

Installing Black Duck SCA using Docker Swarm

Black Duck SCA 2026.7.2

Copyright ©2026 by Black Duck.

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.

03-11-2026

Contents

Preface....7
Black Duck documentation....7
Customer support....7
Black Duck Community....8
Training....8
Black Duck Statement on Inclusivity and Diversity....8
Black Duck Security Commitments....9

1. Overview of Black Duck Deployment with Docker Swarm....10
Components hosted on Black Duck servers....10

2. Installation planning....11
Getting started....11
New installations....11
Upgrading from a previous version of Black Duck....11
Hardware requirements....11
Docker requirements....12
Docker Version....13
Installing Docker Engine on RHEL....13
Operating systems....13
Software requirements....13
Network requirements....14
Database requirements....15
PostgreSQL Versions Supported by Black Duck SCA....16
Migrating on Swarm Overview....16
Proxy server requirements....17
Configuring your NGiNX server to work with Black Duck....17
Amazon services....18
Additional port information....18
Configuring the keepalive setting....19
KnowledgeBase Feedback Service....19
User-agent analytics....20
Disabling the feedback service....20

3. Installing Black Duck....21
Installation files....21
Download from the GitHub page....21
Download using the wget command....22
Distribution....22
Installing Black Duck....23
Rapid Scanning in Black Duck....25

4. Administrative tasks....27
Environment files and variables....27

Duplicate BOM Detection 27
Changing the expiration time for a bearer token 27
Accessing the API documentation through a proxy server 28
Providing access to the REST APIs from a non-Black Duck server 28
Increasing the size of the binary scan file 29
Configuring the Dashboard refresh rate 29
Generating and Managing Certificates 30
    Using custom certificates 31
Getting component migration data from the KnowledgeBase 32
    Enabling the recording of migrations 33
    Retention of migration data 33
    API endpoints 33
Including ignored components in reports 33
Configuring Secure LDAP 33
    Obtaining your LDAP information 34
    Importing the server certificate 34
    LDAP Trust Store Password 35
Downloading log files and heatmap data 36
    Viewing log files 36
Changing the default memory limits 37
    Changing the default webapp container memory limits 37
    Changing the default jobrunner container memory limits 38
    Changing the default scan container memory limits 39
    Changing the default binaryscanner container memory limits 40
    Changing the default bomengine container memory limits 40
Changing hostname for logstash 41
Cleaning up unmapped code locations 41
    Schedule a scan purge job by using a cron string 42
Clearing stuck BOM events 42
Using the override file 42
Configuring analytics in Black Duck 43
Configuring an external PostgreSQL instance 43
    Modifying the PostgreSQL usernames for an existing external database 46
    Using mTLS with Docker Swarm for External Databases 47
Configuring proxy settings 48
Configuring the report database password 50
Scaling job runner, scan, bomengine, and binaryscanner containers 51
    Scaling bomengine containers 51
    Scaling job runner containers 51
    Scaling scan containers 51
    Scaling binaryscanner containers 51
Configuring SAML for Single Sign-On 52
Uploading source files 54
Starting or stopping Black Duck 55
    Starting up Black Duck 55
    Starting up Black Duck when using the override file 56
    Shutting down Black Duck 56
Configuring user session timeout 56
Providing your Black Duck system information to Customer Support 58
Understanding the default sysadmin user 58
Configuring Black Duck reporting delay 59
Configuring the containers' time zone 59
Modifying the default usage 59
    Match Types for bdio2 uploaded jsonld/bdio file 60

Customizing user IDs of Black Duck containers....61
Configuring Web server settings....62
Configuring the hostname....62
Configuring the host port....62
Disabling IPv6....63
Create BOM reports using UTF8 character encoding....63
Scan monitoring....63
Configuring HTML report download size....64
Configuring KB license update and security update jobs....64
Configuring secrets encryption in Black Duck....64
Generating seeds....65
Configuring a backup seed....66
Managing secret rotation....66
Configuring custom volumes for Blackduck Storage....67
Configuring multiple volumes....67
Migrating between volumes....70
Configuring storage volumes for reports....71
Configuring jobrunner thread pools....71
Configuring Rapid Scan for BOM support....72
Changing the long running job threshold....72
Enabling SCM Integration....72
Configuring automatic scan retry header....72
Configuring hierarchical subproject license conflicts....73
Provisioning JWT public/private key pairs....73
Configuring session token invalidation....74
Configuring Preserve Component Relationships on SBOM Import....74

5. Uninstalling Black Duck....75

6. Upgrading Black Duck....76
Installation files....76
Download from the GitHub page....76
Download using the wget command....76
Migration script to purge unused rows in the audit event table....77
Upgrading from an existing Docker architecture....78
Backing up and restoring data....80

7. Docker containers....81
Authentication container....82
Binaryscanner container....83
Bomengine container....84
CFSSL container....84
DB container....85
Documentation container....86
Integration container....87
Jobrunner container....87
Logstash container....88
Rabbitmq container....89
Redis container....89
Registration container....90

Scanmatch container 91
Storage container 92
Webapp container 92
Webserver container 93

Preface

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.
Installing Black Duck using Kubernetes	install_kubernetes.pdf	Contains information about installing and upgrading Black Duck using Kubernetes.
Installing Black Duck using OpenShift	install_openshift.pdf	Contains information about installing and upgrading Black Duck using OpenShift.
Getting Started	getting_started.pdf	Provides first-time users with information on using Black Duck.
Scanning Best Practices	scanning_best_practices.pdf	Provides best practices for scanning.
Getting Started with the SDK	getting_started_sdk.pdf	Contains overview information and a sample use case.
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 Helm. Click the following links to view the documentation.

- Helm is a package manager for Kubernetes that you can use to install Black Duck. Black Duck supports Helm3 and the minimum version of Kubernetes is 1.13.

Black Duck integration documentation is available on:

- https://sig-product-docs.blackduck.com/bundle/detect/page/integrations/integrations.html
- https://docs.blackduck.com/p/bridgeintegrations

Customer support

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

- Online: https://community.blackduck.com/s/contactsupport
- To open a support case, please log in to the Black Duck Community site at https://community.blackduck.com/s/contactsupport.
- Another convenient resource available at all times is the online Community portal.

Black Duck Community

The Black Duck 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 Black Duck 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 Black Duck product users to allow you to learn valuable lessons from a diverse group of industry leading companies. In addition, the Customer Hub puts all the latest product news and updates from Black Duck at your fingertips, helping you to better utilize our products and services to maximize the value of open source within your organization.
- Solve – Quickly and easily get the answers you're seeking with the access to rich content and product knowledge from Black Duck experts and our Knowledgebase.
- Share – Collaborate and connect with Black Duck staff and other customers to crowdsource solutions and share your thoughts on product direction.

Access the Customer Success Community. If you do not have an account or have trouble accessing the system, click here to get started, or send an email to community.manager@blackduck.com.

Training

Black Duck Customer Education 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.

In Black Duck Education, 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://blackduck.skilljar.com/page/black-duck or for help with Black Duck, select **Black Duck Tutorials** from the Help menu () in the Black Duck UI.

Black Duck Statement on Inclusivity and Diversity

Black Duck 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.

Black Duck Security Commitments

As an organization dedicated to protecting and securing our customers' applications, Black Duck is equally committed to our customers' data security and privacy. This statement is meant to provide Black Duck customers and prospects with the latest information about our systems, compliance certifications, processes, and other security-related activities.

This statement is available at: Security Commitments | Black Duck

1. Overview of Black Duck Deployment with Docker Swarm

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.

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 Installing Black Duck for installation instructions.
3. Review the Administrative Tasks section.

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 Upgrading Black Duck for upgrade instructions.
3. Review the Administrative Tasks section.

Hardware requirements

Supported systems

Black Duck supports the following systems for installation and operation:

- 64-bit x86
- ARM64 (AArch64)

Note: BDBA and RL-service are currently not supported on ARM systems.

Black Duck hardware scaling guidelines

For scalability sizing guidelines, see Black Duck Hardware Scaling Guidelines.

Black Duck database

DANGER: Do not delete data from the Black Duck database (`bds_hub`) unless directed to do so by a Black Duck Technical Support representative. Be sure to follow appropriate backup procedures. Deletion of data will cause errors ranging from UI problems to complete failure of Black Duck to start. Black Duck Technical Support cannot recreate deleted data. If no backups are available, Black Duck will provide support on a best-effort basis.

Disk space requirements

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.

BDBA scaling

BDBA scaling is done by adjusting the number of binaryscanner replicas and by adding PostgreSQL resources based on the expected number of binary scans per hour that will be performed. For every 15 binary scans per hour, add the following:

- One binaryscanner replica
- One CPU for PostgreSQL
- 4GB memory to PostgreSQL

If your anticipated scan rate is not a multiple of 15, round up. For example, 24 binary scans per hour would require the following:

- Two binaryscanner replicas,
- Two additional CPUs for PostgreSQL, and
- 8GB additional memory for PostgreSQL.

This guidance is valid when binary scans are 20% or less of the total scan volume (by count of scans).

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

Docker requirements

Docker Swarm, which is the preferred method for installing Black Duck SCA, 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 recommends running Black Duck SCA on a single node Swarm deployment. For full scalability sizing guidelines, see the Container Scalability section of the Black Duck SCA Release Notes.

There are these restrictions when using Black Duck in Docker Swarm:

CAUTION: Do not run anti-virus scans on the PostgreSQL data directory. Antivirus software opens lots of files, puts locks on files, etc. Those things interfere with PostgresSQL operations. Specific errors vary by product, but usually involve the inability of PostgresSQL to access its data files. One example is that PostgresSQL fails with the error "too many open files in system".

- 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 SCA installation supports Docker versions 25.0.2, 28.x, and 29.x.

Note: Docker versions 23.x, 26.x and 27.x are no longer supported as of Black Duck SCA 2026.4.0.

Installing Docker Engine on RHEL

Please note that Docker currently only provides packages for RHEL on s390x (IBM Z). Other architectures are not yet supported for RHEL. Please refer to the Docker and Redhat pages for details.

Operating systems

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

- Red Hat Enterprise Linux server 8.9 and 9.x
- Ubuntu 22.04, 24.04
- SUSE Linux Enterprise server version 12.x (64-bit)
- Oracle Enterprise Linux 7.9

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:

- Safari Version 17.4.1
  - Safari versions 14 and below are no longer supported
- Chrome Version 123.0.6312.124 (Official Build) (x86_64)
  - Chrome versions 91 and below are no longer supported
- Firefox Version 124.0.2 (64-bit)
  - Firefox versions 89 and below are no longer supported
- Microsoft Edge Version 123.0.2420.97 (Official build) (64-bit)
  - Microsoft Edge versions 91 and below are no longer supported

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 SCA requires the following ports to be externally accessible:

- Port 443 – Web server HTTPS port for Black Duck SCA via NGiNX
- Port 55436 – Read-only database port from PostgreSQL for reporting

If your corporate security policy requires registration of specific URLs, connectivity from your Black Duck SCA 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 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.

Allow list addresses and IP ranges

Note: HTTPS is used for all traffic to Black Duck SCA. IPs that include a subnet mask (for example, /22 in 103.21.244.0/22) represent a range of IPs, all of which should be allow listed to ensure Black Duck SCA functions as intended.

Ensure that the following addresses and IPs are on the allow list:

Domain	IP Address(es)
kb.blackducksoftware.com	34.160.126.173, 34.149.112.69, 34.111.46.24, 35.224.73.200, 35.242.234.51, 35.220.236.106
updates.suite.blackducksoftware.com	35.244.241.173
scass.blackduck.com	35.244.200.22
na.scass.blackduck.com	35.244.200.22
na.store.scass.blackduck.com	34.54.95.139
eu.store.scass.blackduck.com	34.54.213.11
eu.scass.blackduck.com	34.54.38.252
repo.blackduck.com	34.149.5.115
production.cloudflare.docker.com	173.245.48.0/20, 103.21.244.0/22, 103.22.200.0/22, 103.31.4.0/22, 141.101.64.0/18, 108.162.192.0/18, 190.93.240.0/20, 188.114.96.0/20, 197.234.240.0/22, 198.41.128.0/17, 162.158.0.0/15, 104.16.0.0/13, 104.24.0.0/14, 172.64.0.0/13, 131.0.72.0/22

Domain	IP Address(es)
hub.docker.com	44.219.3.189, 3.224.227.198, 44.193.181.103
docker.io	44.219.3.189, 3.224.227.198, 44.193.181.103
auth.docker.io	34.226.69.105, 54.196.99.49, 3.219.239.5
registry-1.docker.io	54.196.99.49, 3.219.239.5, 34.226.69.105
github.com	140.82.116.4
static.cloud.coveo.com	
search.cloud.coveo.com	
platform.cloud.coveo.com	
platformhipaa.cloud.coveo.com (for HIPAA compliant deployments)	

Note:

Customers not utilizing the AI-assisted documentation search can still modify the CSP headers by using environment variables for the `blackduck-nginx` container:

```
HUB_CSP_HEADER=default-src 'none'; connect-src 'self'; object-src 'self'; script-src
 'self';
img-src 'self' data: https://s3.amazonaws.com/cloud.ohloh.net/; style-src 'self' 'unsafe-
inline';
font-src 'self'; frame-ancestors 'self'; frame-src 'self';
```

Verifying connectivity

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.

