BLACKDUCK

BY **SYNOPSYS**®

Installing Black Duck using Docker Compose

Version 2019.6.1

This edition of the *Installing Black Duck using Docker Compose* refers to version 2019.6.1 of Black Duck.

This document created or updated on Tuesday, July 9, 2019.

Please send your comments and suggestions to:

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

Contents

Black Duck documentation

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

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

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

Black Duck integration documentation can be found on Confluence.

Customer support

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

You can contact Synopsys Support in several ways:

- Online: https://www.synopsys.com/software-integrity/support.html
- Email: software-integrity-support@synopsys.com
- Phone: See the Contact Us section at the bottom of our support page to find your local phone number.

Another convenient resource available at all times is the online customer portal.

Synopsys Software Integrity Community

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

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

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

Training

Synopsys Software Integrity, Customer Education (SIG Edu) is a one-stop resource for all your Black Duck education needs. It provides you with 24x7 access to online training courses and how-to videos.

New videos and courses are added monthly.

At Synopsys Software Integrity, Customer Education (SIG Edu), you can:

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

Learn more at https://community.synopsys.com/s/education.

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

Important: Synopsys will be deprecating support for Docker Compose starting with the Black Duck 2019.4.0 release. Docker Compose will be supported until December 31, 2019.

Black Duck Architecture

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

The Docker architecture brings these significant improvements to Black Duck:

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

See Docker containers, for more information on the Docker containers that comprise the Black Duck application.

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

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

Components hosted on Black Duck servers

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

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

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

Getting started

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

New installations

For new installation of Black Duck:

1. Read this planning chapter to review all requirements.

2. After ensuring that you meet all requirements, go to Chapter 3 for installation instructions.

3. Review Chapter 4 for any administrative tasks.

Upgrading from a previous version of Black Duck

1. Read this planning chapter to review all requirements,

2. After ensuring that you meet all requirements, go to Chapter 6 for upgrade instructions.

3. Review Chapter 4 for any administrative tasks.

Hardware requirements

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

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

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

- 5 CPUs
- 20 GB RAM

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

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

The descriptions of each container document the individual requirements for each container if it will be running on a different machine or if more than one instance of a container will be running (currently only supported for the job runner, scan, and binaryscanner containers).

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

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

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

Docker requirements

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

Black Duck supports Docker versions 17.12.x, 18.03.x, 18.06.x, and 18.09.x, (CE or EE).

The minimum supported version of docker-compose must be able to read Docker Compose 2.3 files.

Operating systems

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

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

In addition, Black Duck supports other Linux operating systems that support the supported Docker versions.

> Note: Docker CE does not support Red Hat Enterprise Linux. Click here for more information.

Windows operating system is currently not supported.

Software requirements

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

- Chrome 75.0.3770.80 (Official Build) (64-bit)
- Firefox 67.0.1 (64-bit)
- Internet Explorer 11.765.17134.0
- Microsoft Edge 42.17134.1.0
- Microsoft EdgeHTML 17.17134
- Safari 12.1 (14607.1.40.1.4)

Note that Black Duck does not support compatibility mode.

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

Network requirements

Black Duck requires the following ports to be externally accessible:

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

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

- updates.suite.blackducksoftware.com (to register your software)
- kb.blackducksoftware.com (access the Black Duck KB data)

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

Database requirements

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

Prior to installing Black Duck, determine whether you want to use the database container that is automatically installed or an external PostgreSQL instance.

For an external PostgreSQL instance, Black Duck supports:

- PostgreSQL 9.6.x via Amazon Relational Database Service (RDS)
- PostgreSQL 9.6.x via Google Cloud SQL
- PostgreSQL 9.6.x (Community Edition)

Refer to Configuring an external PostgreSQL instance for more information.

PostgreSQL versions

For Black Duck version 2019.6.1, the currently-supported version of PostgreSQL is 9.6.x, which is the version

supplied in Black Duck's PostgreSQL container. If you choose to run your own PostgreSQL instance, you must be at PostgreSQL version 9.6.x for compatibility with Black Duck version 2019.6.1.

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.

  Currently Black Duck requires PostgreSQL version 9.6.x.

Additional port information

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

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

The complete list of ports that must remain unblocked is:

- 443
- 8443
- 8000
- 8888
- 8983
- 16543
- 17543
- 16545
- 16544
- 55436

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.

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-2019.6.1.tar.gz`

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

   `tar xvf hub-2019.6.1.tar`

Download using the wget command

1. Run the following command:

   `wget https://github.com/blackducksoftware/hub/archive/v2019.6.1.tar.gz`

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

   `gunzip v2019.6.1.tar.gz`

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

   `tar xvf v2019.6.1.tar`

Distribution

The `docker-compose` directory has the 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 configured in a `.yml` file.
- `docker-compose.yml`: Docker Compose file when using the database container provided by Black Duck.
- `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.

Installing Black Duck

This section provides instructions for new installations of Black Duck.

Prior to installing Black Duck, determine if there are any settings that need to be configured.

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

> **Note:** These instructions are for new installations of Black Duck. Refer to Chapter 6 for more information about upgrading Black Duck.

In the instructions below, use the version of the `.yml` file located in the `docker-compose` directory.

- To install Black Duck with the PostgreSQL database container:

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

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

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

- To install Black Duck with an external PostgreSQL instance:

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

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

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

Black Duck is installed.

You can confirm that the installation was successful by running the `docker ps` command to view the status

of each container. A "healthy" status indicates that the installation was successful. Note that the containers may be in a "starting" state for a few minutes post-installation.

Once all 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.

This chapter describes these administrative tasks:

A

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

B

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

C

- [Certificate, replacing the existing self-signed certificate](#) for the Web Server with a custom certificate.
- [Component Dashboard, configuring the refresh rate](#)

H

- [Hierarchical BOM, enabling](#)

I

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

L

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

M

- [Memory limits, changing the default memory limits.](#)

O

- [Override file, using](#)

P

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

R

S

T

U

W

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-compose` 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, do the following:

  1. Stop the containers:

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

  2. Open the file and edit the settings.

  3. Reconnect the containers:

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

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-compose` 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.

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

Increasing the size of the binary scan file

When using Black Duck - Binary Analysis, the maximum size of the binary that can be scanned is 6 GB. You

can increase this limit by adding the environment variable BINARY_UPLOAD_MAX_SIZE to the `hub_webserver.env` file in the `docker-compose` directory and specifying a value in megabytes.

