**BLACK**DUCK | Hub

# Installing the Hub using OpenShift

Version 4.4.3

This edition of the *Installing the Hub using OpenShift* refers to version 4.4.3 of the Black Duck Hub.

This document created or updated on Friday, March 16, 2018.

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

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

# Contents

# The Hub documentation

The documentation for the Hub consists of online help and these documents:

| Title | File | Description |
|---|---|---|
| Release Notes | release_notes_bd_hub.pdf | Contains information about the new and improved features, resolved issues, and known issues in the current and previous releases. |
| Installing Hub using Docker Compose | hub_install_compose.pdf | Contains information about installing and upgrading the Hub using Docker Compose. |
| Installing Hub using Docker Swarm | hub_install_swarm.pdf | Contains information about installing and upgrading the Hub using Docker Swarm. |
| Installing Hub using Kubernetes | hub_install_kubernetes.pdf | Contains information about installing and upgrading the Hub using Kubernetes. |
| Installing Hub using OpenShift | hub_install_openshift.pdf | Contains information about installing and upgrading the Hub using OpenShift. |
| Getting Started | hub_getting_started.pdf | Provides first-time users with information on using the Hub. |
| Scanning Best Practices | hub_scanning_best_practices.pdf | Provides best practices for scanning. |
| Getting Started with the Hub SDK | getting_started_hub_sdk.pdf | Contains overview information and a sample use case. |
| Report Database | report_db_bd_hub.pdf | Contains information on using the report database. |

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

# Training

Black Duck Academy is a one-stop resource for all your Black Duck education needs. It provides you with 24x7 access to online training courses and how-to videos.

New videos and courses are added monthly.

At Black Duck Academy, you can:

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

Learn more at https://www.blackducksoftware.com/services/training

View the full catalog of courses and try some free courses at https://academy.blackducksoftware.com

When you are ready to learn, log in or sign up for an account: https://academy.blackducksoftware.com

# Customer Success Community

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

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

Access the Customer Success Community. If you do not have an account or have trouble accessing the system, please send an email to communityfeedback@blackducksoftware.com or call us at +1 781.891.5100 ext. 5.

To see all the ways you can interact with Black Duck Support, visit: https://www.blackducksoftware.com/support/contact-support.

OpenShift™ is an orchestration tool from Red Hat used for managing cloud workloads through containers.

This document provides instructions for installing Black Duck Hub on OpenShift.

## Hub Architecture

The Black Duck Hub is deployed as a set of Docker containers so that third-party orchestration tools such as OpenShift can be leveraged to manage individual Hub services.

This architecture brings these significant improvements to the Hub over monolithic deployments:

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

See containers for more information on the Docker containers that comprise the Hub application.

Visit the Red Hat OpenShift website: https://www.openshift.com for more information.

## Components hosted on Black Duck servers

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

- Registration server: Used to validate the Hub 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 the Hub can display the most up-to-date information about open source software (OSS) without requiring regular updates to your Hub.

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

# Getting started

The process for installing the Hub depends on whether you are installing the Hub for the first time or upgrading from a previous version of the Hub.

## New installations

For new installation of the Hub:

1. Read this planning chapter to review all requirements.

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

3. Review Chapter 4 for any administrative tasks.

## Upgrading from a previous version of the Hub

1. Read this planning chapter to review all requirements,

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

3. Review Chapter 4 for any administrative tasks.

# Hardware requirements

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

- 5 CPU cores
- 20 GB RAM
- 250 GB of free disk space for the database and other Hub containers
- Commensurate space for database backups

Each container description provides the container's requirements, including if running on a different machine or if more than one instance of a container will be running (currently only supported for the Job Runner and Scan 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.

To avoid underlying hardware resource exhaustion by the Hub, ensure that your OpenShift system administrator has put enterprise-level metrics and logging in place to identify unhealthy nodes on the cluster.

# OpenShift requirements

The Hub supports OpenShift Enterprise 3.5, 3.6, and 3.7 on Amazon Web Services (AWS) EC2. The following restriction applies when using Hub on OpenShift:

- The hub-webapp and the hub-logstash services must run on the same pod for proper log integration.

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

# Operating systems

The Dockerized Hub is supported on any OpenShift cluster that passes the standards for OpenShift cluster Conformance. (Click here for more on OpenShift and Kubernetes conformance.) Platforms that support OpenShift include, but are not limited to:

- 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

Windows operating system is currently not supported.

# Software requirements

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

- Chrome 63.0.3239.84 (Official Build) (64-bit)
- Firefox 57.0.1
- Internet Explorer 11.0.9600.18816
- Microsoft Edge 40.15063.674.0
- Microsoft EdgeHTML 15.15063
- Safari 11.0.1 (13604.3.5)

Note that the Hub does not support compatibility mode.

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

# Network requirements

The Hub requires the following ports to be externally accessible:

- Port 443 – Web server HTTPS port for the Hub via NGiNX
- Port 55436 – Read-only database port from PostgreSQL for reporting (or an equivalent exposable port for PostgreSQL read-only)

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

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

> **Note:** If you are using a network proxy, these URLs must be configured as destinations in your proxy configuration. Network proxy information can be expressed as environment variables by placing them in the openshift.yml file's ConfigMap called hub-config. See Environment Variables for more information on using environment variables with OpenShift.

# Database requirements

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

> **Important:** Black Duck Hub on OpenShift supports only an external (for example, Amazon RDS Relational Database Service) PostgreSQL instance. If you would like to run PostgreSQL on an internal OpenShift node, contact your authorized Black Duck support representative for assistance.

⚙ **To use an external PostgreSQL instance:**

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

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

2. Configure your database connection settings.

3. Install or upgrade the Hub.

## PostgreSQL versions

For the Hub version 4.4.3, you must use PostgreSQL version 9.6.x for compatibility with the Hub version 4.4.3. Refer to Chapter 6 for database migration instructions if upgrading from a pre-4.2.0 version of the Hub.

Understanding PostgreSQL's security configuration

PostgreSQL security is derived from CFSSL, which runs as a service inside your cluster.

For your Hub database to be secure, ensure that:

1. The namespace you are running PostgreSQL in is secure.

2. You have control over the users starting containers in that namespace.

3. The node which was labeled for PostgreSQL is protected from SSH by untrusted users.

# Proxy server requirements

The Hub supports:

- No Authentication
- Digest
- Basic
- NTLM

If you are going to make proxy requests to the Hub, work with the proxy server administrator to get the following required information:

- The protocol used by proxy server host (http or https).
- The name of the proxy server host
- The port on which the proxy server host is listening.