IPv4 and IPv6 networks

Black Duck SCA supports IPv4 and IPv6 for ingress and egress traffic. This includes connectivity between Black Duck components, the KnowledgeBase, customer systems, and internet-facing networking pods.

For deployments in IPv6-only environments, ensure that your networking configuration supports IPv6 routing for all required communication paths.

Database requirements

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

CAUTION: Do not delete data from the Black Duck database (bds_hub) unless directed to do so by a Black Duck Technical Support representative. Be sure to follow appropriate backup procedures. Deletion of data will cause errors ranging from UI problems to complete failure of Black Duck to start. Black Duck Technical Support cannot recreate deleted data. If no backups are available, Black Duck will provide support on a best-effort basis.

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 2026.7.0, Black Duck recommends PostgreSQL 18.x for new installs that use external PostgreSQL. PostgreSQL 15.x is no longer supported for external PostgreSQL instances. For users of the internal PostgreSQL container, PostgreSQL 16 is provided.

Refer to:

- PostgreSQL Version Upgrade Schedule for more information regarding supported PostgreSQL versions in Black Duck SCA.
- Configuring an external PostgreSQL instance for more information on setting up your own PostgreSQL instance.

Note: For PostgreSQL sizing guidelines, see the Black Duck Hardware Scaling Guidelines.

PostgreSQL Versions Supported by Black Duck SCA

Black Duck SCA continuously updates its support for PostgreSQL versions to enhance the performance and reliability of its services. This document summarizes supported versions for both internal PostgreSQL containers and external PostgreSQL instances, along with migration guidance.

Supported PostgreSQL Versions

Internal PostgreSQL Container:

- As of **Black Duck SCA 2025.10.0**, PostgreSQL 16 is the supported version for the internal PostgreSQL container.
- Starting with **Black Duck SCA 2023.10.0**, PostgreSQL settings are automatically configured for deployments using the internal PostgreSQL container.

External PostgreSQL Instances:

- For new external PostgreSQL installations, Black Duck recommends using the latest stable version, **PostgreSQL 18**.

Important Notes

- **PostgreSQL Sizing:** Refer to the Black Duck SCA Hardware Scaling Guidelines for sizing recommendations.
- **Antivirus Caution:** Avoid running antivirus scans on the PostgreSQL data directory. Antivirus software may lock files and interfere with database operations, potentially causing errors such as "too many open files in the system."

Migrating on Swarm Overview

- The migration is completely automatic; no additional actions are needed beyond those for a standard Black Duck upgrade.
- The blackduck-postgres-upgrader container MUST run as root to make the layout and UID changes described above.
- On subsequent Black Duck restarts, blackduck-postgres-upgrader will determine that no migration is needed and immediately exit.

- OPTIONAL: After a successful migration, the blackduck-postgres-upgrader container no longer needs to run as root.

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: Black Duck recommends that you use PostgreSQL 15 (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 Black Duck 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 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.

3. Installing Black Duck

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

Black Duck Installation Requirements	
Hardware requirements	
√	You have ensured that your hardware meets the minimum hardware requirements.
Docker requirements	
√	You have ensured that your system meets the docker requirements.
Software requirements	
√	You have ensured that your system and potential clients meet the software requirements.
Network requirements	
√	You have ensured that your network meets the network requirements. Specifically: • 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	
√	You have selected your database configuration. Specifically, you have configured database settings if you are using an external PostgreSQL instance.
Proxy requirements	
√	You have ensured that your network meets the proxy requirements. Configure proxy settings before or after installing Black Duck.
Web server requirements	
√	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-2026.7.2.tar.gz
```

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

```
tar xvf hub-2026.7.2.tar
```

Download using the wget command

1. Run the following command:

```
wget https://github.com/blackducksoftware/hub/archive/v2026.7.2.tar.gz
```

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

```
gunzip v2026.7.2.tar.gz
```

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

```
tar xvf v2026.7.2.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.
- `docker-compose.integration.yaml`: Used to enable SCM integration. No further configuration is required. The following environment variable will be automatically added:

```
webserver:
  environment: {ENABLE_INTEGRATION_SERVICE: "true"}
```

- `external-postgres-init.pgsql`: PostgresSQL.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.

In the sizes-gen02 directory:

- `resources.yaml`: These are resource definition files for Enhanced scanning.

In the sizes-gen03 directory:

- `10sph.yaml`, `120sph.yaml`, `250sph.yaml`, `500sph.yaml`, `1000sph.yaml`, `1500sph.yaml`, `2000sph.yaml`: These are the resource definition files for the various scans per hour sizes. See the Hardware Requirements section for more information.

In the sizes-gen04 directory:

- `10sph.yaml`, `120sph.yaml`, `250sph.yaml`, `500sph.yaml`, `1000sph.yaml`, `1500sph.yaml`, `2000sph.yaml`: These are the resource definition files for the various scans per hour sizes introduced in Black Duck 2024.1.0. See the Hardware Requirements section for more information.

Installing Black Duck

Prior to installing Black Duck, determine if there are any settings that need to be configured. See the Administrative tasks section for additional assistance with system configuration.

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.

Installing Black Duck with the PostgreSQL database container

1.
```
docker swarm init
```

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

2.
```
docker stack deploy -c docker-compose.yml -c sizes-gen05/120sph.yaml -c docker-compose.local-overrides.yml hub
```

   With Black Duck 2022.4.0 and later, you no longer allocate container resources directly in `docker-compose.yml`. Instead, resources are specified in a separate overrides file. The resource allocations used before Black Duck 2022.4.0 are found in `sizes-gen02/resources.yaml`. For Black Duck 2024.1.0 and later, multiple possible allocations are provided in the `sizes-gen05` folder. There are 7 allocations based on load as measured in average scans per hour; if your anticipated load does not match one of the predefined allocations, round up.

   In the example above:

   - `docker-compose.yml`: Stock deployment, this file is not to be edited.

- `sizes-gen05/120.yaml`: The resource definition file. In this example, `sizes-gen05/120.yaml` is used as the desired resource definition, but this can be changed to better fit your scanning patterns and usage. See the Distribution section for a list of all available options. Please note the system requirements for each option as they increase with greater resource definitions.
- `docker-compose.local-overrides.yml`: This is an optional file if your installation has custom configurations like secrets, memory limits, etc.

Installing Black Duck with Black Duck Binary Analysis using the PostgreSQL database container

1. ```
   docker swarm init
   ```

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

2. ```
   docker stack deploy -c docker-compose.yml -c sizes-gen05/120sph.yaml -c docker-compose.bdba.yml hub
   ```

   The resource definition file used above can be changed to better fit your scanning patterns and usage. See the Distribution section for a list of all available options. Please note the system requirements for each option as they increase with greater resource definitions.

Installing Black Duck with an external PostgreSQL instance

1. ```
   docker swarm init
   ```

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

2. ```
   docker stack deploy -c docker-compose.externaldb.yml -c sizes-gen05/120sph.yaml hub
   ```

   The resource definition file used above can be changed to better fit your scanning patterns and usage. See the Distribution section for a list of all available options. Please note the system requirements for each option as they increase with greater resource definitions.

Installing Black Duck with Black Duck Binary Analysis using an external PostgreSQL instance

1. ```
   docker swarm init
   ```

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

2. ```
   docker stack deploy -c docker-compose.externaldb.yml -c sizes-gen05/120sph.yaml -c docker-compose.bdba.yml hub
   ```

   The resource definition file used above can be changed to better fit your scanning patterns and usage. See the Distribution section for a list of all available options. Please note the system requirements for each option as they increase with greater resource definitions.

   Note: When using the new guidance files (for deployment sizing) with external database, remove postgres and postgres-upgrader services from the tshirt size file to be used before deploying.

Installing Black Duck with a file system as read-only

1. ```
   docker swarm init
   ```

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

2.
```
docker stack deploy -c docker-compose.yml -c sizes-gen05/120sph.yaml -c docker-
compose.readonly.yml hub
```

   The resource definition file used above can be changed to better fit your scanning patterns and usage. See the Distribution section for a list of all available options. Please note the system requirements for each option as they increase with greater resource definitions.

   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.

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`.

Verifying the installation

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.

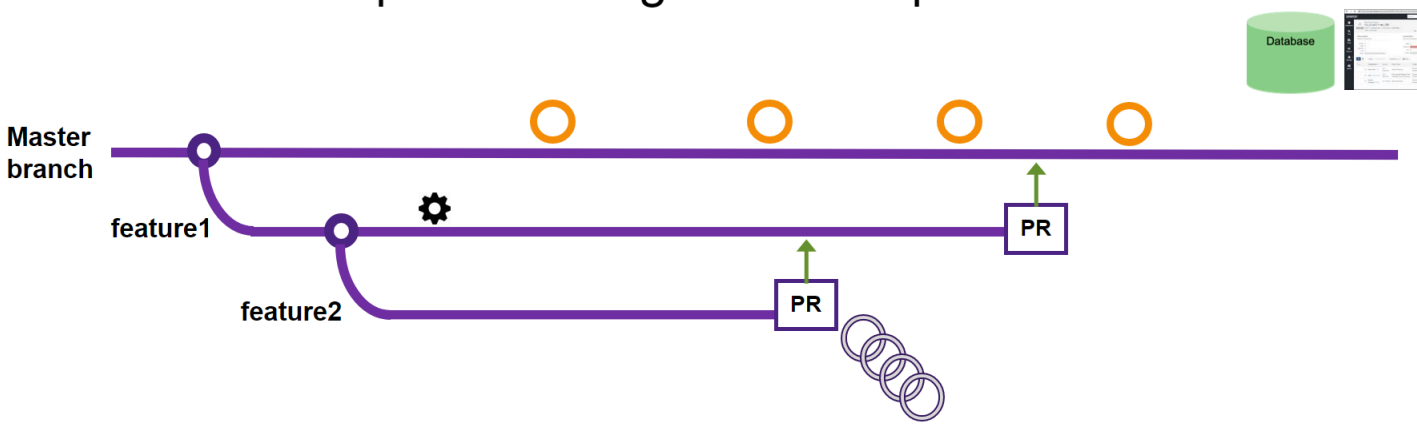
Rapid Scanning in Black Duck

Black Duck Rapid scanning is available by default in Black Duck 2021.6.0.

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.

You must use Black Duck Detect 7.0.0 or later to use Rapid scanning.

Black Duck rapid scanning for development use cases



Rapid Scan
Early check for vulns or policy violations at code commit / pull request

Full SCA Scan
Complete SCA scan to determine all License & Security Risk

- ✓ Scan and do quick check before code comm
- ✓ Pass/fail indicator based on policy violations
- ✓ Visibility into existing vulnerabilities
- ✓ Quick (<1 min) and scalable (>30k scans/da

4. Administrative tasks

Environment files and variables

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.

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
BLACKDUCK_HUB_CORS_ENABLED	Required. Defines whether CORS is enabled; "true" indicates CORS is enabled.
BLACKDUCK_CORS_ALLOWED_ORIGINS_ PROP_NAME	Required. Allowed origins for CORS. The browser sends an origin header when it makes a cross-origin request. This is the origin that must be listed in the blackduck.hub.cors.allowedOrigins/ BLACKDUCK_CORS_ALLOWED_ ORIGINS_PROP_NAME property. For example, if you are running a server that serves a page from http://123.34.5.67:8080, then the browser should set this as the origin, and this value should be added to the property. Note that the protocol, host, and port must match. Use a comma-separated list to specify more than one base origin URL.
BLACKDUCK_CORS_ALLOWED_ HEADERS_PROP_NAME	Optional. Headers that can be used to make the requests.
BLACKDUCK_CORS_EXPOSED_ HEADERS_PROP_NAME	Optional. Headers that can be accessed by the browser requesting CORS.
BLACKDUCK_CORS_ALLOWED_ ORIGIN_PATTERNS_PROP_NAME	Alternative to BLACKDUCK_CORS_ALLOWED_ORIGINS_ PROP_NAME*,* that supports origins declared via wildcard patterns.

Property	Description
	BLACKDUCK_CORS_ALLOWED_ORIGIN_PATTERNS_PROP_NAME overrides BLACKDUCK_CORS_ALLOWED_ORIGINS_PROP_NAME.
BLACKDUCK_CORS_ALLOW_CREDENTIALS_PROP_NAME	Specifies whether the browser should send credentials, such as cookies along with cross domain requests, to the annotated endpoint. The configured value is set on the Access-Control-Allow-Credentials response header of preflight requests. It is invalid to configure ALLOW_CREDENTIALS=true and ALLOWED_ORIGIN=*. If the 'ALL' configuration value is required for allowed origins, it should configured using the ALLOWED_ORIGIN_PATTERNS configuration property instead going forward.

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 100 GB. You can increase this limit by adding the environment variable `BINARY_UPLOAD_MAX_SIZE` to the `docker-compose.local-overrides.yml` file or by setting this globally in `blackduck-config.env` and specifying a value in megabytes.

For example, to increase the maximum binary scan to 10 GB, add the following:

```
webserver:
    environment:
        BINARY_UPLOAD_MAX_SIZE: 10240m
storage:
    environment:
        BINARY_UPLOAD_MAX_SIZE: 10240m
scan:
    environment:
        BINARY_UPLOAD_MAX_SIZE: 10240m
```

After updating the file when deploying the stack, ensure that the `docker-compose.local-overrides.yml` file is included as a command line parameter in the deploy command:

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

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.

Generating and Managing Certificates

Managing certificates is essential for secure communication between Black Duck SCA and its clients. This guide outlines the steps needed to generate and manage required certificates efficiently.

Before You Begin

- Ensure you have OpenSSL installed on your system
- Have administrator access to your Black Duck SCA instance
- Know your organization's certificate requirements

Generate a Certificate in Three Simple Steps

1. **Create Your Certificate Files**

   Run this single command to generate both your private key and certificate signing request:

   ```
   openssl req -new -newkey rsa:2048 -nodes -keyout privateKey.key -out
    certificateSigningRequest.csr
   ```

   You'll be prompted to enter information for your certificate. Complete each field as requested.

2. **Get Your Certificate Signed**

   Choose one option:

   - Option A: Self-signed certificate (for testing environments)

     ```
     openssl x509 -req -days 365 -in certificateSigningRequest.csr -signkey privateKey.key -out
      certificate.crt
     ```

   - Option B: CA-signed certificate (recommended for production) Submit your CSR file to your Certificate Authority and follow their process.

3. **Install Your Certificate**

   a. Place your certificate files in the appropriate directory

   b. Update your Black Duck SCA configuration to reference these files:

   ```
   # Example configuration entry
   hub.webserver.certificate.path=/path/to/certificate.crt
   hub.webserver.privatekey.path=/path/to/privateKey.key
   ```

Verification

Test your certificate installation with:

```
curl -v https://your-blackduck-server.com/api/current-version
```

You should see your certificate details and a successful connection.

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:

   ```
   webserver:
   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
   ```

Troubleshooting

If you encounter the following error, follow the steps below:

`Error response from daemon: rpc error: code = AlreadyExists desc = secret hub_WEBSERVER_CUSTOM_CERT_FILE already exists.`

1. Stop Black Duck.

   ```
   docker stack rm hub
   docker ps : to wait until all containers are down
   ```

2. Remove previous secrets.

```
docker secret rm hub_WEBSERVER_CUSTOM_CERT_FILE
docker secret rm hub_WEBSERVER_CUSTOM_KEY_FILE
```

3. Create secrets again with new valid ones.

```
docker secret create hub_WEBSERVER_CUSTOM_CERT_FILE <certificate file>
docker secret create hub_WEBSERVER_CUSTOM_KEY_FILE <key file>
```

4. Redeploy Black Duck.

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

5. Wait until all containers are healthy including nginx.

Note: If you have already updated your certificate and made changes to the overrides file, ensure you uncomment the secrets portions because the new version of Black Duck will have new files.

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.

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

To connect Black Duck SCA to a secure LDAP server, you must establish a trust connection by importing the LDAP server certificate into the Black Duck SCA LDAP truststore. This is most commonly required when using a self-signed certificate.

Important: If you have configured custom CA certificates using `AUTH_CUSTOM_CA`, `HUB_PROXY_CERT_FILE`, or the `certAuthCACertSecretName` Helm secret, those certificates are imported into the proxy truststore only. They are **not** automatically imported into the LDAP truststore. To establish a trust connection for your LDAP or LDAPS server, you must import the server certificate separately using the Black Duck SCA UI, as described below.

Obtaining your LDAP information

Contact your LDAP administrator and gather the following information before proceeding.

LDAP Server Details

This is the information that Black Duck SCA 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:636`

- (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 User Attributes

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

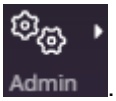
- (required) The absolute base DN under which users can be located.
  Example: `dc=example,dc=com`
- (required) The attribute used to match a specific, unique user. The value of this attribute personalizes the user profile icon with the name of the user.
  Example: `uid={0}`

Test Username and Password

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

Importing the server certificate

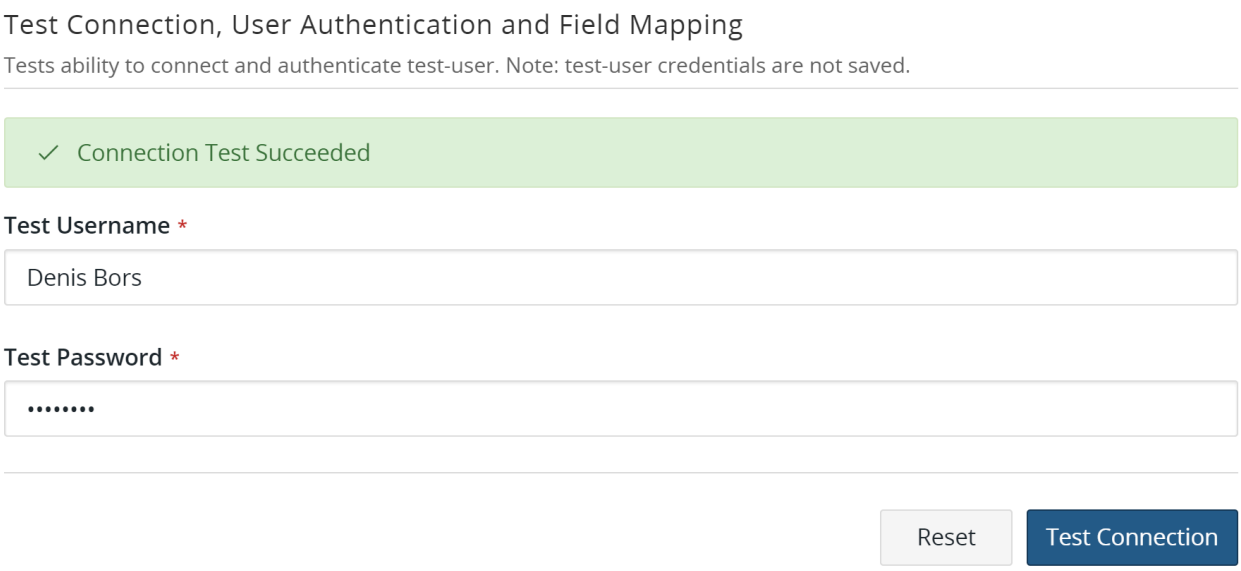
To import the server certificate:

1. Log in to Black Duck SCA as a system administrator.
2. Click 

.
3. Select **Integrations** → **External Authentication**.
4. Click **Lightweight Directory Access Protocol (LDAP)**.
5. Check the **Enable LDAP Configuration** checkbox and complete the information in the **LDAP Server Details** section, as described above. In the **Server URL** field, ensure that the protocol scheme is `ldaps://`.
6. Complete the information in the **LDAP User Attributes** section, as described above.

   Optionally, clear the **Create user accounts automatically in Black Duck SCA** check box to turn off the automatic creation of users when they authenticate with LDAP. This check box is selected by default so that users who do not exist in Black Duck SCA are created automatically when they log in using LDAP. This applies to new installs and upgrades.

7. Enter the user credentials in the **Test Connection, User Authentication and Field Mapping** section and click **Test Connection**.
8. If there are no issues with the certificate, it is automatically imported and the "Connection Test Succeeded" message appears:



9. If there is an issue with the certificate, a dialog box listing details about the certificate will appear. Do one of the following:
   - Click **Cancel** to fix the certificate issues. Once fixed, retest the connection to verify that the issues have been resolved and the certificate has been imported. If successful, the "Connection Test Succeeded" message appears.
   - Click **Save** to import the certificate as-is. 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 SCA 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.

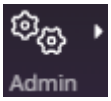
Downloading log files and heatmap data

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 or the heatmap data for your system.

Please note that it may take a few minutes to prepare the log or heatmap files. Refer to the installation guide for more information on obtaining logs and configuring heatmap data.

Accessing log files

To download the log files from the Black Duck UI:

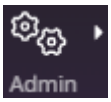
1. Log in to Black Duck with the System Administrator role.
2. Click 

→ **Jobs**.
3. Select the **System Information** tab.
4. Click either **Logs for Last 2 Days (.zip)** or **Logs for Last 14 Days (.zip)**.

Accessing heatmap data

You can review and analyze terminal scan trends by downloading the heatmap as a compressed CSV and create the heatmap as a pivot in a spreadsheet program.

To download the heatmap data for your system:

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

→ **Jobs**.
3. Select the **System Information** tab.
4. Click **Download Heatmap (.zip)** in the Heatmap data section.

Viewing 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

Be 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 memory limits

The following values may be changed if more memory is given to the container when you modify the HUB_MAX_MEMORY value.

- org.quartz.scheduler.periodic.maxthreads: 4
- org.quartz.scheduler.periodic.prefetch: 2
- org.quartz.scheduler.ondemand.maxthreads: 8
- org.quartz.scheduler.ondemand.prefetch: 4

Examples for guidance

```
SMALL
HUB_MAX_MEMORY: 4096m
org.quartz.scheduler.periodic.maxthreads: 4
org.quartz.scheduler.periodic.prefetch: 2
org.quartz.scheduler.ondemand.maxthreads: 8
org.quartz.scheduler.ondemand.prefetch: 4
```

```
Medium
HUB_MAX_MEMORY: 6144m
org.quartz.scheduler.periodic.maxthreads: 4
org.quartz.scheduler.periodic.prefetch: 2
org.quartz.scheduler.ondemand.maxthreads: 12
org.quartz.scheduler.ondemand.prefetch: 8
```