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

> BINARY_UPLOAD_MAX_SIZE=7168m

Managing certificates

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

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

You will also receive a message regarding the certificate when connecting to the Black Duck server when scanning as the scanner cannot verify the certificate because it is a self-signed and is not issued by a CA.

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

To create an SSL certificate keystore

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

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

   where:

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

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

   For example:

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

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

2. Send the CSR to the CA by their preferred method (usually through a web portal).

3. Indicate that you need a certificate for an Apache web server.

4. Provide any requested information about your company to the CA. This information must match your domain registry information.

5. Once you receive your certificate from the CA, use the instructions in the next section to upload the certificate into a Black Duck instance.

Using custom certificates

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

1. Create a file named WEBSERVER_CUSTOM_CERT_FILE with the certificate file.

    For example, if the certificate file you received from a Certificate Authority is `Hub.crt`, rename the file to WEBSERVER_CUSTOM_CERT_FILE.

2. Create a file named WEBSERVER_CUSTOM_KEY_FILE with the key file.

    For example, if the key file you received from a Certificate Authority is `Hub.key`, rename the file to WEBSERVER_CUSTOM_KEY_FILE.

3. Mount a directory that contains both files to `/run/secrets` by adding the volumes section to the webserver container in the `docker-compose.local-overrides.yml` located in the `docker-compose` directory:

    ```
    webserver:
    volumes: ['/directory/where/the/cert-key-
    files/are:/run/secrets']
    ```

    Be sure that you remove the comment character (#) located before the container name.

4. Optionally, add the URL and path to the certificate in the `healthcheck` property in the webserver container by editing the `docker-compose.local-overrides.yml` file located in the `docker-compose` directory:

    Verify that the path to the certificate is correct, otherwise the Webserver container will appear as unhealthy.

5. Start the webserver container.

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. Mount a directory that contains a file named AUTH_CUSTOM_CA, the custom certificate authority certificate file to `/run/secrets` by adding the volumes section to the webserver and authentication services in the `docker-compose.local-overrides.yml` file located in the `docker-compose` directory:

```
webserver:
volumes: ['/path/where/AUTH_CUSTOM_CA/is/stored/on/host:/run/secrets']

authentication:
volumes: ['/path/where/AUTH_CUSTOM_CA/is/stored/on/host:/run/secrets']
```

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

2. Start the webserver container and the authentication service.

3. 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 username of the certificate used for authentication must exist in the Black Duck system as its Common Name (CN).

Configuring the Component Dashboard refresh rate

By default, the Component Dashboard refreshes every five minutes. If you notice a lag between your changes and updates to the Component Dashboard, you may want to modify this refresh rate.

Caution: Decreasing the default refresh rate may increase the load on the Black Duck server.

To modify the refresh rate

1. Add the `com.blackducksoftware.bom.aggregate.component_dashboard_refresh_interval_ms` system property to the `blackduck-config.env` file in the `docker-compose` directory.

    Enter the new value in milliseconds, so to change the refresh rate to one minute, add the following:

    ```
    com.blackducksoftware.bom.aggregate.component_dashboard_refresh_interval_ms=60000
    ```

2. Restart the containers.

Enabling the hierarchical BOM

By default, the hierarchical BOM is disabled. To enable this feature, add the HUB_HIERARCHICAL_BOM environment variable to an `.env` file. Set the value to "true", for example, in the `.env` file, HUB_HIERARCHICAL_BOM=true. Setting the value to "false" disables the feature.

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

```
webapp:
 environment: {HUB_HIERARCHICAL_BOM: true
```

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

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

Including ignored components in reports

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

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

Configuring secure LDAP

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

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

1. Obtaining your LDAP information.

2. Using the Black Duck UI to import the server certificate.

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

Obtaining your LDAP information

Contact your LDAP administrator and gather the following information:

LDAP Server Details

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

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

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

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

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

Example of an LDAP name: `jdoe`

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

LDAP Users Attributes

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

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

  Example: `dc=example,dc=com`

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

  Example: `uid={0}`

Test Username and Password

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

Importing the server certificate
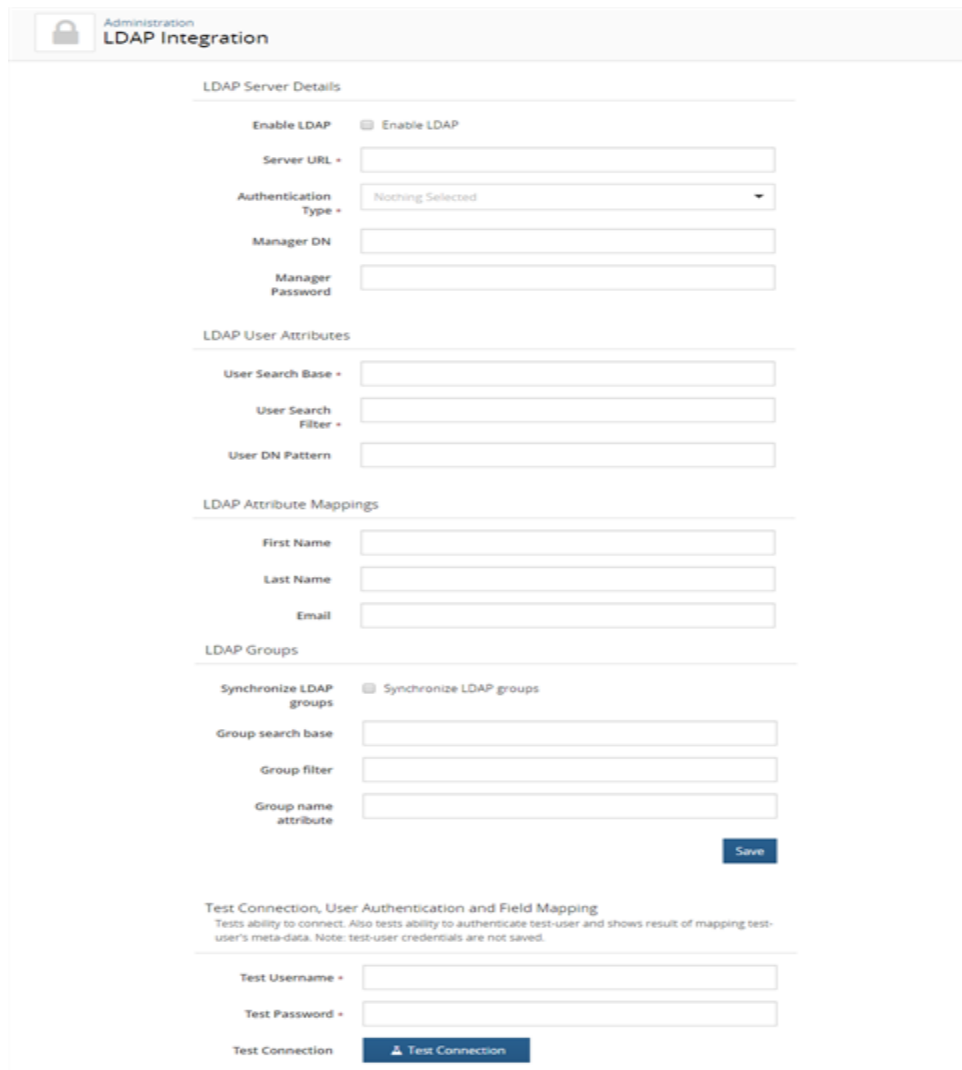
To import the server certificate