# Configuring your NGiNX server to work with the Hub

Given that OpenShift manages load balancing, there is no need to configure an NGiNX reverse proxy outside the OpenShift Container Platform router.

# Amazon services

You can:

- Install the Hub on Amazon Web Services (AWS)

    Refer to your AWS documentation for more information on AWS.

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

    Refer to your Amazon Relational Database Service documentation for more information on Amazon RDS.

    Currently the Hub requires PostgreSQL version 9.6.x.

    Click here for more information on configuring an external PostgreSQL server.

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

| Hub Installation Requirements |
| --- |

| **Hardware requirements** | |
| --- | --- |
| ☐ | You have ensured that your hardware meets the minimum hardware requirements. |

| **OpenShift requirements** | |
| --- | --- |
| ☐ | You have ensured that your system meets the OpenShift 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:<br><br>• Port 443 and port 55436 are externally accessible.<br>• The server has access to updates.suite.blackducksoftware.com which is used to validate the Hub license. |

| **Database requirements** | |
| --- | --- |
| ☐ | You have selected your database configuration.<br><br>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.<br><br>Configure proxy settings before or after installing the Hub. |

| **Web server requirements** | |
| --- | --- |
| ☐ | Configure web server settings before or after installing the Hub. |

Determine the method by which you want to install the Hub. After obtaining the orchestration files, as described in the next section, determine if a basic installation, as described here, would work for your

organization.

# 1. Obtaining the orchestration files

The installation files are available on Github (https://github.com/blackducksoftware/hub).

From your OpenShift bastion host (host with access to the Internet and the OpenShift cluster) with oc installed, 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 differs depending on how you access the file, the contents 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 Hub `.gz` file:

   ```
   gunzip hub-4.4.3.tar.gz
   ```

3. Unpack the Hub `.tar` file:

   ```
   tar xvf hub-4.4.3.tar
   ```

## Download using the wget command

1. Run the following command:

   ```
   wget https://github.com/blackducksoftware/hub/archive/v4.4.3.tar.gz
   ```

2. Uncompress the Hub `.gz` file:

   ```
   gunzip v4.4.3.tar.gz
   ```

3. Unpack the Hub `.tar` file:

   ```
   tar xvf v4.4.3.tar
   ```

## Distributions

The OpenShift directory has the following files you need to install or upgrade:

- `db.yml`: Creates a PostgreSQL service.
- `external-postgres-init.pgsql`: PostgresSQL.sql file used to configure an external PostgreSQL database.
- `openshift.yml`: Creates OpenShift deployments.

# 2. Creating a namespace

Create a virtual cluster, or namespace, (in OpenShift, also called a "project"), for running the Hub containers.

Any valid namespace will work, so long as it does not already exist on your cluster and you do not plan on running other applications in it: the namespace must be unique to the Hub, in order to ensure proper service resolution.

For example:

```
oc new-project my-ns
```

The namespace ensures that all containers, spanning multiple nodes, within the namespace have the same DNS, config maps, and so on.

> **Note:** If you later want to remove the Hub installation, you can do so with the following command:
> ```
> oc delete project my-ns
> ```

# 3. Installing the Hub containers

There are two steps to install the Hub services in OpenShift:

a.  Migrate any data from a previous/legacy non-OpenShift Hub.

b.  Install/run the remaining Hub containers.

## a. Migrating data from a previous version of the Hub

If you intend to migrate data from a previously-existing version of the Hub, follow these instructions for backing up and restoring Hub data. If not, this step can be skipped.

## b. Installing/Running the Hub containers

Now that the database has been set up, run the following command to create the full OpenShift deployment of the Hub:

```
oc create -f openshift.yml
```

### Verifying the Hub containers are running

After you have created the full OpenShift deployment of the Hub, confirm that the installation was successful by running the following command to see the running Hub containers:

```
oc get pods
```

> **Note:** Click here for the full list of containers in the OpenShift Hub.

If you suspect that the proper Hub Docker images cannot be pulled from the appropriate repository, debug this issue by looking at OpenShift events by running the following command:

```
oc get events
```

The output of this command could give hints on permissions or networking limitations that may impact pulling images.

# 4. Creating an OpenShift route

Immediately after the installation of the Hub in OpenShift, the IP addresses of the containers are internal (10.x), and are not visible/routable to the Internet. To make Hub services, such as the Web UI, externally available, you must create an OpenShift route.

Before beginning, make sure that your administrator has set up OpenShift routers. (This is a core feature of Red Hat OpenShift.) This will allow access to an OpenShift service through an exposed OpenShift route *from outside of your cluster*.

> **Note:** Another option for accessing the Hub without a route is creating a service using `--type=NodePort`. That would allow you to access the Hub by going to any machine running OpenShift DNS on your cluster. Contact either your Black Duck or Red Hat support representative for more information on this configuration.

⚙ **Creating the OpenShift route**

1. Create an OpenShift route pointing to the Black Duck Hub NGiNX service, with the sub path 'hub'.

2. The previous step creates a URL for the OpenShift route.

   If the URL for the route is *not* generated, set it in the OpenShift UI to `${external-ip}.xip.io`, where `external-ip` is the IP address of your load balancer. Consult Black Duck or Red Hat for information on how to find this address if you do not know it.

3. In the route, setup TLS to "passthrough".

# Connecting to the Hub

Once all of the containers for the Hub are up, the web application will be exposed on port 443 to the Docker host. Be sure that you have configured the hostname and then you can access the Hub.

Access the Hub, for example, at either:

- https://my-company-openshift/hub
- https://${external-ip}.xip.io/hub

For example, 128.100.200.300.xip.io, if you have a router running on an exposed IP at 128.100.200.300.

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

Enter the registration key provided to you to access the Hub.

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

# Basic Hub deployment

Although there are many ways to configure OpenShift, the easiest method is to initially install it with no

configuration settings and then, working with your administrator, configure the existing installation to meet your organization's needs.

The following instructions install the Hub using OpenShift with a PostgreSQL database inside the cluster and no other settings configured.

⚙ **To complete the basic Hub deployment**

1. cd into the `/openshift` directory then run the following command to create the PostgreSQL service:

   ```
   oc create -f db.yml
   ```

2. Determine the database pod ID by running the following command:

   ```
   oc get pods
   ```

3. After determining the database pod ID, run the following command to open a shell in the container:

   ```
   oc exec -t -i <pod ID> /bin/sh
   ```

   In the container, do the following:

   a. Run the following command:

      ```
      createuser blackduck
      ```

   b. Open the `vi` editor and create the following file:

      ```
      vi /tmp/init-db.sql
      ```

      Copy the contents of the `external-postgres-init.pgsql` file into this file.