```
Large/X-Large
HUB_MAX_MEMORY: 12288m
org.quartz.scheduler.periodic.maxthreads: 8
org.quartz.scheduler.periodic.prefetch: 4
org.quartz.scheduler.ondemand.maxthreads: 24
org.quartz.scheduler.ondemand.prefetch: 16
```

Note: Large and X-Large differ only in the number of jobrunner instances. The memory and thread counts are the same.

Note: The number of max threads for both pools should not exceed 32

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: Black Duck 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 30 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 30 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 `VersionBomEventClearnupJob` 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 `.yml` file, use the `docker-compose.local-overrides.yml`, 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 `.yml` file after each Black Duck upgrade. T

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

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

Configuring analytics in Black Duck

In Black Duck you can disable phone home globally for Black Duck 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 SCA supports using an external PostgreSQL instance. Please refer to PostgreSQL Version Upgrade Schedule for more information regarding supported PostgreSQL versions in Black Duck SCA.

Note: If you are installing using Azure PostgreSQL, a database administrator will need to enable installation of the hstore PostgreSQL extension before installing or upgrading to 2023.4.0 or later. Additionally, ensure that the `azure.extensions` value includes both `pgcrypto` and `hstore` to allow init scripts to access the Azure database.

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 `POSTGRESQL_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.

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 POSTGRESQL_USER with the numeric user names.

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.pgsql` 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.pgsql 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.pgsql` 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 user username and password authentication.

Parameter	Description
HUB_POSTGRES_ENABLE_SSL	Defines the use of SSL in database connections. • 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. • 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:

      1. If necessary, remove the comment character (#) before the name of the service.
      2. 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.

Using mTLS with Docker Swarm for External Databases

This section provides guidance for configuring mTLS (mutual Transport Layer Security) with PostgreSQL in a Docker Swarm environment.

Configure PostgreSQL

Customers are responsible for configuring PostgreSQL to use mTLS. Follow these steps to ensure proper setup:

Database Setup and configuration:

1. Set up the database following the guidelines outlined in the Black Duck Documentation Portal.
2. **Update `hub-webserver.env`:**

   Configure the `PUBLIC_HUB_WEBSERVER_HOST` variable in the `hub-webserver.env` file as documented in other sections.

3. **Update `hub-postgres.env`:**

   Modify the `hub-postgres.env` file with the following values:

   - `HUB_POSTGRES_ENABLE_SSL` - Set to `true`
   - `HUB_POSTGRES_ENABLE_SSL_CERT_AUTH` - Set to `true`
   - `HUB_POSTGRES_HOST` - Enter your PostgreSQL host.
   - `HUB_POSTGRES_PORT` - Enter your PostgreSQL port.
   - `HUB_POSTGRES_USER` - If not using the default, specify the non-admin PostgreSQL user for Black Duck.
   - `HUB_POSTGRES_ADMIN` - If not using the default, specify the admin PostgreSQL user for Black Duck.

Certificates and Keys

To utilize mTLS, you must have the following certificates and keys:

- **Certificates Required:**

  You need either:

  1. A root certificate, or
  2. A client certificate and key, or
  3. An admin certificate and key, or a combination of these.

Note: If using a client certificate, the client key is also required. The same applies for the admin certificate. If either the certificate or key is missing, the startup script will not function properly. The root certificate is standalone and does not require a key.

- **File Ownership and Permissions:**

  All certificates and keys must be stored in the same directory, which should be owned by root and have permissions set to 755. The following naming and permission conventions must be followed:

  - `HUB_POSTGRES_CA` (Root Certificate): Owned by root, permissions 644.
  - `HUB_POSTGRES_CRT` (Client Certificate): Owned by root, permissions 644.
  - `HUB_POSTGRES_KEY` (Client Key): Owned by root, permissions 640. Must be in PKCS#8 format.
  - `HUB_ADMIN_POSTGRES_CRT` (Admin Certificate): Owned by root, permissions 644.
  - `HUB_ADMIN_POSTGRES_KEY` (Admin Key): Owned by root, permissions 640. Must be in PKCS#8 format.

Volume Mount

A new YAML file has been introduced specifically for using mTLS with an external database: `docker-compose.externaldb-cert-volume.yml`. Incorporating this YAML into your deployment will enable mTLS and create a volume mount for the certificates and keys.

Required Edits to YAML File:

1. **HUB_POSTGRES_SSL_MODE:**

   This setting configures the `sslmode` for authenticating with PostgreSQL. Supported modes are `verify-ca` (default) and `verify-full`. Using any other setting will disable mTLS authentication.

2. **Integrations Container:**

   If deploying with the integrations container, uncomment the relevant lines that are currently commented out, starting with `#integration:`.

3. **Volume Path:**

   Set the path for `volumes#hub-certificates#driver_opts#device` to the absolute path of the directory containing the certificates and keys.

Starting Black Duck

When starting Black Duck, ensure to include `docker-compose.externaldb-cert-volume.yml` in your command to enable mTLS functionality.

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
- Integration

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 if authentication is leveraged via proxy:

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

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, Match engine, Bom engine, 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, integration, and scan. This example is for the scan service:

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

Using an authenticated proxy

Due to changes made in JDK 8u111 (Consolidated JDK 8 Release Notes (oracle.com)), customers using a proxy requiring basic authentication may face issues communicating with the Black Duck registration services. To overcome this, the following change should be made to `blackduck-config.env` (Docker Swarm) or the ConfigMap (Kubernetes/Openshift):

```
REGISTRATION_SERVICE_OPTS="-Djdk.http.auth.tunneling.disabledSchemes=''"
```

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: Black Duck 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 SCA's SAML implementation provides single sign-on (SSO) functionality, enabling Black Duck SCA users to be automatically signed-in to Black Duck SCA when SAML is enabled. Enabling SAML applies to all your Black Duck SCA users and cannot be selectively applied to individual users.

Tip: All hosted customers should secure access to their Black Duck SCA application by leveraging out-of-the-box support for single sign-on (SSO) via SAML or LDAP. For information on enabling and configuring these security features, see the installation guides. Customers using a SAML SSO provider that offers two-factor authentication are also encouraged to enable that technology to further secure access to their Black Duck SCA application. For more information, see Configuring SAML for Single Sign-On.

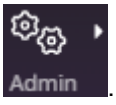
Note the following:

- It is not possible to configure both SAML and LDAP at the same time.
- To enable or disable SAML functionality, you must be a user with the system administrator role.
- Black Duck SCA is able to synchronize and obtain an external user's information (Name, FirstName, LastName and Email) if the information is provided in attribute statements. Note that the first and last name values are case-sensitive.

  Black Duck SCA is also able to synchronize an external user's group information if you enable group synchronization in Black Duck SCA.
- When logging in with SAML enabled, you are re-directed to your identity provider's login page, not Black Duck SCA's login page.
- When SSO users log out of Black Duck SCA, a logout page now appears notifying them that they successfully logged out of Black Duck SCA. This logout page includes a link to log back into Black Duck SCA; users may not need to provide their credentials to successfully log back in to Black Duck SCA.
- If there are issues with the SSO system and you need to disable the SSO configuration, you can enter the following URL to log in to Black Duck SCA:

  ```
  <Black Duck servername>/ui/login
  ```

Enabling or disabling single sign-on using SAML

1. Click 

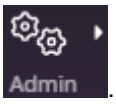
.

2. Select **Integrations** → **External Authentication**.
3. Click **Security Assertion Markup Language (SAML)**, complete the following:
   a. Select the **Enable SAML configuration** check box.
   b. **Service Provider Entity ID** field: Enter the information for the Black Duck SCA server in your environment in the format **https://*host*** where *host* is your Black Duck SCA server.
   c. Select one of the following **Identity Provider Metadata**:
      - **URL** and enter the URL for your identity provider.
      - **XML File** and either drop the file or click in the area shown to open a dialog box from which you can select the XML file.
   d. **Service Provider Entity ID** field. Enter the information for the Black Duck SCA server in your environment in the format **https://*host*** where *host* is your Black Duck SCA server.
   e. **External Black Duck SCA URL** field. The URL of the public URL of the Black Duck SCA server.

      For example: https://blackduck-docker01.dc1.lan
   f. Optionally, select any of the following:
      - **Send Signed Authentication Request**: If this option is enabled, it indicates the asserting party's preference that relying parties should sign the authentication request before sending.
      - **Enable Group Synchronization**: If this option is enabled, upon login, groups from the Identity Provider (IDP) are created in Black Duck SCA 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'.
      - **Enable Local Logout Support**: If this option is enabled, after logging out of Black Duck SCA, the IDP's login page would appear. When local logout support is enabled, SAML requests are sent with ForceAuthn="true". Check with the IDP to confirm that this is supported.
      - **Create user accounts automatically in Black Duck SCA**: If a user logs in using the IdP and the user doesn't exist in Black Duck, we create a local user in Black Duck SCA's database if that option is selected.
4. Click **Save**.

After clicking **Save**, the **Black Duck SCA 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 

.
2. Select **Integrations** → **External Authentication**.
3. Click **Security Assertion Markup Language (SAML)**
4. Clear the **Enable SAML configuration** check box.
5. Click **Save**.

Additional information

- Assertion Consumer Service (ACS): **https://<host>/saml/SSO**

- Recommended Service Provider Entity ID: **https://<host>** where *host* is your Black Duck SCA server location.

Signed Authentication Certificates

Black Duck SCA supports SAML signed authentication certificates, enhancing security by validating authentication requests against a trusted certificate.

To create a signed authentication certificate:

1. Click **+ Certificate**.
2. Enter or select an **Expiration Date** in the **Create SAML Certificate** modal.
3. Click **Create**.

Note: A maximum of two certificates can be created.

To activate, download, or delete a signed authentication certificate:

1. Click ••• for the desired certificate.
2. Select from the following options:
   - **Activate**. Specify this certificate to be used to validate SAML requests. Once activated, the certificate will be marked with the Active.
   - **Download**. Download the certificate as a CERT file.
   - **Delete**. Delete the certificate from the list of signed authentication certificates. Active certificates cannot be deleted.

You will be notified when a signed authentication certificate is nearing its expiration date. Notifications include a banner at the top of the screen, a pop up message, as well as an Expiring Soon icon on the certificate itself 30 days prior to the expiry date. Notifications are triggered well in advance to provide time for system administrators to update or replace the certificate.

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.
   b. The administrator optionally configures secrets encryption. Source code uploads for hosted customers are always encrypted.

2. 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 files are then encrypted. 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.

Enabling file upload

To enable file upload, set the `ENABLE_SOURCE_UPLOADS` environment variable In the `blackduck-config.env` file located in the `docker-swarm` directory, to true:

```
ENABLE_SOURCE_UPLOADS=true
```

Enabling encryption of source code

To enable encryption of source code uploaded along with other sensitive data managed by the storage service `BLACKDUCK_CRYPTO_ENABLED` must be set. See Configuring secrets encryption in Black Duck for more information on secrets encryption.

Note: Source code uploads for hosted customers are always encrypted.

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.

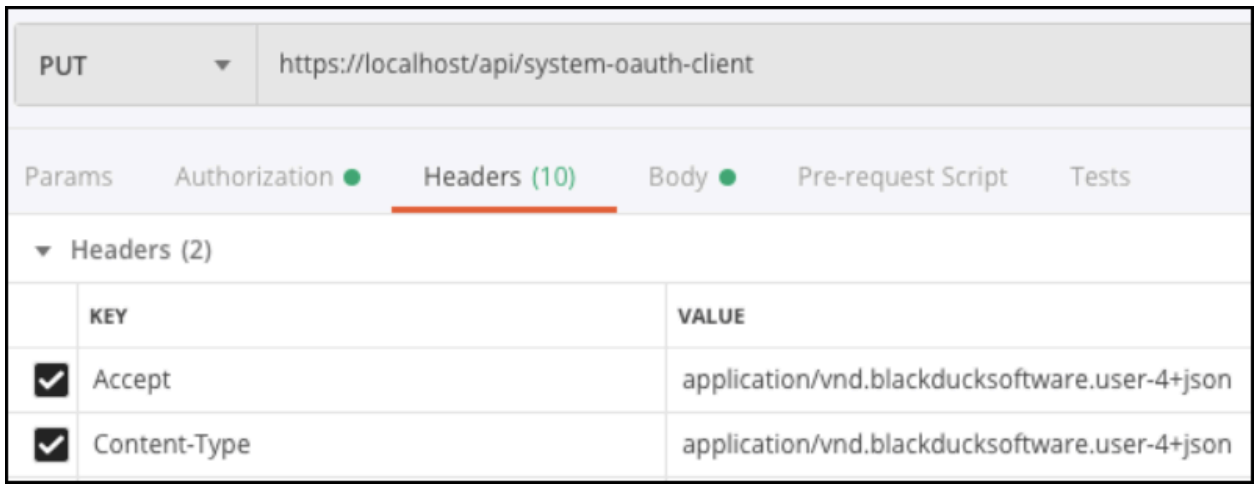
   The value that you type in the PUT request body is the new timeout value.

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

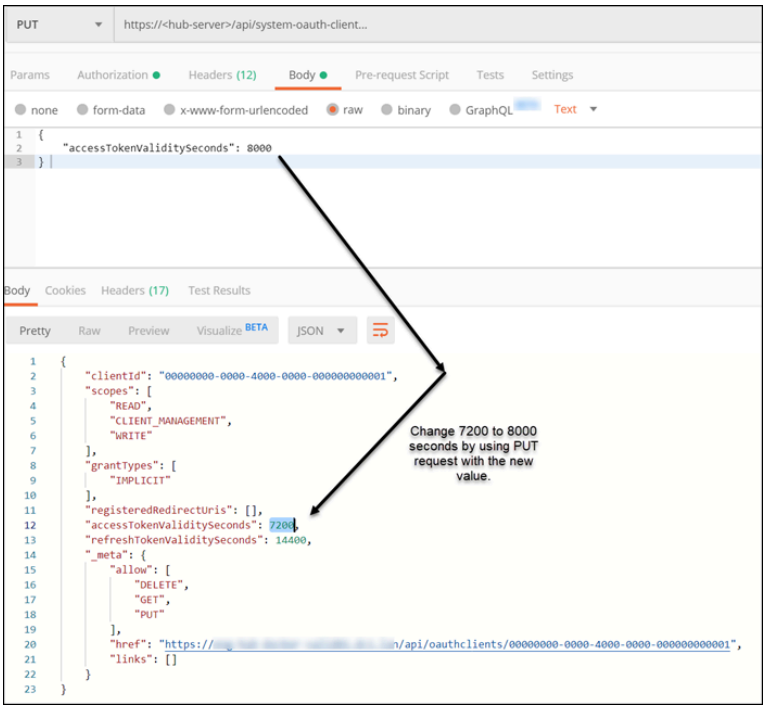
The following media types are accepted:

```
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 SCA, 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 **User Administrator** 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 2026.7.2 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, Black Duck 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
- BLACKDUCK_HUB_SHOW_UNMATCHED: Determines whether the unmatched component count is shown. The default is false (not shown).