1. Log in to Black Duck as a system administrator.

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

   The Administration page appears.

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

4. Select the **Enable LDAP** option and complete the information in the **LDAP Server Details** and **LDAP User Attributes** sections, as described above. In the **Server URL** field, ensure that you have configured the secure LDAP server: the protocol scheme is ldaps://.

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

6. If there are no issues with the certificate, it is automatically imported and the "Connection Test Succeeded" message appears:

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

Do one of the following:

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

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

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

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

LDAP trust store password

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

- Mount a directory that contains a text 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-compose` directory:

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

Be sure to remove the comment character (#) before `webapp`.

Accessing log files

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

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

To download the log files from the Black Duck UI
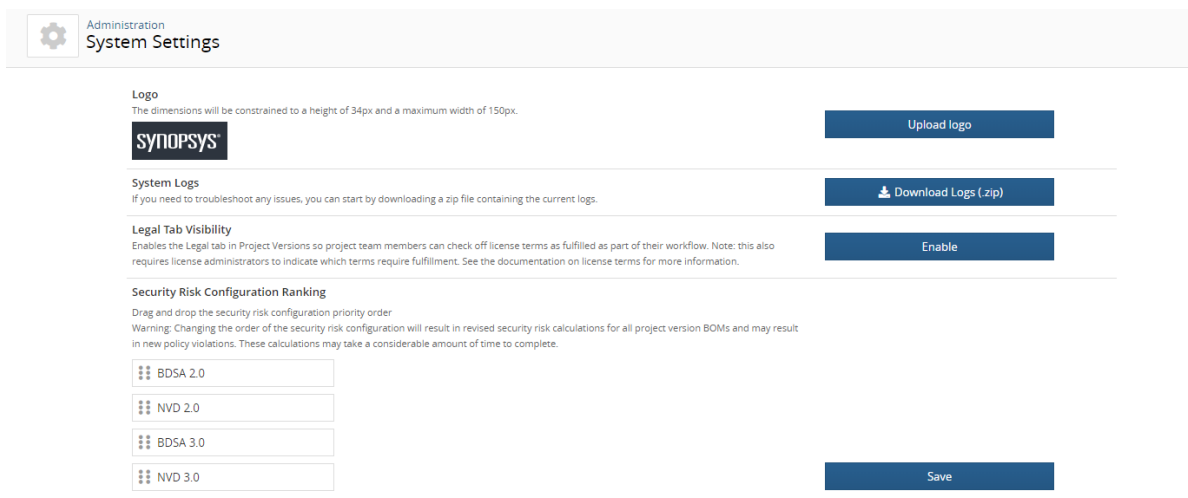
1. Log in to Black Duck with the System Administrator role.

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

   The Administration page appears.

3. Select **System Settings**.

   The System Settings page appears.

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

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

Obtaining logs

To obtain logs from the containers:

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

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

Viewing log files for a container

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

```
docker-compose logs
```

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

Purging logs

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-compose` 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 place 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

Changing the default webapp container memory limits

There are two memory settings for the webapp container:

- The HUB_MAX_MEMORY environment variable controls the maximum Java heap size.

- The `mem_limit` setting controls the overall memory of the web app container.

    Note that the `mem_limit` setting must be higher than the maximum Java heap size. If updating the Java heap size, Synopsys recommends setting the `mem_limit` value to at least 1GB higher 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 webapp container to 8GB and the `mem_limit` to 9GB.

Original values:

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

Updated values:

```
webapp:

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

Changing the default jobrunner container memory limits

There are two memory settings for the jobrunner container:

- The HUB_MAX_MEMORY environment variable controls the maximum Java heap size.

- The `mem_limit` setting controls the overall memory of the job runner container.

  Note that the `mem_limit` setting must be higher than the maximum Java heap size. If updating the Java heap size, Synopsys recommends setting the `mem_limit` value to at least 1GB higher than the maximum Java heap size.

> **Note:** These settings apply to all jobrunner containers, including scaled jobrunner 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 job runner container to 8GB and `mem_limit` to 9GB.

Original values:

```
jobrunner:

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

Updated values:

```
jobrunner:

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

Changing the default scan container memory limits

There are two memory settings for the scan container:

- The HUB_MAX_MEMORY environment variable controls the maximum Java heap size.

- The `mem_limit` setting controls the overall memory of the container.

  Note that the `mem_limit` setting must be higher than the maximum Java heap size. If updating the Java heap size, Synopsys recommends setting the `mem_limit` value to at least 1GB higher 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 changes the maximum Java heap size for the scan container to 4GB and `mem_limit` to 5GB.

Original values:

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

Updated values:

```
scan:
 environment: {HUB_MAX_MEMORY: 4096m}
 #deploy:
  #mem_limit: 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.

```
binaryscanner:
 mem_limit: 4096M
```

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-compose` 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.

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

Configuring an external PostgreSQL instance

Black Duck supports using an external PostgreSQL instance.

To configure an external PostgreSQL database:

1. Initialize the external PostgreSQL cluster with the "C" locale. The method to accomplish this depends on what your external PostgreSQL provider allows you to do. For example, when using the PostgreSQL initdb tool, run the following command:

   ```
   initdb --locale=C -D /path/to/data
   ```

   When using other tools, an equivalent alternative if a locale setting is not available is specifying the SQL_ASCII character encoding.

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. Determine whether your external PostgreSQL account allows you to run the ALTER SYSTEM command.

   a. If you *can* run the ALTER SYSTEM command:

       i. Edit the `external-postgres-init.pgsql` script, located in the `docker-compose` directory, and remove the exit command (`\q`).