      Add the following statements below the `CREATE USER blackduck_user;` line:

      - `ALTER USER blackduck_user WITH password 'blackduck';`
      - `ALTER USER blackduck WITH password 'blackduck';`

   c. Run the following command which initializes the PostgreSQL database:

      ```
      psql -a -f /tmp/init-db.sql
      ```

   d. Exit the container.

4. Edit the existing value of HUB_POSTGRES_HOST in the `openshift.yml` file to 'postgres'. This is the service name of the database pod in the `db.yml` file.

5. Run the following command to create the OpenShift deployment:

   ```
   oc create -f openshift.yml
   ```

6. Confirm that the installation was successful by running the following command to see the running Hub containers:

```
oc get pods
```

7.  Create an OpenShift route, as described [here](here).

This section describes these administrative tasks:

- [Understanding the default sysadmin user](#).
- [Configuring web server settings](#), such as configuring the [hostname](#), [host port](#), or [disabling IPv6](#).
- [Configuring proxy settings](#).
- [Configuring the Hub session timeout value](#).
- [Replacing the existing self-signed certificate for the Web Server with a custom certificate](#).
- [Scaling job runner and scan containers](#).
- [Configuring the report database password](#).
- [Providing access to the API documentation through a proxy server](#).
- [Providing access to the REST APIs from a non-Hub server](#).
- [Configuring secure LDAP](#).
- [Configuring Single Sign-On (SSO)](#).
- [Backing up Postgres volumes](#).

## Understanding the default sysadmin user

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

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

## Environment variables

There are several variables that can added to the openshift.yml's ConfigMap called hub-config. The variables that can be added are:

```
####################
# HUB PROXY SECTION #
####################
# HUB PROXY ENVIRONMENT VARIABLES
```

```
HUB_PROXY_HOST=

HUB_PROXY_PORT=

HUB_PROXY_SCHEME=

HUB_PROXY_USER=


# FOR NTLM PROXIES

HUB_PROXY_WORKSTATION=

HUB_PROXY_DOMAIN=


# THE PROXY PASSWORD CAN BE SPECIFIED HERE IF IT IS NOT SPECIFIED IN A

# SEPARATE MOUNTED FILE OR SECRET.

# HUB_PROXY_PASSWORD=


# HOSTS THAT WON'T GO THROUGH A PROXY.

HUB_PROXY_NON_PROXY_HOSTS=SOLR

#########################

# BLACKDUCK CORS SECTION #

#########################


# CROSS ORIGIN RESOURCE SHARING (CORS) CONFIGURATION

BLACKDUCK_HUB_CORS_ENABLED=

BLACKDUCK_CORS_ALLOWED_ORIGINS_PROP_NAME=

BLACKDUCK_CORS_ALLOWED_HEADERS_PROP_NAME=

BLACK_CORS_EXPOSED_HEADERS_PROP_NAME=


# TO ACCESS API DOCUMENTATION WHEN USING A REVERSE PROXY WITH HUB MOUNTED
UNDER A SUB-PATH

BLACKDUCK_SWAGGER_DISPLAYALL=


####################

# POSTGRES SECTION #

####################


HUB_POSTGRES_ENABLE_SSL="FALSE"

HUB_POSTGRES_HOST=

HUB_POSTGRES_PORT=
```

```
HUB_POSTGRES_USER=BLACKDUCK_USER

HUB_POSTGRES_ADMIN=BLACKDUCK


#########################
# HUB WEBSERVER SECTION #
#########################


# WEBSERVER [NGINX] ENVIRONMENT VARIABLES
# THE PUBLIC HUB WEBSERVER HOST WILL BE SERVED AS A SUBJECT ALTERNATIVE NAME
(SAN) WITHIN THE
# DEFAULT, SERVED CERTIFICATE. THIS VALUE SHOULD BE CHANGED TO THE PUBLICLY-
FACING
# HOSTNAME THAT USERS WILL ENTER IN THEIR BROWSER IN ORDER TO ACCESS HUB.

PUBLIC_HUB_WEBSERVER_HOST=LOCALHOST

# IF THE HOST PORT IS CHANGED, THE PUBLIC HUB WEBSERVER PORT VALUE SHOULD BE
EQUALLY CHANGED.

PUBLIC_HUB_WEBSERVER_PORT=443

# IF THE CONTAINER PORT IS CHANGED, THE HUB WEBSERVER PORT VALUE SHOULD BE
EQUALLY CHANGED.
HUB_WEBSERVER_PORT=8443

# IF IPV6 IS DISABLED FOR A HOST MACHINE, THIS FLAG SHOULD BE SET TO '1' SO
THAT NGINX
# WILL NOT LISTEN ON AN IPV6 ADDRESS. BY DEFAULT NGINX WILL LISTEN ON IPV4 AND
IPV6.

IPV4_ONLY=0


# public URL of the Hub server

HUB_SAML_EXTERNAL_URL

# Session timeout value in seconds.

HUB_WEBAPP_SESSION_TIMEOUT
```

Please reference the JobRunner replication controller (or any other replication controller) in the openshift.yml file for an example of how to inject arbitrary environment variables into a container.

# Web server settings

The following sections describe the required web server settings for a OpenShift environment.

## Host name modification

When the web server starts up, if it does not have certificates configured, a self-signed certificate is

generated. To ensure that the hostname on the self-signed certificate matches the hostname actually used to reach the web server, you must set the web server hostname. Otherwise, the certificate uses the service name as the hostname, and SSL handshake errors could result.

To inform the webserver of the hostname used to reach it, edit the hub-config object in the openshift.yml file to update the desired host name value.

PUBLIC_HUB_WEBSERVER_HOST=LOCALHOST value

## Port modification

In an OpenShift environment, it is common to leverage an OpenShift Route to forward network requests to nodes. In a Hub installation in OpenShift, this OpenShift Route will forward web traffic to the Hub's NGiNX proxy server, which sends traffic along to the Hub's webapp.

If you want to change either the port that external users use to connect to the web server (for example, a web browser connecting to the Hub's web UI), or, the port that the NGiNX proxy server listens on from the External Load Balancer (ELB), you need to edit the hub-config object in the openshift.yml file.

To change the publicly-exposed web server port, edit PUBLIC_HUB_WEBSERVER_PORT from its default value of 443.

To change the port that the NGiNX listens to from the ELB, edit HUB_WEBSERVER_PORT from its default value of 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 value in the HUB WEBSERVER SECTION in the hub-config object in the openshift.yml file to 1.