Match Types for bdio2 uploaded jsonld/bdio file

The default usage for these matches is Dynamically Linked.

- Direct Dependency Binary

- Transitive Dependency Binary

These are full-file matches same as the existing binary matches.

A single file can have multiple Transitive Dependency Binary matches, but only one Direct Dependency Binary.

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): 1001
- 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
- Redis (blackduck-redis): run as any user/group
- Bomengine (blackduck-bomengine): 100
- Matchengine (blackduck-matchengine: 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 1001.
  - 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.

Create BOM reports using UTF8 character encoding

To enable support for UTF8 character encoding in BOM reports when using non-Western characters, add the following to the `blackduck-config.env` file:

```
USE_CSV_BOM=true
```

Scan monitoring

The default period where information is gathered for the scan monitoring endpoint is 1 hour. This can be configured to a different value by using the following property:

`HUB_SCAN_MONITOR_ROLLUP_WINDOW_SECONDS`

The API will only be able to provide information if the system has at least a specific number of scans to analyse in the last time period configured above (default period is 1 hour). If the scans are less than the threshold value, a level 1 request will return a **OK** response by default. The number of scans can be configured to a different value by using the following property:

`HUB_SCAN_MONITOR_MINIMUM_SCAN_COUNT`

A level 1 request returns a response of "OK" or "NOT OK" based on the failure rate thresholds, where a result greater than the upper limit will return a "NOT OK" response. The upper (default 30%) and lower (default 10%) limits are set as a percentage and can be configured with the following properties:

`HUB_SCAN_MONITOR_ERROR_RATE_LOWER_THRESHOLD_PERCENT`

`HUB_SCAN_MONITOR_ERROR_RATE_HIGHER_THRESHOLD_PERCENT`

For more information on the scan monitoring API request, please refer to the REST API Developers Guide in Black Duck.

Configuring HTML report download size

You can configure your environment to allow for larger or smaller HTML report downloads. The default size is 3000 KB. Reports exceeding this limit will return a 503 Service Unavailable error message.

To change this limit, change the value of the `HUB_MAX_HTML_REPORT_SIZE_KB` property in your `blackduck-config.env` file.

Configuring KB license update and security update jobs

To disable the KB license update and security update jobs, add the following property in your `blackduck-config.env` file:

```
KB_UPDATE_JOB_ENABLED=FALSE
```

To change the frequency of the KB license update job, add the following property in your `blackduck-config.env` file:

```
KB_LICENSE_UPDATER_PERIOD_MINUTES=<time in minutes>
```

Configuring secrets encryption in Black Duck

Black Duck supports encryption at rest of critical data within the system. This encryption is based upon a secret provisioned to the Black Duck installation by the orchestration environment (Docker Swarm or Kubernetes). The process to create and manage this secret, create a backup secret, and rotate the secret based upon your own organization's security policies is outlined below.

The critical data being encrypted are the following:

- SCM Integration OAuth tokens
- SCM Integration provider OAuth application client secrets
- LDAP credentials
- SAML private signing certificates
- RSA keys

Note: Once secrets encryption is enabled, it can never be disabled.

What is an encryption secret?

An encryption secret is a random sequence used to generate an internal cryptographic key in order to unlock resources within the system. The encryption of secrets in Black Duck is controlled by 3 symmetric keys, the root, backup and previous keys. These three keys are generated by seeds passed into Black Duck as Kubernetes and Docker Swarm secrets. The three secrets are named:

- `crypto-root-seed`
- `crypto-backup-seed`
- `crypto-prev-seed`

In normal conditions, all three seeds will not be in active use. Unless a rotation action is in progress, the only seed active will be the root seed.

Securing the root seed

It is important to protect the root seed. A user possessing your root seed along with a copy of the system data could unlock and read the protected contents of the system. Some Docker Swarm or Kubernetes systems do not encrypt their secrets at rest by default. It is strongly advised to configure these orchestration systems to be encrypted internally so that secrets created afterwards in the system remain secure.

The root seed is necessary to recreate the system state from backup as part of a disaster recovery plan. A copy of the root seed file should be stored in a secret location separate from the orchestration system so that the combination of the seed plus the backup can recreate the system. Storing the root seed in the same location as the backup files is not advised. If one set of files is leaked or stolen – both will be, therefore, having separate locations for backup data and seed backups is recommended.

Enabling secrets encryption in Docker Swarm

To enable secrets encryption in Docker Swarm:

1. First create the initial seed secrets
2. Edit `blackduck-config.env` to set the `BLACKDUCK_CRYPTO_ENABLED` to `true`
3. Start using `docker-compose.encryption.yml` as part of the Docker Swarm deployment

The Docker Swarm deployment command would look something like:

```
docker stack deploy -c docker-compose.yml [-c <other compose yaml files> …] -c docker-
compose.encryption.yml blackduck
```

Key seed administration scripts

You can find sample administration scripts in the Black Duck GitHub public repository:

https://github.com/blackducksoftware/secrets-encryption-scripts

These scripts are not meant to be used for administering Black Duck secrets encryption, but rather to illustrate the use of the low-level Docker and Kubernetes commands documented here. There are two sets of scripts, each in its own sub-directory, corresponding to use on Kubernetes and Docker Swarm platforms. There is a one-to-one correspondence between the individual scripts, where applicable, for Kubernetes and Docker Swarm. For example, both sets of scripts contain a script called:

`createInitialSeeds.sh`

Generating seeds

Generating seeds in OpenSSL

The content of the seeds can be generated using any mechanism that generates secure random contents at least 1024 bytes long. As soon as a seed has been created and saved in a secret, it should be removed from your file system and saved in a private, secure location.

The OpenSSL command is as follows:

```
openssl rand -hex 1024 > root_seed
```

Generating seeds in Docker Swarm

In Docker Swarm, Black Duck must be stopped in order to create and delete secrets. The Docker Swarm command is as follows:

```
docker secret create crypto-root-seed ./root_seed
```

In Docker Swarm, a secret configured in the orchestration files must exist and it cannot be zero length. To work around this restriction, Black Duck treats "placeholder" encryption seeds of 2 or less bytes as if they do not exist. As such, the previous key secret can be deleted in Docker Swarm with the following command:

```
echo -n "1" | docker secret create crypto-prev-seed –
```

Once cryptography is enabled via the orchestration files, the three seed secrets must be created with commands analogous to those shown here, before enabling Black Duck secrets encryption; see sample scripts.

Configuring a backup seed

Having a backup root seed is recommended to ensure the system can be recovered in a disaster recovery scenario. The backup root seed is an alternative root seed that can be put in place in order to recover a system. Consequently, it must be stored securely in the same way as a root seed.

The backup root seed has some special features in that once it is associated with the system, it remains viable even across root seed rotations. Once a backup seed is processed by the system, it should be removed from the secrets to limit its exposure to attacks and leakage. The backup root seed may have a different (less often) rotation schedule as the secret should not be “active” in the system at any point in time.

When you need or want to rotate a root seed, you first need to define the current root seed as the previous root seed. You can then generate a new root seed and put that in place.

When the system processes these seeds, the previous root key will be used to rotate resources to use the new root seed. After this processing, the previous root seed should be removed from the secrets to complete the rotation and clean up the old resources.

Creating a backup root seed

Once created initially, the backup seed/key wraps the TDEK (tenant decrypt, encrypt key) low-level key. The sample script `createInitialSeeds.sh` will create both a root and a backup seed. Once Black Duck is running, it uses both keys to wrap the TDEK.

After that operation is complete and both the root and backup seeds are securely stored elsewhere, the backup seed secret should be deleted; see sample script `cleanupBackupSeed.sh`.

If the root key is lost or leaked, the backup key can be used to replace the root key; see sample script `useRootSeed.sh`.

Rotating the backup seed

Similarly to the root key, the backup seed should be rotated periodically. Unlike for the root seed, where the old root seed is stored as a previous seed secret and a new root seed secret presented to the system, the backup seed is rotated just by creating a new backup seed. See the sample script `rotateBackupSeed.sh`.

After the rotation is complete, the new backup seed should be stored securely and removed from the Black Duck host file system.

Managing secret rotation

It is good practice to rotate the root seed in use on a periodic basis according to your organization’s security policy. In order to do this, an additional secret is necessary to perform the rotation. To rotate the root seed, the current root seed is configured as the “previous root seed”, and a newly generated root seed is

generated and configured as the root seed. Once the system processes this configuration (specifics below), the secrets will have been rotated.

At that point in time both the old and the new seeds are able to unlock the system contents. By default, the new root seed will be used, allowing you to test and make sure the system is working as intended. Once everything has been verified, you complete the rotation by removing the “previous root seed”.

Once the previous root seed is removed from the system it can no longer be used to unlock the contents of the system and can be discarded. The new root seed is now the proper root seed which should be backed up and secured appropriately.

The root key is used to wrap the low-level TDEKs (tenant decrypt, encrypt key) that actually encrypt and decrypt Black Duck secrets. Periodically, at times convenient for Black Duck administrators and conforming to user organization rules, the root key should be rotated.

The procedure to rotate the root key would be create a previous seed secret with the contents of current root seed. Then a new root seed should be created and stored in the root seed secret.

Secret rotation in Docker Swarm

For Docker Swarm, Black Duck must be stopped, the three seed operations performed, and then Black Duck started up again. The root seed extracted from the running Black Duck instance as the previous seed, `extractRootAsPreviousSeed.sh`. See the Docker Swarm sample script `rotateRootSeed.sh`.

After the rotation completes the previous seed secret should be removed; see sample script `cleanupPreviousSeed.sh`. In Docker Swarm, the system must be brought down, the previous key removed and then Black Duck started up again.

The state of the rotation can be tracked by looking at crypto diagnostics tab, in the user interface by going to Admin > System Information > crypto.

Configuring custom volumes for Blackduck Storage

The storage container may be configured to use multiple volumes, up to three (3), for the storage of file based objects such as reports. In addition, the configuration can be set up to migrate objects from one volume to another.

Configuring multiple volumes

The swarm deployment files contain a file named `docker-compose.storage-overrides.yml`. This file is a template that must be customized and included in the deployment in order to override the default storage configuration to use more volumes.

This file has three sections: Environment, Storage Volumes and Service Volumes.

Why more than one volume?

By default, the storage container uses a single volume to store all objects. This volume is sized based on typical customer usage for stored objects. As each customer is different, it may become necessary to have more space available than the volume can provide. Since not all volumes are expandable, it may become necessary to add a different, larger volume and migrate the data to the new volume.

Another reason why multiple volumes may become necessary is if the volume is hosted on a remote system (NAS or SAN) and that remote system is due to be decommissioned. A second volume hosted on a new system would need to be created and the content moved to it.

Environment

The environment section is structured as follows:

```
services:
                storage:
                environment:
                # Provider 1 settings
                BLACKDUCK_STORAGE_PROVIDER_ENABLED_1: 'true'
                BLACKDUCK_STORAGE_PROVIDER_PREFERENCE_1: 10
                BLACKDUCK_STORAGE_PROVIDER_READONLY_1: 'false'
                BLACKDUCK_STORAGE_PROVIDER_MIGRATION_MODE_1: 'none'
                # Provider 2 settings
                BLACKDUCK_STORAGE_PROVIDER_ENABLED_2: 'false'
                BLACKDUCK_STORAGE_PROVIDER_PREFERENCE_2: 20
                BLACKDUCK_STORAGE_PROVIDER_READONLY_2: 'false'
                BLACKDUCK_STORAGE_PROVIDER_MIGRATION_MODE_2: 'none'
                # Provider 3 settings
                BLACKDUCK_STORAGE_PROVIDER_ENABLED_3: 'false'
                BLACKDUCK_STORAGE_PROVIDER_PREFERENCE_3: 30
                BLACKDUCK_STORAGE_PROVIDER_READONLY_3: 'false'
                BLACKDUCK_STORAGE_PROVIDER_MIGRATION_MODE_3: 'none'
                ...
```

These settings enable or disable each of the three storage providers. The configuration in the file is shipped with the same configuration as the default. (provider 1 enabled, 2 and 3 disabled).

