducksoftware/blackduck-documentation:2021.4.0
Description	The Documentation container supplies documentation for the application.
Scalability	There is a single instance of this container. It should not be scaled.
Links/Ports	<p>This container must connect to these other containers/services:</p> <ul style="list-style-type: none"> • logstash • cfssl <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 Swarm uses. These environment variables can be set to override the default host names:</p> <ul style="list-style-type: none"> • logstash: \$HUB_LOGSTASH_HOST • cfssl: \$HUB_CFSSL_HOST
Resources/Constraints	<ul style="list-style-type: none"> • Default Max Java Heap Size: 512MB • Container Memory: 512MB • Container CPU: unspecified
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>
Environment File	blackduck-config.env

Jobrunner container

Container Name: blackduck-jobrunner	
Image Name	blackducksoftware/blackduck-jobrunner:2021.4.0
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	<p>The Job Runner container needs to connect to these containers/services:</p> <ul style="list-style-type: none"> • postgres • registration • logstash • cfssl
Alternate Host Name Environment Variables	<p>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:</p> <ul style="list-style-type: none"> • postgres: \$HUB_POSTGRES_HOST • registration: \$HUB_REGISTRATION_HOST • logstash: \$HUB_LOGSTASH_HOST • cfssl: \$HUB_CFSSL_HOST
Resources/Constraints	<ul style="list-style-type: none"> • Default max Java heap size: 4GB • Container memory: 4.5GB • Container CPU: 1 CPU
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>
Environment File	blackduck-config.env

Logstash container

Container Name: blackduck-logstash	
Image Name	blackducksoftware/blackduck-logstash:1.0.9
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"> • Default max Java heap size: 1GB • Container memory: 1GB • Container CPU: Unspecified
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>
Environment File	blackduck-config.env

Registration container

Container Name: blackduck-registration	
Image Name	blackducksoftware/blackduck-registration:2021.4.0
Description	The container is a small service that handles registration requests from the other containers. At periodic intervals, this container connects to the Black Duck Registration Service and obtains registration updates.
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"> • logstash • cfssl <p>The container needs to expose port 8443 to other containers that link to it.</p>
Alternate Host Name Environment Variables	There are times when running in other types of orchestrations that it is useful to have host names set for

Container Name: blackduck-registration	
	<p>these containers that are not the default that Docker Swarm uses. These environment variables can be set to override the default host names:</p> <ul style="list-style-type: none"> • logstash: \$HUB_LOGSTASH_HOST • cfssl: \$HUB_CFSSL_HOST
Resources/Constraints	<ul style="list-style-type: none"> • Default max Java heap size: 512MB • Container memory: 640MB • Container CPU: Unspecified
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. 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>
Environment File	blackduck-config.env

Scan container

Container Name: blackduck-scan	
Image Name	blackducksoftware/blackduck-scan:2021.4.0
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"> • postgres • registration • logstash • cfssl <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 Swarm uses. These environment variables can be set to override the default host names:</p> <ul style="list-style-type: none"> • postgres: \$HUB_POSTGRES_HOST • registration: \$HUB_REGISTRATION_HOST

Container Name: blackduck-scan	
	<ul style="list-style-type: none"> • logstash: \$HUB_LOGSTASH_HOST • cfssl: \$HUB_CFSSL_HOST
Resources/Constraints	<ul style="list-style-type: none"> • Default max Java heap size: 2GB • Container memory: 2.5GB • Container CPU: 1 CPU
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>
Environment File	<code>blackduck-config.env</code>

Uploadcache container

Container Name: Uploadcache	
Image Name	<code>blackducksoftware/blackduck-upload-cache:1.0.16</code>
Description	This container will be used to temporarily store uploads for binary analysis and when uploading source files for snippet matching.
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"> • cfssl • logstash • rabbitmq (if Black Duck - Binary Analysis is enabled) <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 Swarm uses. These environment variables can be set to override the default host names:</p> <ul style="list-style-type: none"> • cfssl: \$HUB_CFSSL_HOST • logstash: \$HUB_LOGSTASH_HOST • rabbitmq: \$RABBIT_MQ_HOST

Container Name: Uploadcache	
Constraints	<ul style="list-style-type: none"> • Default max Java heap size: N/A • Container memory: 512MB • Container CPU: Unspecified
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).
Environment File	hub-bdba.env blackduck-config.env

Webapp container

Container Name: blackduck-webapp	
Image Name	blackducksoftware/blackduck-webapp:2021.4.0
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.
Links/Ports	<p>The webapp container needs to connect to these containers/services:</p> <ul style="list-style-type: none"> • postgres • registration • logstash • cfssl <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 Swarm uses. These environment variables can be set to override the default host names:</p> <ul style="list-style-type: none"> • postgres: \$HUB_POSTGRES_HOST • registration: \$HUB_REGISTRATION_HOST

Container Name: blackduck-webapp	
	<ul style="list-style-type: none"> • logstash: \$HUB_LOGSTASH_HOST • cfssl: \$HUB_CFSSL_HOST
Resources/Constraints	<ul style="list-style-type: none"> • Default max Java heap size: 2GB • Container memory: 2.5GB • Container CPU: 1 CPU
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>
Environment File	blackduck-config.env

Webserver container

Container Name: blackduck-webserver	
Image Name	blackducksoftware/blackduck-nginx:1.0.31
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> <p>HTTP/2 and TLS 1.3 are supported</p>
Scalability	The container should not be scaled.
Links/Ports	<p>The Web App container needs to connect to these containers/services:</p> <ul style="list-style-type: none"> • webapp • cfssl • documentation • scan • authentication • upload cache (if Black Duck - Binary Analysis is enabled) <p>This container exposes port 443 outside of the Docker network.</p>
Alternate Host Name Environment Variables	There are times when running in other types of orchestrations that it is useful to have host names set for these containers that are not the default that Docker Swarm uses. These environment variables can be set to