# Proxy settings

There are currently four services requiring access to services hosted by Black Duck Software:

- Registration
- Jobrunner
- Webapp
- Scan

If a proxy is required for external internet access, you must configure it.

⚙ **To configure the proxy for external internet access:**

1. Add the required parameters for your proxy setup to the hub-config object in the openshift.yml file.

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.

## Authenticated proxy password

There are two methods for specifying a proxy password when using OpenShift:

- Specify an environment variable called HUB_PROXY_PASSWORD that contains the proxy password.
- Add a OpenShift secret called HUB_PROXY_PASSWORD_FILE.

### Environment Variables

The easiest method is to specify an environment variable called HUB_PROXY_PASSWORD in the hub-config object in the openshift.yml file that contains the proxy password.

### OpenShift Secret

The most secure way to specify the proxy password is to add it as an OpenShift secret and to inject that secret into the pod.

To store the proxy password as a secret, place the password in a file (called hpp in the following example), then run the oc create secret command:

```
echo "mypassword1234" > hpp
oc create secret generic hub-proxy-password --from-file=./hpp
```

After running these commands, for safety reasons, delete the hpp file.

Now the that the secret has been created in OpenShift, you must expose the secret to the Hub services – Webapp, Registration, Jobrunner – that require access to it.

For example, the following text added to the openshift.yml file exposes the secret to Webapp:

In each of these pod specifications, add the secret injection next to the image that is using them.

For example, add the following text to the openshift.yml file you use to launch the cluster to expose the secret to the Webapp service:

```
image: hub-webapp:<version number>
env:
- name: HUB_PROXY_PASSWORD
  valueFrom:
    secretKeyRef:
      name: hub_proxy_password
      key: hpp
```

Add this section of text for the Registration, Jobrunner and Scan services by replacing `image: hub-`

`webapp:4.4.3` with `image: registration:4.4.3,` `image: jobrunner:4.4.3,` and `image: scan:4.4.3` respectively.

# Configuring the Hub session timeout

By default, the Hub session timeout value is 2 hours.

To specify a different value, use the HUB_WEBAPP_SESSION_TIMEOUT property to specify the new timeout value, expressed in number of seconds. For example, a timeout value of one hour would be 3600 seconds.

As with all other environment variables, there are two ways to set an environment variable.

- If you are familiar with how to inject environment variables into containers from config maps, you can set them in the openshift.yml file, and then use `envFrom` statements to inject them.

- Alternatively, if you are not familiar with config maps, it can be simpler to inject these environment variables directly into your pod stanza in the openshift.yml, in the following format:

   For example:

   ```
   image: hub-webapp:4.4.3
   env:
   - name: HUB_WEBAPP_SESSION_TIMEOUT
     value: 3600
   ```

# Configuring an external PostgreSQL instance

You must use an external PostgreSQL instance when installing the Hub on OpenShift. The external PostgreSQL instance needs to be initialized and information must be provided to the Webapp, Jobrunner, and Scan containers.

## PostgreSQL configuration

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

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

   No other specific values are required.

2. Run the `external-postgres-init.pgsql` script to create users, databases, and other necessary items. For example:

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

3. 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.

   To change the passwords, open a terminal to the database using the following command:

```
psql -U blackduck -h <hostname> -p <port> postgres
```

Then run the following three commands in the terminal to change the passwords for the three accounts:

```
ALTER ROLE blackduck WITH PASSWORD 'my_admin_password_here';

ALTER ROLE blackduck_user WITH PASSWORD 'my_user_password_here';

ALTER ROLE blackduck_reporter WITH PASSWORD 'my_reporter_password_here';
```

## Hub configuration

1. Ensure that the following Hub environment variables are set properly for your external PostgreSQL instance:

   - HUB_POSTGRES_ADMIN: "blackduck"

   - HUB_POSTGRES_ENABLE_SSL: "false"

     Because the connection to an external RDS is username/password rather than SSL, make sure to set the HUB_POSTGRES_ENABLE_SSL value to "false".

   - HUB_POSTGRES_HOST: "hostname"

   - HUB_POSTGRES_PORT: "5432"

   - HUB_POSTGRES_USER: "blackduck_user"

   **Note:** If using a cloud PostgreSQL with a firewall, you may need to allow firewall ingress from 'anywhere' (or at least from a range of IPs that you allocate based on your network egress IP IT information), because you may not know beforehand the IP address of the database that your containers will be connecting to.

2. Like setting proxy passwords, there are two methods to expose the database passwords to your Hub services:

   - In the clear in the config map.

   - As OpenShift secrets in the config map.

     This method is more secure, although more difficult to configure.

## Storing passwords

You can inject your PostgreSQL passwords as environment variables in the hub-config object in the openshift.yml file. The following passwords must be injected into your Web App, Job Runner, and Scan pods:

```
- name: blackduck_hub_db_password
value: blackduck
- name: blackduck_hub_report_admin_db_password
value: blackduck
- name: blackduck_hub_admin_db_password
value: blackduck
```

Alternatively, you can use password files for the three database users (**blackduck**, **blackduck_user**, and

**blackduck_reporter**), and load those as config map entities as follows:

1. Create a /tmp/password file for the **blackduck** (admin) user. The file should have the following content:

```
- apiVersion: v1
  kind: ConfigMap
  metadata:
   name: hpup-admin
  data:
   HUB_POSTGRES_ADMIN_PASSWORD_FILE: |
   <password for admin user here>
```

2. Run the following command to create the config map:

```
oc create -f /tmp/password
```

3. Repeat Steps 1 and 2 for the **blackduck_user**:

   a. Create a /tmp/password file for the **blackduck_user** user. The file should have the following content:

```
- apiVersion: v1

  kind: ConfigMap
  metadata:
   name: hpup-user
  data:
   HUB_POSTGRES_USER_PASSWORD_FILE: |
   <password for user here>
```

   b. Run the following command to create the config map:

```
oc create -f /tmp/password
```

4. Repeat Steps 1 and 2 for the **blackduck_reporter**.

## Putting DB passwords in OpenShift secrets and exposing them to Hub services

In the previous section, database passwords were configured in the clear. For added security, you can store the passwords as OpenShift secrets, and expose those secrets to the appropriate Hub services. The password secrets must be added to the pod specifications for the following Hub services:

- Jobrunner
- Webapp
- Scan

1. To store a DB password as a secret, place the password in a file, then run the oc create secret command:

```
echo "adminpw123" > admin_pwd
echo "userpw123" > user_pwd
```

```
echo "reporterpw123" > reporter_pwd

oc create secret generic hub_postgres_admin_password --from-file=./admin_
pwd
oc create secret generic hub_postgres_user_password --from-file=./user_pwd
oc create secret generic hub_postgres_reporter_password --from-
file=./reporter_pwd
```

Delete these password files after running these commands.