To configure a provider, change the settings for the given provider. The settings are:

Setting	Details
`BLACKDUCK_STORAGE_PROVIDER_ENABLED_(N)`	**Default**: for provider 1, `true`, for others `false` **Valid values**: `true` or `false` **Notes**: This enables or disables a provider. AT LEAST ONE PROVIDER MUST BE ENABLED OR THE SYSTEM CANNOT START.
`BLACKDUCK_STORAGE_PROVIDER_PREFERENCE_(N)`	**Default**: provider number times 10 **Valid values**: `0-999` **Notes**: This value indicates the preference between the providers. The provider with the highest priority (lowest preference number) will be used to store new content. The other providers in the configuration will continue to serve their content and all other functions - but will not have new content added to them. NOTE: All active providers must have UNIQUE preference numbers. Two providers cannot share the same value.
`BLACKDUCK_STORAGE_PROVIDER_READONLY_(N)`	**Default**: `false` **Valid values**: `true` or `false` **Notes**: This indicates a provider is read-only. The highest priority (lowest preference number) provider cannot be read-only or the system cannot function. A read only provider will not have the storage volume altered by addition of data or removal of data, however metadata in the database will be manipulated to record object deletions and other changes.

Setting	Details
BLACKDUCK_STORAGE_PROVIDER_MIGRATION_MODE_(N)	**Default**: none **Valid values**: none, drain, delete, duplicate **Notes**: This configures the migration mode for the provider, Details of what this mode is and how to use it are provided in the migration section of this document.

Storage volumes

This section assigns named storage volumes with specific mount points on the system where the software expects them to be. The volume for provider one is named "storage-volume" and is defined elsewhere. It will always exist, even if storage provider 1 is not enabled. The storage volume section of the overrides file is as follows:

```
services:
                storage:
                environment:
                ...
                volumes:
                ## NOTE: File provider 1's volume mount point is always
                ## present since it is defined as the upstream default
                ## It will only be used if the file provider 1 is enabled
                ##
                ## File provider 2's volume mount point
                #- storage-volume2:/opt/blackduck/hub/uploads2
                ##
                ## File provider 3's volume mount point
                #- storage-volume3:/opt/blackduck/hub/uploads3
                ...
```

For the other providers, you should merely uncomment the sections out for each provider you have enabled. For example if provider two (2) is enabled, this section of the file should look like:

```
...
                ##
                ## File provider 2's volume mount point
                - storage-volume2:/opt/blackduck/hub/uploads2
                ##
                ...
```

Service volumes

This section defines the volumes by name and their configuration. The configuration provided uses Docker Swarm's default volume driver. Similar to the Service Volume section, you only need to uncomment the appropriate pieces based on which providers are enabled. The service volume section of the file looks like this:

```
services:
                storage:
                environment:
                ...
                volumes:
                ...
                volumes: {
                ## NOTE: File provider 1's storage volume is always present
                ## since it is defined as the upstream default. It will
                ## only be used if the file provider 1 is enabled
                ##
                ## File provider 2's storage volume
                #storage-volume2: null,
                ##
                ## File provider 3's storage volume
```

```
                #storage-volume3: null,
                }
```

Note: Configurations other than the default storage driver are possible here. A storage volume named "storage-volume2" could be created backed by NFS, a NAS or a SAN. These configurations would need to be worked out based on standard docker swarm params in conjunction with the vendors settings or extensions as appropriate to the customer's environment, and so cannot be documented here.

Migrating between volumes

With multiple volumes configured, it is possible to migrate content from one or more provider volumes to a new provider volume. This can only be done for providers that are not the highest priority (lowest preference). To do this, configure the volumes with one of the following migration modes. Once configured, Black Duck needs to be restarted in order to initiate the migration which is performed by a job in the background until it is completed.

Migration Mode	Details
`none`	**Purpose**: To indicate no migration is in progress. **Notes**: The default migration mode.
`drain`	**Purpose**: This mode moves content from the configured provider to the highest priority (lowest preference number) provider. Once content is moved, it is immediately removed from the source provider. **Notes**: This is a straight move operation - adding it to the target provider and removing it from the source.
`delete`	**Purpose**: This mode copies content from the configured provider to the highest priority (lowest preference number) provider. Once content is copied, it is marked for deletion in the source provider. The standard deletion retention periods apply - after that period the content is removed. **Notes**: This is a move that allows for the ability for the system to be recovered from backup within the retention window so that content in the source provider remains viable. The default retention period is 6 hours.
`duplicate`	**Purpose**: This mode copies content from the configured provider to the highest priority (lowest preference number) provider. Once content is copied, the source is left unaltered, including the metadata. **Notes**: After the duplicate migration, you will have two volumes with all of the content and the metadata in the database. If you take the next step in the “duplicate and dump” process and unconfigure the original volume, the files will be deleted but the metadata will remain in the database - referencing an unknown volume generating a warning in the pruner jobs (a job error). To resolve the error, use

Migration Mode	Details
	the following property to enable the pruning of the orphaned metadata records: `storage.pruner.orphaned.data.pruning.enable=true`

Configuring storage volumes for reports

You can configure report properties for a storage volume in your system properties, such as `blackduck-config.env` with the following properties:

- `blackduck.hub.maximum.report.age`: Sets the maximum age of reports which are stored in the Black Duck persistent storage volume. Reports older than this value (in days) are deleted. Default value is 180 days.

  The maximum suggested value for this property is 365 days.

- `blackduck.hub.maximum.reports.per.user`: Sets the maximum number of reports each user can generate in Black Duck. When exceeding this value, reports generated by this user will start to be deleted starting from the oldest reports generated by this user. Defaut value is 100 reports.

  The maximum suggested value for this property is 1000 reports. This value assumes less than 100 users generating large numbers of reports. Exceeding 100,000 total reports is not recommended.

Note: Black Duck estimates that 25,000 reports in Black Duck requires roughly 3GB of additional disk storage and scales linearly as more reports are added. We recommend administrators monitor disk space usage to ensure adequate growth capacity.

Warning: Long term storage of 100,000 or more reports in Black Duck is not recommended. External data storage outside of the Black Duck system is recommended for long term arrival or auditing storage. Exceeding this recommendation can lead to performance degradation for report generation, APIs and general Black Duck database performance.

Configuring jobrunner thread pools

In Black Duck, there are two job pools - one that runs scheduled jobs (called the periodic pool) and one that runs jobs that are initiated from some event, including API or user interactions (called the ondemand pool).

Each pool has two settings: max threads, and prefetch.

Max threads is the maximum number of jobs a jobrunner container can run at the same time. Adding together periodic and ondemand max threads should never be larger than 32 as most jobs use the database and there are at most 32 connections. It is very easy to saturate the jobrunner memory, so the default thread counts are set very low.

Prefetch is the number of jobs each jobrunner container with grab in each round trip to the database. Setting this higher is more efficient, but setting it lower will spread the load more evenly across multiple jobrunners (although even load is not a design goal of jobrunner in general).

In Docker Swarm, add the following settings to your `blackduck-config.env` file:

```
org.quartz.scheduler.periodic.maxthreads=2
org.quartz.scheduler.periodic.prefetch=1
org.quartz.scheduler.ondemand.maxthreads=4
org.quartz.scheduler.ondemand.prefetch=2
```

Configuring Rapid Scan for BOM support

You can configure Rapid Scan to provide a full results format to include data points for BOM support. To do so, set the following environment variable: `BLACKDUCK_RAPID_SCAN_EXTENDED_DATA=true`.

Changing the long running job threshold

You can configure the threshold to determine long running jobs by adding the following variable to your `blackduck-config.env` file:

- BLACKDUCK_DEFAULT_JOB_RUNTIME_THRESHOLD_HOURS={value in hours}

The default value for this environment variable is 24 hours.

Enabling SCM Integration

This feature is not enabled by default in Black Duck and must be activated by adding the feature to your Product Registration key.

To enable SCM integration, you must deploy `docker-compose.integration.yml`, no further configuration is required. The following environment variable will be automatically added:

```
webserver:
  environment: {ENABLE_INTEGRATION_SERVICE: "true"}
```

Configuring automatic scan retry header

The automatic scan retry header can help ensure that scans are successfully completed even if there are temporary issues such as network instability, service interruptions, or other issues that may interfere with scans. By automatically retrying scans using a queuing system, the system can reduce the need for manual intervention. Configuring the automatic scan retry header allows Administrators to tailor the retry behavior to their specific needs and environment. This customization helps optimize the retry process, ensuring that it aligns with the system's performance and operational requirements.

You can modify the Retry-After header found in the 429 response for Detect to know when to attempt a rate limited scan again by adding the following properties in the `blackduck-config.env` file:

- `BLACKDUCK_USE_QUEUE_RATE_LIMITING`: Set to `true` to enable queue base rate limiting in your environment. Default value is `false`.
- `BLACKDUCK_INITIAL_RATE_LIMIT_DURATION_BRACKET_THRESHOLD_MINUTES`: Dictates the duration the system must be rate limiting before the system moves from the first retry duration to the second retry duration. Default value is 2 minutes.
- `BLACKDUCK_RATE_LIMIT_DURATION_THRESHOLD_BRACKET_INCREMENT_MINUTES`: Dictates the duration that the system will stay in a rate limiting bracket before going to the next retry duration and multiplier. Default value is 2 minutes.
- `BLACKDUCK_INITIAL_RETRY_DURATION_MINUTES`: The initial duration of the Retry-After header in the first rate limiting bracket. Default value is 1 minute.
- `BLACKDUCK_RETRY_DURATION_MULTIPLIER_MINUTES`: The amount to multiply the Retry-After Duration by each time the system reaches a new rate limit duration bracket. Default value is 2 minutes.

```
BLACKDUCK_USE_QUEUE_RATE_LIMITING=TRUE
BLACKDUCK_INITIAL_RATE_LIMIT_DURATION_BRACKET_THRESHOLD_MINUTES=2
BLACKDUCK_RATE_LIMIT_DURATION_THRESHOLD_BRACKET_INCREMENT_MINUTES=2
BLACKDUCK_INITIAL_RETRY_DURATION_MINUTES=1
RETRY_DURATION_MULTIPLIER_MINUTES_KEY=2
```

Configuring hierarchical subproject license conflicts

By default, hierarchical subproject license conflicts are enabled in your environment. You can disable hierarchical subproject license conflicts by setting the following parameter:

```
USE_HIERARCHICAL_LICENSE_CONFLICTS=FALSE
```

Subproject depth is set to 5 levels by default, but can be configured with the following parameter:

```
HIERARCHICAL_LICENSE_CONFLICT_DEPTH_LIMIT=<value desired>
```

Provisioning JWT public/private key pairs

To enhance the security and flexibility of JWT management, our system now supports the optional provisioning of public/private key pairs. This allows you to securely provide and manage these keys, ensuring they are only used by the appropriate services, such as the Authentication service for private keys and public API services for public keys.

Currently, only RSA keys (PEM encoded) are supported. Specifically, public keys must be in X.509 format, and private keys must be in PKCS#8 format.