Container Name: blackduck-webserver	
	<p>override the default host names:</p> <ul style="list-style-type: none"> • webapp: \$HUB_WEBAPP_HOST • cfssl: \$HUB_CFSSL_HOST • scan: \$HUB_SCAN_HOST • documentation: \$HUB_DOC_HOST • authentication: \$HUB_AUTHENTICATION_HOST • upload cache: \$HUB_UPLOAD_CACHE_HOST
Resources/Constraints	<ul style="list-style-type: none"> • Default max Java heap size: N/A • Container memory: 512MB • Container CPU: Unspecified
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>
Environment File	hub-webserver.env, blackduck-config.env

Redis container

Container Name: blackduck-redis	
Image Name	blackducksoftware/blackduck-redis:2021.4.0
Description	<p>This container enables more consistent caching functionality in Black Duck and is used to improve application performance.</p> <p>Redis is enabled by default as a primary cache mechanism.</p>
Configuration	<p>To configure Redis, use the following settings:</p> <ul style="list-style-type: none"> ■ blackduck-config.env environment file to configure redis settings <ul style="list-style-type: none"> • Redis settings: <ul style="list-style-type: none"> ◦ BLACKDUCK_REDIS_MODE: Redis mode can be standalone or sentinel ◦ BLACKDUCK_REDIS_TLS_ENABLED: Whether to enforce TLS/SSL connections between Redis client and server.

Container Name: blackduck-redis	
	<ul style="list-style-type: none"> ■ Use docker-compose.yml for Redis standalone mode which requires 1,024 MB additional memory ■ Use both docker-compose.yml and docker-compose.redis.sentinel.yml for redis sentinel mode which provides high availability but also requires 3,168 MB additional memory. ■ Redis container integrates with logstash and filebeat like other services. ■ Redis container has a health status check. ■ In Black Duck Hub system info page, there is a redis-cache tab where Redis debug info is shown.
Scalability	The container should not be scaled.
Links/Ports	<p>The Redis container needs to connect to these containers/services:</p> <ul style="list-style-type: none"> • logstash • cfssl
Alternate Host Name Environment Variables	N/A
Resources/Constraints	<ul style="list-style-type: none"> • Default max Java heap size: N/A • Container memory: <ul style="list-style-type: none"> • Standalone 1024 MB • Sentinel mode 3168 MB • Container CPU: 1
Users/Groups	Can run as any user/group, for example. {"runAsUser": 4567, "runAsGroup": 4567}
Environment File	blackduck-config.env

Rabbitmq container

Container Name: rabbitmq	
Image Name	blackducksoftware/rabbitmq:1.2.2
Description	This container facilitates upload information to the binary analysis worker. It also enables the bomengine container to receive notification to start BOM computation. It exposes ports within the Docker network, but not outside the Docker network.
Scalability	There should only be a single instance of this container. It should not be scaled.

Container Name: rabbitmq	
Links/Ports	<p>This container needs to connect to these other containers/services:</p> <ul style="list-style-type: none"> • cfssl <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 Swarm uses. These environment variables can be set to override the default host names:</p> <ul style="list-style-type: none"> • cfssl: \$HUB_CFSSL_HOST
Constraints	<ul style="list-style-type: none"> • Default max Java heap size: N/A • Container memory: 1GB • Container CPU: Unspecified
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 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>
Environment File	blackduck-config.env

bomengine container

Container Name: bomengine	
Image Name	blackducksoftware/blackduck-bomengine:2021.4.0
Description	The bomengine container is responsible for building BOMs and keeping them up-to-date.
Scalability	The container can be scaled
Links/Ports	<p>The bomengine container needs to connect to the following container/services:</p> <ul style="list-style-type: none"> • postgres • registration • logstash • cfssl
Alternate Host Name Environment Variables	There are times when running in other types of orchestrations that any individual service name may be

Container Name: bomengine	
	<p>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:</p> <ul style="list-style-type: none"> • postgres: \$HUB_POSTGRES_HOST • registration: \$HUB_REGISTRATION_HOST • logstash: \$HUB_LOGSTASH_HOST • cfssl: \$HUB_CFSSL_HOST
Resources/Constraints	<ul style="list-style-type: none"> • Default max Java heap size: N/A • Container memory: 4.5GB
Users/Groups	<p>This container runs as UID 100</p> <p>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).</p>
Environment File	<code>blackduck-config.env</code>

Black Duck - Binary Analysis container

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

Binaryscanner container

Container Name: bdba-worker	
Image Name	image: sigsynopsys/bdba-worker:2021.03
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.

Container Name: bdba-worker	
Links/Ports	<p>This container needs to connect to these containers/services:</p> <ul style="list-style-type: none"> • cfssl • logstash • rabbitmq • webserver <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 Swarm uses. These environment variables can be set to override the default host names:</p> <ul style="list-style-type: none"> • cfssl: \$HUB_CFSSL_HOST • logstash: \$HUB_LOGSTASH_HOST • rabbitmq: \$RABBIT_MQ_HOST • webserver: \$HUB_WEBSERVER_HOST
Resources/Constraints	<ul style="list-style-type: none"> • Default max Java heap size: N/A • Container memory: 2GB • Container CPU: 1 CPU
Users/Groups	This container runs as UID 0.
Environment File	hub-bdba.env