          This command is located before the `ALTER SYSTEM SET...` section.

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

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

  b. If you *cannot* run the ALTER SYSTEM command:

       i. Run the `external-postgres-init.pgsql` script, located in the `docker-compose` directory, to create users and databases. For example:

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

      ii. Use your preferred PostgreSQL administration tool to configure the PostgreSQL configuration parameters that the `external-postgres-init.pgsql` script defines.

          These parameters are located in the `ALTER SYSTEM SET...` section of the script.

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

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

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

Creating and configuring accounts using custom usernames and passwords

Use these instructions to create custom database usernames.

In these instructions:

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

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

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

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

    No other specific values are required.

2. Edit the `external-postgres-init.pgsql` script, located in the `docker-compose` directory with the account names you wish to use for **DBAdminName**, **DBUserName**, and **DBReporterName**.

3. Determine whether your external PostgreSQL account allows you to run the ALTER SYSTEM

command.

    a. If you *can* run the ALTER SYSTEM command:

        i. Edit the `external-postgres-init.pgsql` script, located in the `docker-compose` directory, and remove the exit command (`\q`).

           This command is located before the `ALTER SYSTEM SET...` section.

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

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

    b. If you *cannot* run the ALTER SYSTEM command:

        i. Run the `external-postgres-init.pgsql` script, located in the `docker-compose` directory, to create users and databases. For example:

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

        ii. Use your preferred PostgreSQL administration tool to configure the PostgreSQL configuration parameters that the `external-postgres-init.pgsql` script defines.

           These parameters are located in the `ALTER SYSTEM SET...` section of the script.

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 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-compose` directory. For each service (authentication, webapp, jobrunner, and scan), do the following:

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

      ii. Add the volume mount to the service.

   This example adds the volume to the webapp service:

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

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

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 containers by editing the `docker-compose.local-overrides.yml` file. See step 2c as an example.

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.

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-compose` 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. In the `bdio` database, apply the following permission changes. In the following example, `NewName1` is the username that replaces **blackduck_user**.

```
GRANT USAGE ON SCHEMA gui_schema TO NewName1;
GRANT SELECT, INSERT, UPDATE, TRUNCATE, DELETE, REFERENCES ON ALL TABLES
IN SCHEMA gui_schema TO NewName1;
GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA gui_schema to NewName1;
ALTER DEFAULT PRIVILEGES IN SCHEMA gui_schema GRANT SELECT, INSERT,
UPDATE, TRUNCATE, DELETE, REFERENCES ON TABLES TO NewName1;
ALTER DEFAULT PRIVILEGES IN SCHEMA gui_schema GRANT ALL PRIVILEGES ON
SEQUENCES TO NewName1;
GRANT USAGE ON SCHEMA sqlg_schema TO NewName1;
GRANT SELECT, INSERT, UPDATE, TRUNCATE, DELETE, REFERENCES ON ALL TABLES
IN SCHEMA sqlg_schema TO NewName1;
GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA sqlg_schema to NewName1;
ALTER DEFAULT PRIVILEGES IN SCHEMA sqlg_schema GRANT SELECT, INSERT,
UPDATE, TRUNCATE, DELETE, REFERENCES ON TABLES TO NewName1;
ALTER DEFAULT PRIVILEGES IN SCHEMA sqlg_schema GRANT ALL PRIVILEGES ON
```

```
SEQUENCES TO NewName1;
alter function first_agg owner to NewName1 ;
alter function first owner to NewName1 ;
```

5. Restart Black Duck.

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:

- Authentication
- Registration
- Job Runner
- Web app
- Scan

Proxy environment variables are:

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

The environment variables for NTLM proxies are:

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

Proxy password

The following services require the proxy password:

- authentication
- webapp
- registration
- jobrunner
- scan

There are two 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.

  Use the `docker-compose.local-overrides.yml` file located in the `docker-compose` directory.

For each service (authentication, webapp, registration, jobrunner, and scan), do the following:

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

    b. Add the volume mount to the service.

For example, suppose the HUB_PROXY_PASSWORD_FILE is located in the `opt/local/proxy` host directory. This example adds the volume to the webapp service:

```
webapp:
 volumes: ['opt/local/proxy:run/secrets']
```

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

■ Specify an environment variable called HUB_PROXY_PASSWORD that contains the proxy password.

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.

Importing a proxy certificate

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

1. Mount a directory that contains a text file called HUB_PROXY_CERT_FILE to `/run/secrets`.

2. in the `docker-compose.local-overrides.yml` file located in the `docker-compose` directory, add the secret by adding a volume mount string to the authentication, webapp, registration, jobrunner, and scan services.

    This example adds the secret to the webapp service:

```
webapp:
 volumes: ['/directory/where/the/cert-folder/is:/run/secrets']
```

    Be sure that you remove the comment character (#) before the service name.

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-compose/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 jobrunner, scan, and binaryscanner containers

The jobrunner, scan, and binaryscanner containers can be scaled.

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

Scaling jobrunner containers

This example adds a second job runner container:

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

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

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

Scaling scan containers

This example adds a second scan container:

- `docker-compose -p hub scale scan=2`

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

- `docker-compose -p hub scale scan=1`

Scaling binaryscanner containers

Binaryscaner containers are used with Black Duck - Binary Analysis.

This example adds a second binaryscanner container:

- `docker-compose -p hub scale binaryscanner=2`

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

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

- `docker-compose -p hub scale binaryscanner=1`

Configuring SAML for Single Sign-On

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

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

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

For additional SAML information:

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

Note the following:

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

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

- When logging in with SAML enabled, you are re-directed to your identity provider's login page, not Black Duck's login page.

- When SSO users log out of Black Duck, a logout page now appears notifying them that they successfully logged out of Black Duck. This logout page includes a link to log back into Black Duck; users may not need to provide their credentials to successfully log back in to Black Duck.

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

 To enable single sign-on using SAML

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

   The Administration page appears.

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

3. In the **SAML Configuration Details** settings, complete the following:

    a. Select the **Enable SAML** check box.

    b. Optionally, select the **Enable Group Synchronization** check box. If this option is enabled, upon login, groups from the Identity Provider (IDP) are created in Black Duck and users will be assigned to those groups. Note that you must configure IDP to send groups in attribute statements with the attribute name of 'Groups'.

    c. Optionally, select the **Enable Local Logout Support** check box. If this option is enabled, after logging out of Black Duck, the IDP's login page would appear.