2. Now the that the secrets have been created in OpenShift, you must expose the secrets to the Hub services (Webapp, Jobrunner, and Scan) that require access to it. For example, the following .yml text exposes the secret to Webapp:

```
image: hub-webapp:<version number>
env:
- name: HUB_POSTGRES_ADMIN_PASSWORD
  valueFrom:
    secretKeyRef:
      name: hub_postgres_admin_password
      key: admin_pwd
- name: HUB_POSTGRES_USER_PASSWORD
  valueFrom:
    secretKeyRef:
      name: hub_postgres_user_password
      key: user_pwd
- name: HUB_POSTGRES_REPORTER_PASSWORD
  valueFrom:
    secretKeyRef:
      name: hub_postgres_reporter_password
      key: reporter_pwd
```

3. Add this section of text to the Jobrunner and Scan services by replacing `image: hub-webapp:4.4.3` with `image: jobrunner:4.4.3` and `image: scan:4.4.3` in the openshift.yml file.

# Managing certificates

By default, the Hub 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 the Hub 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 Hub server when scanning as the Hub Scanner cannot verify the certificate because it is a self-signed and is not issued by a CA. Note that the Hub Scanner 2.0 does provide an option that allows you to connect to the Hub instance with a self-signed certificate.

You can obtain a signed SSL certificate from a Certificate Authority of your choice. To obtain a signed SSL certificate, create a Certificate Signing Request (CSR), which the CA then uses to create a certificate that will identify the server running your Hub 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 the Hub instance.

## Using a custom web server certificate-key pair

You can use your own web server certificate-key pairs for establishing secure socket layer (SSL) connections to the Hub's web server.

1. To use a custom certificate, create two OpenShift secrets called `WEBSERVER_CUSTOM_CERT_FILE` and `WEBSERVER_CUSTOM_KEY_FILE` with the custom certificate and custom key, respectively, in your namespace:

   ```
   oc secret create WEBSERVER_CUSTOM_CERT_FILE --from-file=<certificate file>
   oc secret create WEBSERVER_CUSTOM_KEY_FILE --from-file=<key file>
   ```

2.  For the webserver service, add the secrets by copying their values into the env values for the pod specifications for NGiNX. The sample .yml configuration goes into the hub-nginx image stanza in the openshift.yml file :

```
env:
- name: WEBSERVER_CUSTOM_CERT_FILE
  valueFrom:
    secretKeyRef:
    name: ws-cust-cert
```

3.  Add an equivalent stanza for the custom key file:

```
env:

- name: WEBSERVER_CUSTOM_KEY_FILE
  valueFrom:
    secretKeyRef:
    name: ws-cust-key
```

# Scaling job runner and scan containers and setting memory

The job runner and scan containers can be scaled.

## Scaling job runner containers

Job runners can be scaled up or down.

This example adds a second jobrunner container:

```
oc scale rc jobrunner --replicas=2
```

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

```
oc scale rc jobrunner --replicas=1
```

## Scaling scan containers

This example adds a second scan container:

```
kubectl scale rc scan --replicas=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:

```
kubectl scale rc scan --replicas=1
```

## Setting Memory

The default memory assigned to a job runner in openshift.yml is 4 GB. Depending on the velocity of data, more memory might need to be allocated (12 GB or more, depending on circumstances).

The default memory assigned to Web App in openshift.yml is 2 GB. Depending on circumstances, more memory might need to be allocated (10GB or more).

Work with your authorized support representative to determine the right amount of memory for Job Runner and Web App for your environment.

# Configuring the report database password

A PostgreSQL report database provides access to the Hub data for reporting purposes. The database port is exposed to the OpenShift network for connections to the report user and report database.

Note the following:

- Exposed port: 55436
- Username: blackduck_reporter. This user has read-only access to the database.
- Reporting database name: `bds_hub_report`
- Reporting user password. Not initially set.

To change the blackduck_reporter password, use the following command, which in this example, sets the password to blackduck:

```
ALTER USER blackduck_reporter WITH password 'blackduck'
```

After the password is set, you can connect to the reporting database.

Note that if you create a headless service such that your external database is expressed through a PostgreSQL endpoint (this is not common), then you can use the following commands to obtain information about the internal and external IP address for your PostgreSQL service:

```
oc get service postgres -o wide
```

The command displays information such as the following:

```
NAME          CLUSTER-IP          EXTERNAL-IP        PORT(S)        AGE
postgres      1.2.3.4             <none>             5432/TCP       9d
```

Since your PostgreSQL client is outside the cluster in an OpenShift installation, take the external IP address and run the following command to open an interactive Postgres terminal to the remote database:

```
psql -U blackduck_reporter -p 55436 -h $external_ip_from_above -W bds_hub_
report
```

# Accessing the API documentation through a proxy server

If you are using a reverse proxy and that reverse proxy has Hub 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 Hub path. For example, if you have Hub being accessed under 'https://customer.companyname.com/hub' then the value of BLACKDUCK_SWAGGER_PROXY_PREFIX would be 'hub'.

As with all other environment variables, you can specify the values in the hub-nginx image stanza in the hub-config object in the openshift yml file.

# Accessing the REST APIs from a non-Hub server

You may wish to access the Hub REST APIs from a web page that was served from a non-Hub server.

To enable this feature, Cross Origin Resource Sharing (CORS) must be enabled.

The properties used to enable and configure CORS for Hub 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.<br><br>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.<br><br>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.<br><br>Note that the protocol, host, and port must match. Use a comma-separated list to specify more than one base origin URL. |

| Property | Description |
|---|---|
| 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. |

As with all other environment variables, you can specify the values in the hub-nginx image stanza in the hub-config object in the openshift yml file.

# Configuring secure LDAP

If you see certificate issues when connecting your secure LDAP server to the Hub, the most likely reason is that the Hub server has not set up a trust connection to the secure LDAP 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 Hub LDAP truststore by:

1. Obtaining your LDAP information.

2. Using the Hub UI to import the server certificate.

## Obtaining your LDAP information

Contact your LDAP administrator and gather the following information:

**LDAP Server Details**

This is the information that the Hub 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 the Hub 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 the Hub as a system administrator.

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

    The Administration page appears.

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

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

For assistance in modifying an LDAP trust store password in a OpenShift environment, contact your authorized Black Duck support representative.

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

To enable or disable SAML functionality, you must be a Sysadmin user.

For additional SAML information:

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

Note the following:

- The Hub is able to sync 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.

  The Hub is also able to sync an external user's group information if you enable group synchronization in the Hub.
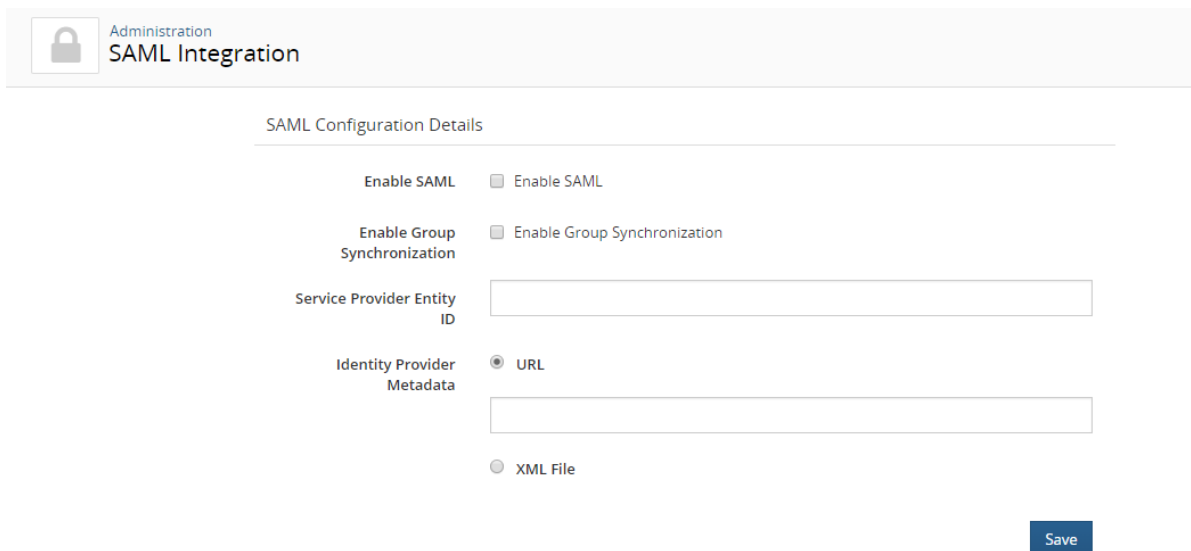
- When logging in with SAML enabled, you are re-directed to your identity provider's login page, not to The Hub's login page.

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

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

⚙ **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 IDP are created in the Hub 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. **Service Provider Entity ID** field. Enter the information for the Hub server in your environment in the format **https://host** where *host* is your Hub server.

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

4. Click **Save**.

5. Add the HUB_SAML_EXTERNAL_URL to your hub-proxy.env file (for Docker Swarm or Docker Compose) or pods.env (for Kubernetes) or as an environment variable (for OpenShift). The value is the public URL of the Hub server. For example:

   ```
   HUB_SAML_EXTERNAL_URL=https://blackduck-docker01.dc1.lan
   ```

> **Note:** You must restart the Hub for your configuration changes to take effect.

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

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

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

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

4. Click **Save**.

> **Note:** You must restart the Hub for your configuration changes to take effect.

# Backing up PostgreSQL volumes

Ensure that the volumes you use for PostgreSQL data storage are backed up on a regular basis. Consult your OpenShift/Docker/PostgreSQL system administrator for information on how to back up PostgreSQL data volumes.

This chapter describes how to upgrade an existing Hub on OpenShift to a newer version of the Hub on OpenShift.

> **Note:** Upgrading a Hub from a non-OpenShift Hub installation (for example, AppMgr Hub) to OpenShift is simply a fresh Hub install on OpenShift plus a data migration. See Chapter 3 for information on fresh Hub installs.

# Upgrading the Hub

OpenShift applications can be upgraded using native OpenShift image update commands. As such, upgrading the Hub on OpenShift is basically upgrading the Hub's deployments (pods, essentially).

## Backing up the PostgreSQL database

Black Duck recommends completing a PostgreSQL database backup prior to upgrading the Hub.

This section describes the process of backing up and restoring Hub database data. This section covers backing up AppMgr Hub data (for migration purposes).

## Backing up an PostgreSQL database from an AppMgr architecture

If you have a version of the Hub using AppMgr whose data you want to migrate to a new OpenShift PostgreSQL node, follow these steps to back up the data:

⚙ **To back up the original PostgreSQL database**

1. Log in to the Hub server as the **blckdck** user.

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

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

   ```
   export PATH=$PATH:/opt/blackduck/hub/postgresql/bin
   export PGPORT=55436
   pg_dump -Fc -f /tmp/bds_hub.dump bds_hub
   ```

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

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

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

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

"To migrate this backed-up data to your external relational database system, contact your relational database administrator.

## Upgrading the Hub

**Note:** Black Duck recommends that no scans be active/initiated and that users remained logged off the Hub web UI while the upgrade is occurring.

The command to upgrade a container in OpenShift is:

```
oc set image <image> hub-image=hub-image:version
```

The following Hub containers each needs to be individually updated:

- hub-cfssl
- hub-documentation
- hub-jobrunner
- hub-webapp
- hub-nginx
- hub-logstash
- hub-registration
- hub-solr
- hub-zookeeper
- hub-scan

For example, here are the specific commands that must be run to upgrade to the Hub 4.4.3:

```
oc set image deployment/cfssl cfssl=cfssl:4.4.3

oc set image deployment/documentation documentation=documentation:4.4.3

oc set image deployment/jobrunner jobrunner=jobrunner:4.4.3

oc set image deployment/webapp-nginx-logstash webapp=webapp:4.4.3

oc set image deployment/webapp-nginx-logstash nginx=nginx:4.4.3

oc set image deployment/webapp-nginx-logstash logstash=logstash:4.4.3

oc set image deployment/registration registration=registration:4.4.3
```

```
oc set image deployment/solr solr=solr:4.4.3

oc set image deployment/zookeeper zookeeper=zookeeper:4.4.3

oc set image deployment/scan scan=scan:4.4.3
```

This chapter provides information on debugging a Hub on OpenShift deployment. The procedures can help you determine whether your OpenShift cluster is working properly.

## Viewing running pods

Use the following command to see which pods are running:

```
oc get pods
```

You should see output similar to the following:

```
NAME                                          READY   STATUS    RESTARTS   AGE

cfssl-258485687-m3szc                         1/1     Running   0          3h
jobrunner-1397244634-xgcn2                    1/1     Running   2          26m
nginx-webapp-logstash-2564656559-6fbq8        3/3     Running   0          26m
postgres-1794201949-tt4gj                     1/1     Running   0          3h
registration-2718034894-7brjv                 1/1     Running   0          26m
solr-1180309881-sscsl                         1/1     Running   0          26m
zookeeper-3368690434-rnz3m                    1/1     Running   0          26m
```

In the above example, there are pods containing a single container each (cfssl, jobrunner, postgres, registration, solr, and zookeeper) and a pod containing three containers (nginx, webapp, logstash).

## Executing Docker commands and viewing container log files

You can use the "oc exec" command to execute a Docker command inside a Docker container inside a pod. This is especially helpful in viewing log files. The generic syntax is:

```
oc exec -t -i <pod_name> -c <container name> <Docker command>
```

For example, to view the log file of the load balancer shown in the previous example, the command is:

```
oc exec -t -i nginx-webapp-logstash-2564656559-6fbq8 -c nginx cat
/var/log/nginx/nginx-access.log
```

The command displays the following output:

```
192.168.21.128 - - [12/Jul/2017:18:13:12 +0000] "GET /api/v1/registrations?summary=true&_=1499883191824 HTTP/1.1" 200
192.168.21.128 - - [12/Jul/2017:18:13:12 +0000] "GET /api/internal/logo.png HTTP/1.1" 200 7634 "https://a0145b939671d
10.0.25.32 - - [12/Jul/2017:18:25:42 +0000] "GET / HTTP/1.1" 200 21384 "-" "curl/7.47.0" "-"
```

In another example, use the "oc logs" command to view the Docker log files (from standard out) for the Hub's Webapp container:

```
oc logs nginx-webapp-logstash-2564656559-6fbq8 -c webapp
```

Which displays the following information:

```
2017-07-12 18:13:12,064 [http-nio-8080-exec-4] INFO  com.blackducksoftware.core.regupdate.impl.RegistrationApi - Exec
2017-07-12 18:13:12,071 [http-nio-8080-exec-4] ERROR com.blackducksoftware.core.regupdate.impl.RegistrationApi - Unab
2017-07-12 18:25:42,596 [http-nio-8080-exec-1] INFO  com.blackducksoftware.usermgmt.sso.impl.BdsSAMLEntryPoint - Sing
2017-07-12 18:27:52,670 [scanProcessorTaskScheduler-1] INFO  com.blackducksoftware.scan.bom.scheduler.ScanPurgeJobMon
2017-07-12 18:30:00,059 [job.engine-0] WARN  com.blackducksoftware.job.integration.handler.KbCacheUpdater - KB projec
```

This shows all standard output from Webapp (the Hub's web server). Although a full description of the content of these log files is beyond the scope of this chapter, a large time period without log message would suggest an issue with the 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 Hub UI**

1. Log in to the Hub with the System Administrator role.

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

   The Administration page appears.

3. Select **System Settings**.

   The System Settings page appears.

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

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

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

1. Web App

2. Scan

3. Job Runner App

4. Solr

5. Registration

6. DB

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

7. WebServer

8. Zookeeper

9. LogStash

10. CA

11. Documentation

The following tables provide more information on each container.

## Web App container

| Container Name: Web App | |
|---|---|
| Image Name | blackducksoftware/hub-webapp:4.4.3 |
| Description | The Web App container is the container that all Web/UI/API requests are made against. It also processes any UI requests. In the diagram, the ports for the Web App are not exposed outside of the Docker network. There is an NGiNX reverse proxy (as described in the WebServer container) that is exposed outside of the Docker network instead. |

| Container Name: Web App | |
|---|---|
| Scalability | There should only be a single instance of this container. It should not be scaled. |
| Links/Ports | The Web App container needs to connect to these containers/services:<br><br>• postgres<br>• solr<br>• zookeeper<br>• registration<br>• logstash<br>• cfssl<br><br>The container needs to expose port 8080 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:<br><br>• postgres: $HUB_POSTGRES_HOST<br>• solr: This should be taken care of by ZooKeeper.<br>• zookeeper: $HUB_ZOOKEEPER_HOST<br>• registration: $HUB_REGISTRATION_HOST<br>• logstash: $HUB_LOGSTASH_HOST<br>• cfssl: $HUB_CFSSL_HOST |
| Constraints | • Default max Java heap size: 4GB<br>• Container memory: 4GB<br>• Container CPU: 1 CPU |
| Volumes | log-volume:/opt/blackduck/hub/logs<br><br>webapp-volume:/opt/blackduck/hub/hub-webapp/security |

## Scan container

| Container Name: Scan | |
|---|---|
| Image Name | blackducksoftware/hub-scan:4.4.3 |
| Description | The Hub scan service is the container that all scan data requests are made against. |
| Scalability | This container can be scaled. |

| Container Name: Scan | |
|---|---|
| Links/Ports | The Scan container needs to connect to these containers/services:<br><br>• postgres<br>• zookeeper<br>• registration<br>• logstash<br>• cfssl<br><br>The container needs to expose port 8080 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:<br><br>• postgres: $HUB_POSTGRES_HOST<br>• zookeeper: $HUB_ZOOKEEPER_HOST<br>• registration: $HUB_REGISTRATION_HOST<br>• logstash: $HUB_LOGSTASH_HOST<br>• cfssl: $HUB_CFSSL_HOST |
| Constraints | • Default max Java heap size: 4GB<br>• Container memory: 4GB<br>• Container CPU: 1 CPU |
| Volumes | log-volume:/opt/blackduck/hub/logs<br><br>scan-volume:/opt/blackduck/hub/hub-scan/security |

## Job runner container

| Container Name: Job Runner | |
|---|---|
| Image Name | blackducksoftware/hub-jobrunner:4.4.3 |
| Description | The Job Runner container is the container that is responsible for running all Hub jobs. This includes matching, BOM building, reports, data updates, and so on. This container does not have any exposed ports. |
| Scalability | This container can be scaled. |

| Container Name: Job Runner | |
|---|---|
| Links/Ports | The Job Runner container needs to connect to these containers/services:<br><br>• postgres<br>• solr<br>• zookeeper<br>• registration<br>• logstash<br>• cfssl |
| Alternate Host Name Environment Variables | There are times when running in other types of orchestrations that it is useful to have host names set for these containers that are not the default. 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:<br><br>• postgres: $HUB_POSTGRES_HOST<br>• solr: This should be taken care of by ZooKeeper.<br>• zookeeper: $HUB_ZOOKEEPER_HOST<br>• registration: $HUB_REGISTRATION_HOST<br>• logstash: $HUB_LOGSTASH_HOST<br>• cfssl: $HUB_CFSSL_HOST |
| Constraints | • Default max Java heap size: 4GB<br>• Container memory: 4GB<br>• Container CPU: 1 CPU |
| Volumes | N/A |

## Solr container

| Container Name: Solr | |
|---|---|
| Image Name | blackducksoftware/hub-solr:4.4.3 |
| Description | Solr is an open source enterprise search platform. The Hub uses Solr as its search server for project data.<br><br>This container has Apache Solr running within it. There is only a single instance of this container. The Solr container exposes ports internally to the Docker network, but not outside of the Docker network. |
| Scalability | This container should not be scaled. |

| Container Name: Solr | |
|---|---|
| Links/Ports | The Solr container needs to connect to these containers/services:<br><br>• zookeeper<br>• logstash<br><br>The container needs to expose port 8080 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:<br><br>• zookeeper: $HUB_ZOOKEEPER_HOST<br>• logstash: $HUB_LOGSTASH_HOST |
| Constraints | • Default max Java heap size: 512MB<br>• Container memory: 512MB<br>• Container CPU: Unspecified |
| Volumes | solr6-volume:/opt/blackduck/hub/solr/cores.data |

## Registration container

| Container Name: Registration | |
|---|---|
| Image Name | blackducksoftware/hub-registration:4.4.3 |
| 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:<br><br>• logstash<br><br>The container needs to expose port 8080 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. These environment variables can be set to override the default host names:<br><br>• logstash: $HUB_LOGSTASH_HOST |

| Container Name: Registration | |
|---|---|
| Constraints | • Default max Java heap size: 256MB<br><br>• Container memory: 256MB<br><br>• Container CPU: Unspecified |
| Volumes | config-volume:/opt/blackduck/hub/registration/config |

# DB container

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

| Container Name: DB | |
|---|---|
| Image Name | blackducksoftware/hub-postgres:4.4.3 |
| Description | The DB container holds the PostgreSQL database which is an open source object-relational database system. The Hub uses the PostgreSQL database to store data.<br><br>There is a single instance of this container. This is where all Hub data is stored. This is the connection that the Hub App, Job Runner, and potentially other containers use. This port is secured via certificate authentication. A second port is exposed outside of the Docker network. This allows a read-only user to connect via a password set using the `hub_reportdb_changepassword.sh` script. This port and user can be used for reporting and data extraction.<br><br>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:<br><br>• logstash<br><br>• cfssl<br><br>The container needs to expose port 5432 to other containers that will link to it within the Docker network.<br><br>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 defaul. These environment variables can be set to override the default host names: |

| Container Name: DB | |
|---|---|
| | • logstash: $HUB_LOGSTASH_HOST<br>• cfssl: $HUB_CFSSL_HOST |
| Constraints | • Default max Java heap size: N/A<br>• Container memory: 3GB<br>• Container CPU: 1 CPU |
| Volumes | data-volume:/var/lib/postgresql/data |

## WebServer container

| Container Name: WebServer | |
|---|---|
| Image Name | blackducksoftware/hub-nginx:4.4.3 |
| Description | The WebServer container is a reverse proxy for the Hub Web App. It has a port exposed outside of the Docker network. This is the container configured for HTTPS. There are config volumes here for configuration of HTTPS. |
| Scalability | The container should not be scaled. |
| Links/Ports | The Web App container needs to connect to these containers/services:<br><br>• webapp<br>• cfssl<br>• documentation<br>• scan<br><br>This container exposes port 443 outside of the Docker network. |
| Alternate Host Name Environment Variables | There are times when running in other types of orchestrations that it is useful to have host names set for these containers that are not the default. These environment variables can be set to override the default host names:<br><br>• webapp: $HUB_WEBAPP_HOST<br>• cfssl: $HUB_CFSSL_HOST<br>• documentation: $HUB_DOC_HOST<br>• scan: $HUB_SCAN_HOST |
| Constraints | • Default max Java heap size: N/A<br>• Container memory: 512MB<br>• Container CPU: Unspecified |
| Volumes | webserver-volume:/opt/blackduck/hub/webserver/security |

## ZooKeeper container

| Container Name: Zookeeper | |
|---|---|
| Image Name | blackducksoftware/hub-zookeeper:4.4.3 |
| Description | This container stores data for the Hub App, Job Runners, Solr, and potentially other containers. It exposes ports within the Docker network, but not outside the Docker network. |
| Scalability | This container should not be scaled. |
| Links/Ports | The Zookeeper container needs to connect to this container/service:<br><br>• logstash<br><br>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. These environment variables can be set to override the default host names:<br><br>• logstash: $HUB_LOGSTASH_HOST |
| Constraints | • Default max Java heap size: 256MB<br>• Container memory: 256MB<br>• Container CPU: Unspecified |
| Volumes | N/A |

## LogStash container

| Container Name: LogStash | |
|---|---|
| Image Name | blackducksoftware/hub-logstash:4.4.3 |
| 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. |

| Container Name: LogStash | |
|---|---|
| Constraints | • Default max Java heap size: 1GB<br>• Container memory: 1GB<br>• Container CPU: Unspecified |
| Volumes | log-volume:/var/lib/logstash/data |

## CA container

| Container Name: CA | |
|---|---|
| Image Name | blackducksoftware/hub-cfssl:4.4.3 |
| Description | The CA container uses CFSSL which is used for certificate generation for PostgreSQL, NGiNX, and clients that need to authenticate to Postgres. |
| Scalability | There should only be a single instance of this container. It should not be scaled. |
| Links/Ports | The container needs to expose port 8888 within the Docker network to other containers/services that link to it. |
| Constraints | • Default max Java heap size: N/A<br>• Container memory: 512MB<br>• Container CPU: Unspecified |
| Volumes | cert-volume:/etc/cfssl |

## Documentation container

| Container Name: Documentation | |
|---|---|
| Image Name | blackducksoftware/hub-documentation:4.4.3 |
| Description | The Documentation container supplies documentation for the Hub. |
| 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:<br><br>• logstash<br><br>The documentation container must expose port 8080 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. These environment variables can be set to override the default |

| Container Name: Documentation | |
|---|---|
| | host names:<br><br>• logstash: $HUB_LOGSTASH_HOST |
| Constraints | • Default Max Java Heap Size: 512MB<br><br>• Container Memory: 512MB<br><br>• Container CPU: unspecified |
| Volumes | N/A |