Creating Docker secrets

To create public and private secrets in Docker:

1. Enter the following commands:

   ```
   docker secret create hub_JWT_PUBLIC_KEY public-key.pem
   docker secret create hub_JWT_PRIVATE_KEY private-key.pem
   ```

2. Edit `docker-compose.local-overrides.yml` to use JWT secrets and deploy:

   ```
   docker stack deploy -c docker-compose.yml -c docker-compose.local-overrides.yml jwt-swarm
   ```

Sample overrides file

Here is a sample `docker-compose.local-overrides.yml` file (integration service configured as needed). The comments in this file show how to override some of the most popular set of options. However, it is possible to override any Docker configuration setting, for example Port mappings, by adding the override here.

```
version: '3.6'
services:
  authentication:
    secrets:
      - JWT_PUBLIC_KEY
      - JWT_PRIVATE_KEY
  webapp:
    secrets:
      - JWT_PUBLIC_KEY
      - JWT_PRIVATE_KEY
```

```
  scan:
    secrets:
      - JWT_PUBLIC_KEY
      - JWT_PRIVATE_KEY
  storage:
    secrets:
      - JWT_PUBLIC_KEY
      - JWT_PRIVATE_KEY
  jobrunner:
    secrets:
      - JWT_PUBLIC_KEY
      - JWT_PRIVATE_KEY
  bomengine:
    secrets:
      - JWT_PUBLIC_KEY
      - JWT_PRIVATE_KEY
  matchengine:
    secrets:
      - JWT_PUBLIC_KEY
      - JWT_PRIVATE_KEY
  #integration:
  #  secrets:
  #    - JWT_PUBLIC_KEY
  #    - JWT_PRIVATE_KEY
secrets:
  JWT_PUBLIC_KEY:
    external: true
    name: "hub_JWT_PUBLIC_KEY"
  JWT_PRIVATE_KEY:
    external: true
    name: "hub_JWT_PRIVATE_KEY"
```

Configuring session token invalidation

To activate session token invalidation, configure the `blackduck-config.env` file and set `JWT_BLOCK_LIST_CHECK=true`.

Configuring Preserve Component Relationships on SBOM Import

Preserving component relationships on SBOM import is disabled by default. To enable this feature, you must configure the following setting:

```
blackduck.scan.sbom.preserve.relationships=true
```

Once this configuration is set, Black Duck SCA will automatically preserve relationship data from imported SBOM reports.

5. Uninstalling Black Duck

Follow these instructions to uninstall Black Duck:

- Stop and remove the containers, networks and secrets.

```
docker stack rm ${stack name}
```

- Remove all unused volumes:

```
docker volume prune -a
```

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.

6. Upgrading Black Duck

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

Note: A database administrator will need to enable installation of the `hstore` PostgreSQL extension before installing or upgrading to 2023.4.0 or later.

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 Black Duck 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-2026.7.2.tar.gz
   ```
3. Unpack the Black Duck`.tar` file:

   ```
   tar xvf hub-2026.7.2.tar
   ```

Download using the wget command

1. Run the following command:

   ```
   wget https://github.com/blackducksoftware/hub/archive/v2026.7.2.tar.gz
   ```
2. Uncompress the Black Duck `.gz` file:

   ```
   gunzip v2026.7.2.tar.gz
   ```
3. Unpack the Black Duck`.tar` file:

   ```
   tar xvf v2026.7.2.tar
   ```

Migration script to purge unused rows in the audit event table

Please note that this section is only applicable when upgrading from Black Duck versions older than 2019.12.0.

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.

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.
  4. 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

Once the upgrade is complete, it is strongly recommended that you run the `VACUUM FULL` command on the `audit_event` table to optimize PostgreSQL performance.

- Depending on your system usage, running the `VACUUM FULL` 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 FULL` command, there may be a degradation of performance.

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

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

1. Get the size of the `audit_event` table and ensure that you have enough space to run the `VACUUM FULL` command.
2. Run the `docker ps` command to get the ID of the PostgreSQL container.
3. Run the following command to access the PostgreSQL container.

   ```
   docker exec -it <container_ID> psql bds_hub
   ```
4. Run the following `VACUUM FULL` 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 FULL` command, and when it's finished, you restart the deployment.

Upgrading from an existing Docker architecture

To upgrade from a previous version of Black Duck:

1. Migrate your PostgreSQL database.
2. Upgrade Black Duck.

Migrating your PostgreSQL databases

Black Duck 2022.10.0 has migrated its PostgreSQL image from version 11 to version 13 and supports upgrading from versions using either the PostgreSQL 9.6 container (versions 4.2 through and including 2021.10.x) or the PostgreSQL 11 container (versions 2022.2.0 through and including 2022.7.x). During installation, the `blackduck-postgres-upgrader` container will migrate the existing database to PostgreSQL 13 and then exit upon completion.

Please note the following:

- Customers with non-core PG extensions are STRONGLY encouraged to uninstall them before migrating and reinstall them after the migration completes successfully; otherwise, the migration is likely to fail.
- Customers with replication set up will need to follow the instructions in the pg_upgrade documentation BEFORE they migrate. If the preparations described there are not made, the migration will likely succeed, but the replication setup will break.
- Customers not using the Black Duck-supplied PostgreSQL image must ensure that they are using a supported version. Black Duck 2022.10.0 supports PostgreSQL versions 13.x and 14.x, and Black Duck recommends using the latest release of 14.x.
    - Community PostgreSQL users must follow PostgreSQL 14 instructions (https://www.postgresql.org/docs/14/pgupgrade.html) for PostgreSQL upgrades.
    - Users of third-party PostgreSQL, for example, RDS, must follow instructions from their providers.
- Users of the internal PostgreSQL container, PostgreSQL will be upgraded automatically if needed.
- Black Duck users on Black Duck versions prior to 4.2.0 should contact Black Duck Technical Support before upgrading.

Important: Before starting the migration:

- Ensure that you have an extra 10% disk space to avoid unexpected issues arising from disk usage due to the data copying of system catalogs.
- Review root directory space and volume mounts to avoid running out of disk space as this can cause Linux system disruptions.

The migration is completely automatic; no extra actions are needed beyond those for a standard Black Duck upgrade. The `blackduck-postgres-upgrader` container MUST run as root in order to make the layout and UID changes described above.

On subsequent Black Duck restarts, `blackduck-postgres-upgrader` will determine that no migration is needed and immediately exit.

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 -c sizes-gen04/120sph.yaml hub
       ```

       (using the DB container)
     - ```
       docker stack deploy -c docker-compose.externaldb.yml -c sizes-gen04/120sph.yaml 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 sizes-gen04/120sph.yaml -c docker-
       compose.bdba.yml hub
       ```

       (using the DB container with Black Duck Binary Analysis)
     - ```
       docker stack deploy -c docker-compose.externaldb.yml -c sizes-gen04/120sph.yaml -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 sizes-gen04/120sph.yaml -c docker-
       compose.local-overrides.yml hub
       ```

       (using the DB container)
     - ```
       docker stack deploy -c docker-compose.externaldb.yml -c sizes-gen04/120sph.yaml -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 sizes-
       gen04/120sph.yaml -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
        sizes-gen04/120sph.yaml -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 sizes-gen04/120sph.yaml  -c docker-
     compose.readonly.yml -c docker-compose.local-overrides.yml hub
     ```

Note: The resource definition file used in the commands above can be changed to better fit your scanning patterns and usage. See the Distribution section for a list of all available options. Please note the system requirements for each option as they increase with greater resource definitions.

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

Backing up and restoring data

If the database is internal (i.e. using the postgres container as the backend database), you could make use of the scripts under the `docker-swarm/bin` folder for the backup and restore.

Backing up your data

1. Start HUB as database migrate mode:

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

2. Run the following command:

```
./hub_create_data_dump.sh /pathtodbdump
```

You will have the following files under `/pathtodbdump` after the script completes:

- `bds_hub.dump`
- `globals.sql`

Restoring your data

You must first delete the existing `hub_postgres96-data-volume` before you can restore your database. Use the following command to do so:

```
STACK=${STACK:-hub}  -- change the "hub" part to match your actual deployment
docker volume rm ${STACK}_postgres96-data-volume
```

Once you have deleted the existing volume, you can restore your database by following these steps:

1. Start a brand new HUB as database migrate mode and then use `hub_db_migrate.sh` to restore the data:

```
docker stack deploy -c docker-compose.dbmigrate.yml hub
./hub_db_migrate.sh /pathtodbdump
```

2. And finally, start HUB normally:

```
docker stack rm hub
docker stack deploy -c docker-compose.yml -c sizes-gen04/120sph.yaml -c docker-compose.local-overrides.yml hub
```

The resource definition file used above can be changed to better fit your scanning patterns and usage. See the Distribution section for a list of all available options. Please note the system requirements for each option as they increase with greater resource definitions.

If the database is external, please contact your database administrator for the data backup.

7. Docker containers

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

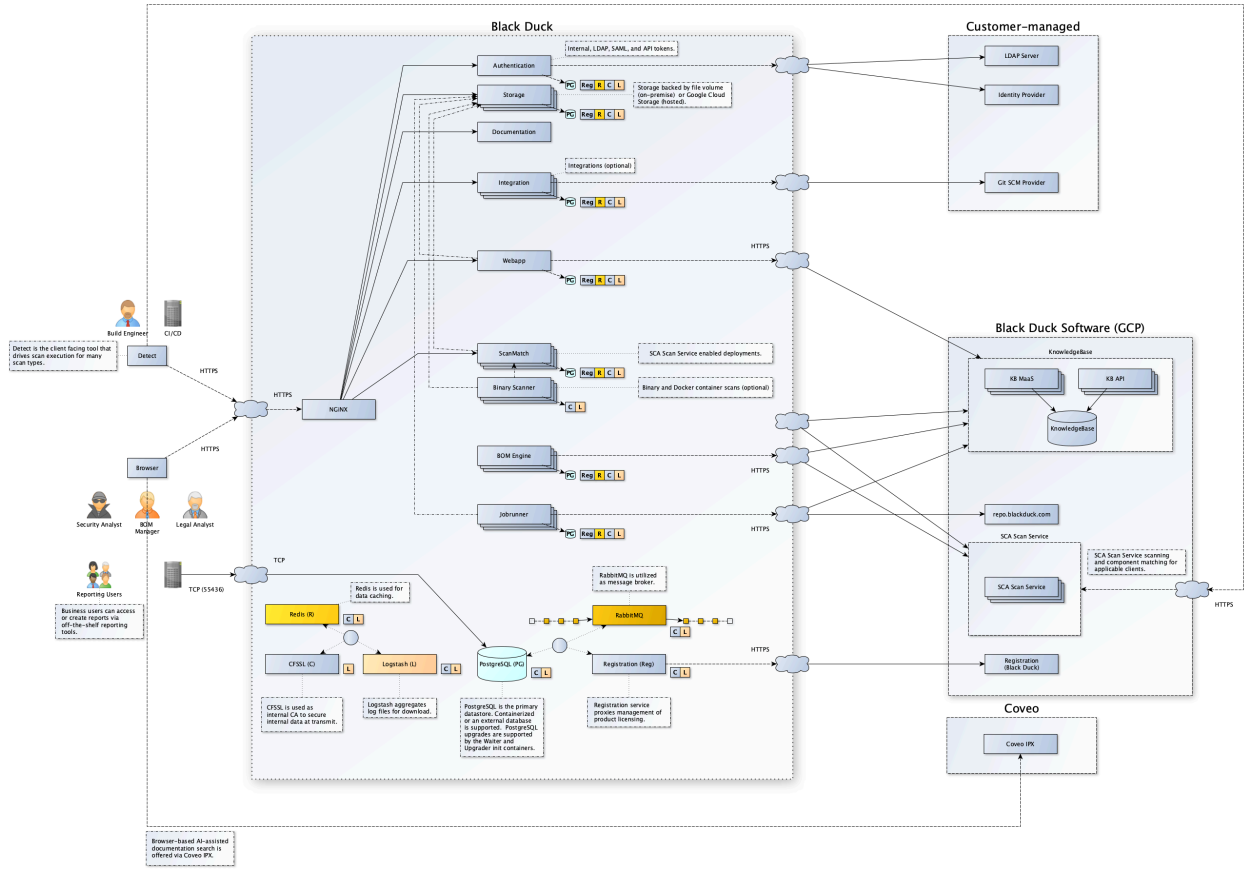
Application services

- Authentication
- Binary Analysis - Required if Black Duck Binary Analysis is enabled.
- BOM Engine
- DB - This container is not included in the Black Duck application if you use an external PostgreSQL instance.
- Documentation
- Integration
- Jobrunner
- Registration
- Scanmatch
- Storage
- Web application
- Web server

Infrastructure services

- CFSSL
- Logstash
- Rabbitmq
- Redis

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	blackducksoftware/blackduck-authentication:2026.7.2
Description	The authentication service is the container that all authentication-related requests are made against.
Scalability	There should only be a single instance of this container. It currently cannot be scaled.
Links/Ports	Nothing external (8443 internally). This container will need to connect to these other containers/services: • postgres • cfssl • logstash • registration • webapp

Container Name: blackduck-authentication	
	The container needs to expose 8443 to other containers that will link to it.
Alternate Host Name Environment Variables	There are times when running in other types of orchestrations that it is useful to have host names set for these containers that are not the default that Docker Swarm uses. These environment variables can be set to override the default host names: • postgres - $HUB_POSTGRES_HOST • cfssl - $HUB_CFSSL_HOST • logstash - $HUB_LOGSTASH_HOST • registration - $HUB_REGISTRATION_HOST • webapp - $HUB_WEBAPP_HOST
Resources/ Constraints	• Default max Java heap size: 512MB • Container memory: 1GB • Container CPU: 1 CPU
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	`blackduck-config.env`

Binaryscanner container

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

Container Name: bdba-worker	
Image Name	image: blackducksoftware/bdba-worker:2023.03
Description	This container analyzes binary files. This container is currently only used if Black Duck - Binary Analysis is enabled.
Scalability	This container can be scaled.
Links/Ports	This container needs to connect to these containers/services: • cfssl • logstash • rabbitmq • webserver The container will need to expose port 5671 to other containers that will link to it.
Alternate Host Name Environment Variables	There are times when running in other types of orchestrations that it is useful to have host names set for these containers that are not the default that Docker Swarm uses. These environment variables can be set to override the default host names: • cfssl: $HUB_CFSSL_HOST • logstash: $HUB_LOGSTASH_HOST • rabbitmq: $RABBIT_MQ_HOST • webserver: $HUB_WEBSERVER_HOST

Container Name: bdba-worker	
Resources/ Constraints	• 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

Bomengine container

Container Name: bomengine	
Image Name	blackducksoftware/blackduck-bomengine:2026.7.2
Description	The bomengine container is responsible for building BOMs and keeping them up-to-date.
Scalability	The container can be scaled
Links/Ports	The bomengine container needs to connect to the following container/services: • 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 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: • postgres: $HUB_POSTGRES_HOST • registration: $HUB_REGISTRATION_HOST • logstash: $HUB_LOGSTASH_HOST • cfssl: $HUB_CFSSL_HOST
Resources/ Constraints	• Default max Java heap size: 4GB • Container memory: 4.5GB
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	`blackduck-config.env`

CFSSL container

Container Name: blackduck-cfssl	
Image Name	blackducksoftware/blackduck-cfssl:1.0.2

Container Name: blackduck-cfssl	
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	• 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	`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:9.6-1.1
Description	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. 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 `hub_reportdb_changepassword.sh` script. This port and user can be used for reporting and data extraction. Refer to the *Report Database* guide for more information on the report database.
Scalability	There should only be a single instance of this container. It should not be scaled.
Links/Ports	The DB container needs to connect to these containers/services: • logstash • cfssl The container needs to expose port 5432 to other containers that will link to it within the Docker network. This container exposes port 55436 outside of the Docker network for database reporting.

Container Name: blackduck-postgres	
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 override the default host names: • logstash: $HUB_LOGSTASH_HOST • cfssl: $HUB_CFSSL_HOST
Resource/ Constraints	• Default max Java heap size: N/A • Container memory: 3GB • Container CPU: 1 CPU
Users/Groups	This container runs as UID 1001. If the container is started as UID 0 (root) then the user will be switched to UID 1001:root before executing its main process. This container is not able to start with any other user id.
Environment File	N/A

Documentation container

Container Name: blackduck-documentation	
Image Name	blackducksoftware/blackduck-documentation:2026.7.2
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	This container must connect to these other containers/services: • logstash • cfssl The documentation container must expose port 8443 to other containers that link to it.
Alternate Host Name Environment Variables	There are times when running in other types of orchestrations that it is useful to have host names set for these containers that are not the default that Docker Swarm uses. These environment variables can be set to override the default host names: • logstash: $HUB_LOGSTASH_HOST • cfssl: $HUB_CFSSL_HOST
Resources/ Constraints	• Default Max Java Heap Size: 512MB • Container Memory: 512MB • Container CPU: unspecified
Users/Groups	This container runs as UID 8080. If the container is started as UID 0 (root) then the user will be switched to UID 8080:root before executing its main process. This container is also able to be started as a random UID as long as it is also started within the root group (GID/fsGroup 0).
Environment File	`blackduck-config.env`

Integration container

Container Name: blackduck-integration	
Image Name	blackducksoftware/blackduck-integration:2026.7.2
Description	The integration service that is responsible for Artifactory integration and Git SCM provider integration scan functionality.
Scalability	There can be multiple instances of this container. • Requirement that all scan services are on the same Docker network as the other containers. • Integration services can be present on different hosts or nodes.
Links/Ports	This container must connect to these other containers/services: • cfssl • postgres • rabbitmq • registration • logstash • scan The integration container must expose port 8443 to other containers that link to it.
Alternate Host Name Environment Variables	There are times when running in other types of orchestrations that it is useful to have host names set for these containers that are not the default that Docker Swarm uses. These environment variables can be set to override the default host names: • postgres: $HUB_POSTGRES_HOST • registration: $HUB_REGISTRATION_HOST • logstash: $HUB_LOGSTASH_HOST • cfssl: $HUB_CFSSL_HOST • rabbitmq: $RABBIT_MQ_HOST, $RABBIT_MQ_PORT
Resources/ Constraints	• Default Max Java Heap Size: 5120MB • Container Memory: 5120MB • Container CPU: 1 CPU
Users/Groups	This container runs as UID 8080. If the container is started as UID 0 (root) then the user will be switched to UID 8080:root before executing its main process. This container is also able to be started as a random UID as long as it is also started within the root group (GID/fsGroup 0).
Environment File	`blackduck-config.env`

Jobrunner container

Container Name: blackduck-Jobrunner	
Image Name	blackducksoftware/blackduck-jobrunner:2026.7.2

Container Name: blackduck-Jobrunner	
Description	The Job Runner container is the container that is responsible for running all of the application's jobs. This includes matching, BOM building, reports, data updates, and so on. This container does not have any exposed ports.
Scalability	This container can be scaled.
Links/Ports	The Job Runner container needs to connect to these containers/services: • 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 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: • postgres: $HUB_POSTGRES_HOST • registration: $HUB_REGISTRATION_HOST • logstash: $HUB_LOGSTASH_HOST • cfssl: $HUB_CFSSL_HOST
Resources/ Constraints	• Default max Java heap size: 4GB • Container memory: 4.5GB • Container CPU: 1 CPU
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	`blackduck-config.env`

Logstash container

Container Name: blackduck-logstash	
Image Name	blackducksoftware/blackduck-logstash:1.0.10
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	• Default max Java heap size: 1GB • Container memory: 1GB • 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.

Container Name: blackduck-logstash	
	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	`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.
Links/Ports	This container needs to connect to these other containers/services: • cfssl The container needs to expose port 5671 to other containers that will link to it.
Alternate Host Name Environment Variables	There are times when running in other types of orchestrations that it is useful to have host names set for these containers that are not the default that Docker Swarm uses. These environment variables can be set to override the default host names: • cfssl: $HUB_CFSSL_HOST
Constraints	• Default max Java heap size: N/A • Container memory: 1GB • 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 it's 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	`blackduck-config.env`

Redis container

Container Name: blackduck-redis	
Image Name	redis:7.0.9-alpine3.17
Description	This container enables more consistent caching functionality in Black Duck and is used to improve application performance. Redis is enabled by default as a primary cache mechanism.
Configuration	To configure Redis, use the following settings: • blackduck-config.env environment file to configure redis settings • Redis settings:

Container Name: blackduck-redis	
	• 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. • 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	The Redis container needs to connect to these containers/services: • logstash • cfssl
Alternate Host Name Environment Variables	N/A
Resources/ Constraints	• Default max Java heap size: N/A • Container memory: • 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`

Registration container

Container Name: blackduck-registration	
Image Name	blackducksoftware/blackduck-registration:2026.7.2
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	The Registration container needs to connect to this containers/services: • logstash • cfssl The container needs to expose port 8443 to other containers that link to it.

Container Name: blackduck-registration	
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 override the default host names: • logstash: $HUB_LOGSTASH_HOST • cfssl: $HUB_CFSSL_HOST
Resources/ Constraints	• Default max Java heap size: 512MB • Container memory: 640MB • Container CPU: Unspecified
Users/Groups	This container runs as UID 8080. If the container is started as UID 0 (root) then the user will be switched to UID 8080:root before executing its main process. This container is also able to be started as a random UID as long as it is also started within the root group (GID/fsGroup 0).
Environment File	`blackduck-config.env`

Scanmatch container

Container Name: blackduck-scanmatch	
Image Name	blackducksoftware/blackduck-scanmatch:2026.7.2
Description	The scanmatch service functions as the container that handles all scan data requests and retrieves component match information from the Cloud Knowledge Base.
Scalability	This container can be scaled.
Links/Ports	This container needs to connect to these containers/services: • postgres • registration • logstash • cfssl • rabbitmq The container needs to expose port 8443 to other containers that will link to it. Connects externally to Cloud KB services.
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 override the default host names: • postgres: HUB_POSTGRES_HOST, HUB_POSTGRES_PORT • registration: HUB_REGISTRATION_HOST, HUB_REGISTRATION_PORT • logstash: HUB_LOGSTASH_HOST • cfssl: HUB_CFSSL_HOST, HUB_CFSSL_PORT • rabbitmq: RABBIT_MQ_HOST, RABBIT_MQ_PORT
Resources/ Constraints	Set in the `gen05` sizings.
Users/Groups	This container runs as UID 8080. If the container is started as UID 0 (root) then the user will be switched to UID 8080:root before executing its main process.

Container Name: blackduck-scanmatch	
	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	`blackduck-config.env`

Storage container

Container Name: blackduck-storage	
Image Name	blackducksoftware/blackduck-storage:2026.7.2
Description	The storage service provides functionality for many users with the ability to upload files, download files, and define the default file when a file has multiple available versions.
Scalability	This container can be scaled.
Links/Ports	This container needs to connect to these containers/services: • postgres • registration • rabbitmq • logstash • cfssl
Alternate Host Name Environment Variables	There are times when running in other types of orchestrations that any individual service name may be different. For example, you may have an external PostgreSQL endpoint which is resolved through a different service name. To support such use cases, these environment variables can be set to override the default host names: • postgres: $HUB_POSTGRES_HOST • registration: $HUB_REGISTRATION_HOST • rabbitmq: $RABBIT_MQ_HOST • logstash: $HUB_LOGSTASH_HOST • cfssl: $HUB_CFSSL_HOST
Resources/ Constraints	• Default max java heap size: 512MB • Container memory: 1GB
Users/Groups	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	`blackduck-config.env`

Webapp container

Container Name: blackduck-webapp	
Image Name	blackducksoftware/blackduck-webapp:2026.7.2
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

Container Name: blackduck-webapp	
	not exposed outside of the Docker network. There is an NGiNX reverse proxy (as described in the WebServer container) that is exposed outside of the Docker network instead.
Scalability	There should only be a single instance of this container. It should not be scaled.
Links/Ports	The webapp container needs to connect to these containers/services: • postgres • registration • logstash • cfssl The container needs to expose port 8443 to other containers that will link to it.
Alternate Host Name Environment Variables	There are times when running in other types of orchestrations that it is useful to have host names set for these containers that are not the default that Docker Swarm uses. These environment variables can be set to override the default host names: • postgres: $HUB_POSTGRES_HOST • registration: $HUB_REGISTRATION_HOST • logstash: $HUB_LOGSTASH_HOST • cfssl: $HUB_CFSSL_HOST
Resources/ Constraints	• Default max Java heap size: 2GB • Container memory: 2.5GB • Container CPU: 1 CPU
Users/Groups	This container runs as UID 8080. If the container is started as UID 0 (root) then the user will be switched to UID 8080:root before executing its main process. This container is also able to be started as a random UID as long as it is also started within the root group (GID/fsGroup 0).
Environment File	`blackduck-config.env`

Webserver container

Container Name: blackduck-webserver	
Image Name	blackducksoftware/blackduck-nginx:1.0.32
Description	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. HTTP/2 and TLS 1.3 are supported
Scalability	The container should not be scaled.
Links/Ports	The Web App container needs to connect to these containers/services: • webapp • cfssl • documentation • scan • authentication

Container Name: blackduck-webserver	
	This container exposes port 443 outside of the Docker network.
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 override the default host names: • webapp: $HUB_WEBAPP_HOST • cfssl: $HUB_CFSSL_HOST • scan: $HUB_SCAN_HOST • documentation: $HUB_DOC_HOST • authentication: $HUB_AUTHENTICATION_HOST
Resources/ Constraints	• 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-webserver.env`, `blackduck-config.env`