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

    d. **Service Provider Entity ID** field. Enter the information for the Black Duck server in your environment in the format **https://**_host_ where _host_ is your Black Duck server.

    e. **Identity Provider Metadata**. Select one of the following:
        - **URL** and enter the URL for your identity provider.
        - **XML File** and either drop the file or click in the area shown to open a dialog box from which you can select the XML file.

    f. **External Black Duck Url** field. The URL of the public URL of the Black Duck server.

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

4. Click **Save**.

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

To disable single sign-on using SAML

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

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

3. In the **SAML Configuration Details** settings, clear the **Enable SAML** check box.

4. Click **Save**.

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.

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

1. Administrators must enable the upload of source files.

   a. The administrator enables the feature using an environment variable and provides a Docker-related secret consisting of a "user key" known as a seal key.

   b. Black Duck retrieves the seal key and generates a "master key" which is used for encrypting and decrypting the source files.

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

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

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

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

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

Note the following:

- Ensure that you have enough disk space for file uploads.

- The maximum total source size that you can upload at one time is 4 GB (4000 MB). This value is configurable.

- Uploaded files are deleted after 180 days. This value is configurable.

- Files are deleted when the upload service reaches 95% of the maximum disk setting.

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

To enable file upload

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

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

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

   ```
   ENABLE_SOURCE_UPLOADS=true
   ```

3. Mount a directory that contains the SEAL KEY to `/run/secrets` by adding the volume section to the uploadcache service in the `docker-compose.local-overrides.yml` file located in the `docker-compose` directory. If necessary, remove the comment character (#) before the `uploadcache` service:

   ```
   uploadcache:
   volumes: ['/directory/where/the/seal-key-file/is:/run/secrets']
   ```

   In our example:

   ```
   uploadcache:
   volumes: ['opt/secrets:/run/secrets']
   ```

4. [Restart the containers](#).

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

Backing up the seal key and master key

Back up the seal key and master key. If these keys are not backed up, there may be a loss of data if the seal key is lost.

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

Replacing the seal key

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

To replace the seal key, use the following script:

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

Additional configuration options

Black Duck provides these additional configurations:

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

  Use the DATA_RETENTION_IN_DAYS environment variable to change this setting.

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

  Use the MAX_TOTAL_SOURCE_SIZE_MB environment variable to change this setting.

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

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

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

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

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

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

Starting or stopping Black Duck

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

Starting up Black Duck

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

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

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

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

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

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

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

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

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

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

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

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

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

```
docker-compose -f docker-compose.yml -f docker-compose.bdba.yml -f docker-
compose.local-overrides.yml -p hub up -d
```

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

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

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

Shutting down Black Duck

- Run the following command to shut down Black Duck with the PostgreSQL database container:

```
docker-compose -f docker-compose.yml -p hub down
```

- Run the following command to shut down Black Duck with an external database:

```
docker-compose -f docker-compose.externaldb.yml -p hub down
```

- Run the following command to shut down Black Duck with Black Duck - Binary Analysis and with the
PostgreSQL database container:

```
docker-compose -f docker-compose.yml -f docker-compose.bdba.yml -p hub
down
```

- Run the following command to shut down Black Duck with Black Duck - Binary Analysis and with an
external database:

```
docker-compose -f docker-compose.externaldb.yml -f docker-compose.bdba.yml
-p hub down
```

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-compose/bin` directory:

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

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

Understanding the default sysadmin user

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

> **Tip:** As a best practice, you should use the default sysadmin account for your initial log in and then immediately change the default password–blackduck–so that the server is secure. To change your password, select **My Profile** from your username/user profile icon in the upper right corner of the Black Duck UI.

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-compose` 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.

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 provides variables in the `blackduck-config.env` file that you can use to change the default usage for similar match types. The variables are:

■ BLACKDUCK_HUB_FILE_USAGE_DEFAULT. Defining a usage for this variable sets the default value for the following match types:

- Binary
- Exact Directory
- Exact File
- Files Added/Deleted
- Files Modified
- Partial

■ BLACKDUCK_HUB_DEPENDENCY_USAGE_DEFAULT. Defining a usage for this variable sets the default value for the following match types:

- File Dependency
- Direct Dependency
- Transitive Dependency

■ BLACKDUCK_HUB_SOURCE_USAGE_DEFAULT. Defining a usage for this variable sets the default value for the following match types:

- Snippet

■ BLACKDUCK_HUB_MANUAL_USAGE_DEFAULT. Defining a usage for this variable sets the default value for the following match types:

- Manually Added
- Manually Identified

To configure different usage values

1. Edit the `blackduck-config.env` file located in the `docker-compose` 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,

   You can edit one or more variables.

   For example, to change default usage for files to statically linked:

   `BLACKDUCK_HUB_FILE_USAGE_DEFAULT=STATICALLY_LINKED`

   > **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.

2. Restart the containers.

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 (appcheck-worker): 0
- CA (blackduck-cfssl): 100 user:
- DB (blackduck-postgres): 70
- Documentation (blackduck-documentation): 8080
- Job Runner (blackduck-jobrunner): 100
- Logstash (blackduck-logstash): 100
- RabbitMQ (rabbitmq): 100
- Registration (blackduck-registration): 8080
- Scan (blackduck-scan): 8080
- Webapp (blackduck-webapp): 8080
- webserver (blackduck-nginx): 100
- Uploadcache (blackduck-uploadcache): 100
- Zookeeper (blackduck-zookeeper): 1000 user

Changing the UID consists of adding the new value for a container to the `docker-compose.local-overrides.yml` located in the `docker-compose` 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 and the binaryscanner container *cannot* be changed.

    - For the postgres container, the UID must equal 70.

    - For the binaryscanner container, the UID must be 0 (root).

- Although some containers have the same UID value (for example, the documentation, registration, scan and webapp 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 webapp container from 8080 to 1001 does not change the value of the scan, documentation, or registration containers - the UID value for these three 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. [Restart](#) 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-compose` directory.

   Use the `webserver` section and if necessary remove the comment character. Add the port information in the format: `ports: ['443:`*`NewValue`*`']` For example to change the port to 8443:

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

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

   ```
   PUBLIC_HUB_WEBSERVER_PORT=8443
   ```

Disabling IPv6

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

Follow these instructions to uninstall Black Duck.

To uninstall Black Duck, do one of the following:

- Stop and remove the containers and remove the volumes.

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

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

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

Note that the PostgreSQL database is not backed up. Use these instructions to [back up the database](#).

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

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

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

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

Installation files

The installation files are available on GitHub.

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

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

Download from the GitHub page

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

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

   ```
   gunzip hub-2019.6.1.tar.gz
   ```

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

   ```
   tar xvf hub-2019.6.1.tar
   ```

Download using the wget command

1. Run the following command:

   ```
   wget https://github.com/blackducksoftware/hub/archive/v2019.6.1.tar.gz
   ```

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

   ```
   gunzip v2019.6.1.tar.gz
   ```

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

   ```
   tar xvf v2019.6.1.tar
   ```

Upgrading from the AppMgr architecture

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

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

Upgrading to the multi-container Docker architecture consists of:

1. Migrating your PostgreSQL database.

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

2. Upgrading Black Duck.

Migrating your PostgreSQL database

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

1. Backing up the original PostgreSQL database.

2. Restoring the data.

To back up the original PostgreSQL database

1. Log in to Black Duck server as the **blckdck** user.

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

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

   ```
   export PATH=$PATH:/opt/blackduck/hub/postgresql/bin
   export PGPORT=5432
   pg_dump -Fc -f /tmp/bds_hub.dump bds_hub
   ```

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

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

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

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

To restore the PostgreSQL data

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

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

2. After the DB container has started, run the migration script located in the `docker-compose` directory. This script restores the data from the existing database dump file.

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

After stopping the containers, you can upgrade to the multi-image Docker version of Black Duck.

Error messages

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

"ERROR: role "blckdck" does not exist"

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

WARNING: errors ignored on restore: 7

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

Upgrading Black Duck

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

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

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

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

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

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

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

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

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

- `docker-compose -f docker-compose.yml -p hub up -d` (Using the PostgreSQL database container)
- `docker-compose -f docker-compose.externaldb.yml -p hub up -d` (Using an external database)
- `docker-compose -f docker-compose.yml -f docker-compose.bdba.yml -p hub up -d` (Using the PostgreSQL database container with Black Duck - Binary Analysis)
- `docker-compose -f docker-compose.externaldb.yml -f docker-compose.bdba.yml -p hub up -d` (Using an external PostgreSQL instance with Black Duck - Binary Analysis)

Upgrading from a single-container AppMgr Black Duck

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

Upgrading to the multi-container Docker architecture consists of:

1. Migrating your PostgreSQL database.

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

2. Upgrading Black Duck.

Migrating your PostgreSQL database

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

1. Backing up the original PostgreSQL database.

2. Restoring the data.

To back up the PostgreSQL database

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

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

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

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

To restore the PostgreSQL data

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

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

2. After the DB container has started, run the migration script located in the docker-compose directory. This script restores the data from the existing database dump file.

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

After stopping the containers, you can upgrade to the multi-image Docker version of Black Duck.

Error messages

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

   "ERROR: role "blckdck" does not exist"

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

   WARNING: errors ignored on restore: 7

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

Upgrading Black Duck

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

   - `docker-compose -f docker-compose.yml -p hub up -d` (Using the PostgreSQL database container)
   - `docker-compose -f docker-compose.externaldb.yml -p hub up -d` (Using an external database)
   - `docker-compose -f docker-compose.yml -f docker-compose.bdba.yml -p hub up -d` (Using the PostgreSQL database container with Black Duck - Binary Analysis)
   - `docker-compose -f docker-compose.externaldb.yml -f docker-compose.bdba.yml -p hub up -d` (Using an external PostgreSQL instance with Black Duck - Binary Analysis)

Upgrading from an existing Docker architecture

Upgrading from a previous version of Black Duck consists of:

1. Migrating your PostgreSQL databases.

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

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

If you are upgrading from version 4.2.0 or higher, then migrating your databases is optional.

2. Upgrading Black Duck.

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

Migrating your PostgreSQL databases

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

1. Backing up the PostgreSQL databases: the Black Duck database (`bds_hub`), the reporting database (`bds_hub_report`), and the scan data database (`bdio`).

2. Bringing down Black Duck containers.

3. Restoring the data.

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

To back up the PostgreSQL databases that are automatically installed with Black Duck

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

> **Important:** You *must* run the 2019.6.1 version of the `hub_create_data_dump.sh` script located in the `docker-compose/bin` directory.

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

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

To bring down Black Duck containers

Run one of the following commands to bring down Black Duck containers that have the previous version of Black Duck running:

- Run the following command to shut down Black Duck with the PostgreSQL database container:

  ```
  docker-compose -f docker-compose.yml -p hub down
  ```

- Run the following command to shut down Black Duck with an external database:

  ```
  docker-compose -f docker-compose.externaldb.yml -p hub down
  ```

- Run the following command to shut down Black Duck with Black Duck - Binary Analysis and with the PostgreSQL database container:

  ```
  docker-compose -f docker-compose.yml -f docker-compose.bdba.yml -p hub down
  ```

- Run the following command to shut down Black Duck with Black Duck - Binary Analysis and with an external database:

  ```
  docker-compose -f docker-compose.externaldb.yml -f docker-compose.bdba.yml -p hub down
  ```

To restore the PostgreSQL data

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

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

2. After the DB container has started, run the migration script which restores the data from the existing database data files.

   > **Important:** You *must* run the 2019.6.1 version of the `hub_db_migrate.sh` script located in the `docker-compose/bin` directory.

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

You can now upgrade Black Duck.

Upgrading Black Duck

To upgrade Black Duck:

1. Stop the existing containers:

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

2. 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-compose` directory in the newer version of Black Duck:
     - `docker-compose -f docker-compose.yml -p hub up -d` (Using the PostgreSQL database container)

- ○ `docker-compose -f docker-compose.externaldb.yml -p hub up -d` (Using an external database)

  You will need to redo any modifications you made to the `.yml` file.

- If you are upgrading Black Duck - Binary Analysis, run one of the following commands, using the files located in the `docker-compose` directory in the newer version of Black Duck:
    - ○ `docker-compose -f docker-compose.yml -f docker-compose.bdba.yml -p hub up -d` (Using the PostgreSQL database container with Black Duck - Binary Analysis
    - ○ `docker-compose -f docker-compose.externaldb.yml -f docker-compose.bdba.yml -p hub up -d` (Using an external PostgreSQL instance 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 `docker-compose.local-overrides.yml` file that contains your modifications; for all other files, use the files located in the `docker-compose` directory in the newer version of Black Duck:
    - ○ `docker-compose -f docker-compose.yml -f docker-compose.local-overrides.yml -p hub up -d` (Using the PostgreSQL database container)
    - ○ `docker-compose -f docker-compose.externaldb.yml -f docker-compose.local-overrides.yml -p hub up -d` (Using an external database)
    - ○ `docker-compose -f docker-compose.yml -f docker-compose.bdba.yml -f docker-compose.local-overrides.yml -p hub up -d` (Using the PostgreSQL database container with Black Duck - Binary Analysis
    - ○ `docker-compose -f docker-compose.externaldb.yml -f docker-compose.bdba.yml -f docker-compose.local-overrides.yml -p hub up -d` (Using an external PostgreSQL instance with Black Duck - Binary Analysis

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

  You will *not* need to redo any modifications you made to the `.yml` file.

> **Note:** If you are upgrading from a version prior to 2018.11.x and you previously edited the `hub-proxy.env` file, those edits must be copied over to the `blackduck-config.env` file.

3. If you are upgrading from a pre-2018.12.1 version of Black Duck and are using an external PostgreSQL database, do the following:

   a. Using your preferred PostgreSQL administration tool, make these global system changes:

   ```
   autovacuum_max_workers = 20
   autovacuum_vacuum_cost_limit = 2000
   ```

   b. Restart PostgreSQL.

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

1. Authentication

2. CA

3. DB

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

4. Documentation

5. Jobrunner

6. Logstash

7. Registration

8. Scan

9. Uploadcache - This container is also used for Black Duck - Binary Analysis.

10. Webapp

11. Webserver

12. Zookeeper

If Black Duck - Binary Analysis is enabled, there are these additional containers:

1. Binaryscanner

2. RabbitMQ

The following diagram shows the basic relationships among the containers and which ports are exposed outside of the Docker network.
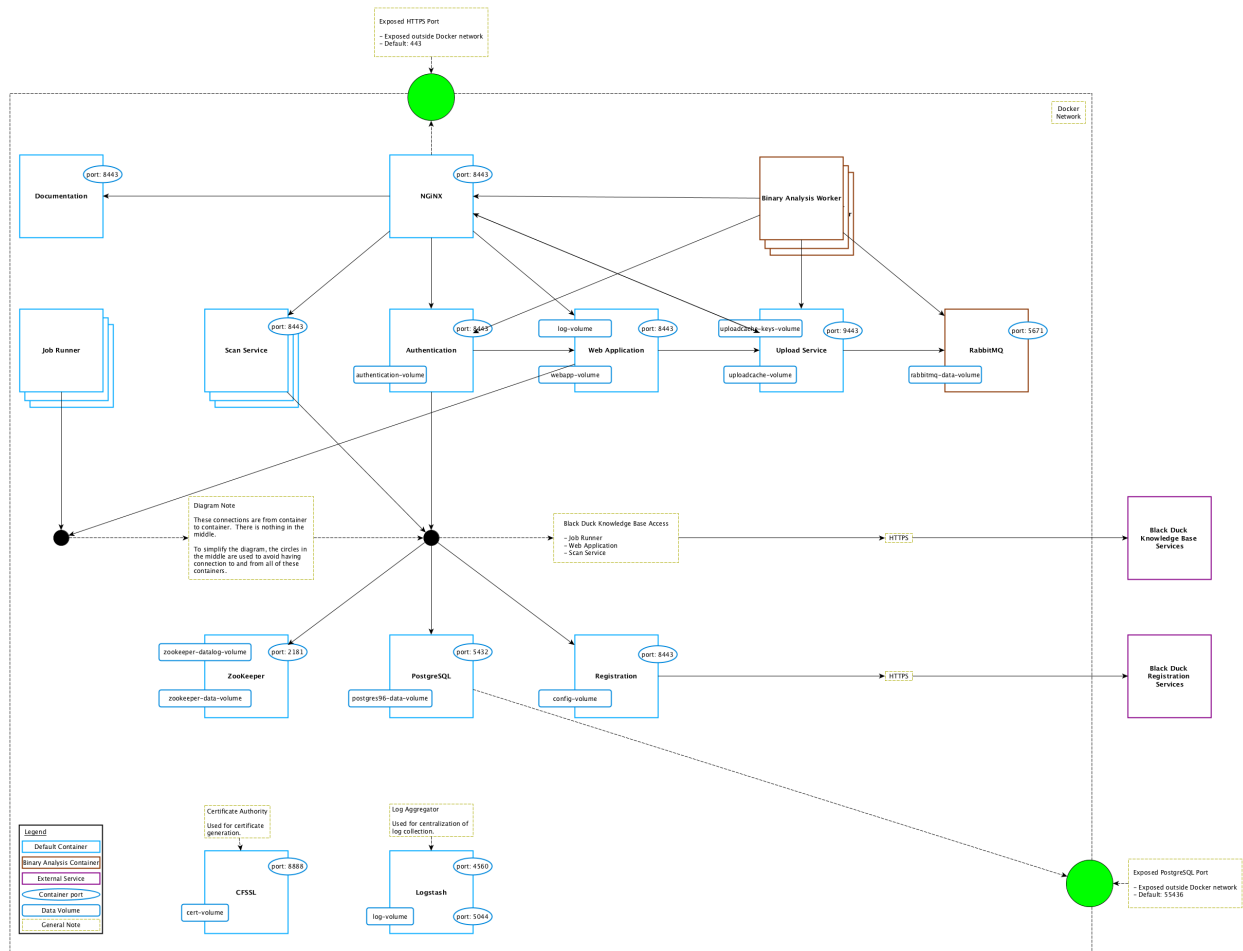
The following tables provide more information on each container.

Authentication container

Container Name: blackduck-authentication	
Image Name	blackducksoftware/blackduck-authentication:2019.6.1
Description	The authentication service is the container that all authentication-related requests are made against.
Scalability	There should only be a single instance of this container. It currently cannot be scaled.

Container Name: blackduck-authentication	
Links/Ports	Nothing external (8443 internally). This container will need to connect to these other containers/services: • postgres • cfssl • logstash • registration • zookeeper • webapp 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 Compose or Docker Swarm use. 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 • zookeeper - $HUB_ZOOKEEPER_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`

CA container

Container Name: blackduck-cfssl	
Image Name	blackducksoftware/blackduck-cfssl:1.0.0
Description	This container uses CFSSL which is used for certificate generation for PostgreSQL, NGiNX, and clients that need to authenticate to Postgres. This container is also used to generate TLS certificates for the internal containers that make up the application.
Scalability	There should only be a single instance of this container. It should not be scaled.
Links/Ports	The container needs to expose port 8888 within the Docker network to other containers/services that link to it.
Resources/Constraints	- 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:1.0.7
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.
Alternate Host Name Environment Variables	There are times when running in other types of orchestrations that it is useful to have host names set for these containers that are not the default that Docker Compose or Docker Swarm use. These environment variables can be set to override the default host names: • logstash: $HUB_LOGSTASH_HOST • cfssl: $HUB_CFSSL_HOST
Resource/Constraints	• Default max Java heap size: N/A • Container memory: 3GB • Container CPU: 1 CPU
Users/Groups	This container runs as UID 70. If the container is started as UID 0 (root) then the user will be switched to UID 70:root before executing its main process.

Container Name: blackduck-postgres	
	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:2019.6.1
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 Compose or Docker Swarm use. These environment variables can be set to override the default host names: • logstash: $HUB_LOGSTASH_HOST • cfssl: $HUB_CFSSL_HOST
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`

Jobrunner container

Container Name: blackduck-Jobrunner	
Image Name	blackducksoftware/blackduck-jobrunner:2019.6.1
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 • zookeeper • 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 • zookeeper: $HUB_ZOOKEEPER_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:1.0.4
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. 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`

Registration container

Container Name: blackduck-registration	
Image Name	blackducksoftware/blackduck-registration:2019.6.1
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.
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

Container Name: blackduck-registration	
	Compose or Docker Swarm use. These environment variables can be set to override the default host names: • logstash: $HUB_LOGSTASH_HOST • cfssl: $HUB_CFSSL_HOST
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`

Scan container

Container Name: blackduck-scan	
Image Name	blackducksoftware/blackduck-scan:2019.6.1
Description	The scan service is the container that all scan data requests are made against.
Scalability	This container can be scaled.
Links/Ports	This container needs to connect to these containers/services: • postgres • zookeeper • 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 Compose or Docker Swarm use. These environment variables can be set to override the default host names: • postgres: $HUB_POSTGRES_HOST • zookeeper: $HUB_ZOOKEEPER_HOST

Container Name: blackduck-scan	
	• 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`

Uploadcache container

Container Name: Uploadcache	
Image Name	blackducksoftware/blackduck-upload-cache:1.0.8
Description	This container will be used to temporarily store uploads for binary analysis and when uploading source files for snippet matching.
Scalability	There should only be a single instance of this container. It should not be scaled.
Links/Ports	This container needs to connect to these containers/services: • cfssl • logstash • rabbitmq (if Black Duck - Binary Analysis is enabled) The container exposes ports 9443 and 9444 to other containers that link to it.
Alternate Host Name Environment Variables	There are times when running in other types of orchestrations that it is useful to have host names set for these containers that are not the default that Docker Compose or Docker Swarm use. These environment variables can be set to override the default host names: • cfssl: $HUB_CFSSL_HOST • logstash: $HUB_LOGSTASH_HOST • rabbitmq: $RABBIT_MQ_HOST

Container Name: Uploadcache	
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-bdba.env`
	
	`blackduck-config.env`

Webapp container

Container Name: blackduck-webapp	
Image Name	blackducksoftware/blackduck-webapp:2019.6.1
Description	The webapp container is the container that all Web/UI/API requests are made against. It also processes any UI requests. In the diagram, the ports for the webapp are not exposed outside of the Docker network. There is an NGiNX reverse proxy (as described in the WebServer container) that is exposed outside of the Docker network instead.
Scalability	There should only be a single instance of this container. It should not be scaled.
Links/Ports	The webapp container needs to connect to these containers/services: - postgres - zookeeper - 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 Compose or Docker Swarm use. These environment variables can be set to override the default host names: postgres: $HUB_POSTGRES_HOST

Container Name: blackduck-webapp	
	• zookeeper: $HUB_ZOOKEEPER_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:1.0.7
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.
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

• upload cache (if Black Duck - Binary Analysis is enabled)

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 |

Container Name: blackduck-webserver	
	these containers that are not the default that Docker Compose or Docker Swarm use. These environment variables can be set to override the default host names:
	- webapp: $HUB_WEBAPP_HOST
	- cfssl: $HUB_CFSSL_HOST
	- scan: $HUB_SCAN_HOST
	- documentation: $HUB_DOC_HOST
	- authentication: $HUB_AUTHENTICATION_HOST
	- upload cache: $HUB_UPLOAD_CACHE_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`

ZooKeeper container

Container Name: Zookeeper	
Image Name	blackducksoftware/blackduck-zookeeper:1.0.0
Description	This container stores data for the other containers. It exposes ports within the Docker network, but not outside the Docker network.
Scalability	This container should not be scaled.
Links/Ports	The Zookeeper container needs to connect to this container/service: - logstash The container needs to expose port 2181 within the Docker network to other containers that will link to it.
Alternate Host Name Environment Variables	There are times when running in other types of orchestrations that it is useful to have host names set for these containers that are not the default that Docker Compose or Docker Swarm use. These environment variables can be set to override the default host names:

Container Name: Zookeeper	
	• logstash: $HUB_LOGSTASH_HOST
Constraints	• Default max Java heap size: 256MB
	• Container memory: 384MB
	• Container CPU: Unspecified
Users/Groups	This container runs as UID 1000. If the container is started as UID 0 (root) then the user will be switched to UID 1000:root before executing its main process. 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`

Black Duck - Binary Analysis containers

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

Binaryscanner container

Container Name: appcheck-worker	
Image Name	blackducksoftware/appcheck-worker:2019.03
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 Compose or Docker Swarm use. These environment variables can be set to override the default host names: • cfssl: $HUB_CFSSL_HOST

Container Name: appcheck-worker	
	• logstash: $HUB_LOGSTASH_HOST • rabbitmq: $RABBIT_MQ_HOST • webserver: $HUB_WEBSERVER_HOST
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`

Rabbitmq container

Container Name: rabbitmq	
Image Name	blackducksoftware/rabbitmq:1.0.0
Description	This container facilitates upload information to the binary analysis worker. It exposes ports within the Docker network, but not outside the Docker network. This container is currently only used if Black Duck - Binary Analysis is enabled.
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 Compose or Docker Swarm use. 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.

Container Name: rabbitmq	
	This container is also able to be started as a random UID as long as it is also started within the root group (GID/fsGroup 0).
Environment File	`hub-bdba.env`