



Docker Swarmを使用したBlack Duckのインストール

Black Duck 2022.10.0

目次

まえがき.....	6
Black Duck ドキュメント.....	6
カスタマサポート.....	6
Synopsys Software Integrityコミュニティ.....	7
トレーニング.....	7
包括性と多様性に関するSynopsysの声明.....	7
 1. 概要.....	9
Black Duck アーキテクチャ.....	9
Black Duckサーバーでホストされているコンポーネント.....	9
 2. インストールの計画.....	10
使用する前に.....	10
新規インストール.....	10
以前のバージョンのからのアップグレード: Black Duck.....	10
ハードウェア要件.....	10
Dockerの要件.....	12
Dockerバージョン.....	12
RHELへのDockerエンジンのインストール.....	13
オペレーティングシステム.....	13
ソフトウェア要件.....	13
ネットワーク要件.....	13
データベース要件.....	14
PostgreSQLのバージョン.....	15
一般的な移行プロセス.....	15
Swarmでの移行の概要.....	16
プロキシサーバーの要件.....	16
連携するNGiNXサーバーの設定: Black Duck.....	16
Amazonサービス.....	17
ポートに関するその他の情報.....	17
キープアライブ設定の構成.....	18
ナレッジベースフィードバックサービス.....	18
ユーザーエージェント解析.....	19
フィードバックサービスの無効化.....	19
 3. インストール: Black Duck.....	20
インストールファイル.....	20
GitHubページからダウンロードする場合.....	20
wgetコマンドを使用してダウンロードする場合.....	21
配布.....	21
インストール: Black Duck.....	22
高速スキャン: Black Duck.....	23
 4. 管理タスク.....	25

A.....	25
B.....	25
C.....	25
D.....	25
H.....	25
I.....	25
L.....	25
M.....	26
O.....	26
P.....	26
R.....	26
S.....	26
T.....	26
U.....	26
W.....	26
環境ファイルの使用.....	27
環境変数とバイナリのスキャン.....	27
重複している構成表の検出.....	27
Bearerトークンの有効期限の変更.....	27
KBMATCH_SENDFPATHパラメータについて.....	28
プロキシサーバー経由でのAPIドキュメントへのアクセス.....	28
Black Duckサーバー以外からのREST APIへのアクセス.....	28
バイナリスキャンファイルのサイズ値の増大.....	30
ダッシュボードの更新間隔の構成.....	30
証明書の管理.....	30
カスタム証明書の使用.....	31
ナレッジベースからのコンポーネント移行データの取得.....	32
移行の記録の有効化.....	33
移行データの保持.....	33
APIエンドポイント.....	33
階層構成表の有効化.....	33
無視されたコンポーネントをレポートに含める.....	33
セキュリティ保護されたLDAPの設定.....	34
LDAP情報の取得.....	34
サーバー証明書のインポート.....	34
LDAP信頼ストアのパスワード.....	36
ログファイルとヒートマップデータのダウンロード.....	36
ログファイルの表示.....	37
デフォルトのメモリ制限の変更.....	37
webappコンテナのデフォルトのメモリ制限の変更.....	38
jobrunnerコンテナのデフォルトのメモリ制限の変更.....	38
scanコンテナのデフォルトのメモリ制限の変更.....	40
binaryscannerコンテナのデフォルトのメモリ制限の変更.....	40
bomengineコンテナのデフォルトのメモリ制限の変更.....	41
logstashのホスト名の変更.....	41
マップされていないコードの場所のクリーンアップ.....	42
cron文字列を使用したスキャンパージジョブのスケジュール.....	42
スタックしている構成表イベントのクリア.....	42
上書きファイルの使用.....	43
分析の構成: Black Duck.....	43
外部PostgreSQLインスタンスの構成.....	43
既存の外部データベースのPostgreSQLユーザー名の変更.....	47
プロキシ設定の構成.....	48
プロキシパスワード.....	48

プロキシ証明書のインポート.....	49
レポートデータベースのパスワードの設定.....	49
job runner、scan、bomengine、およびbinaryscannerコンテナのスケールリング.....	50
bomengineコンテナのスケールリング.....	50
job runnerコンテナのスケールリング.....	50
scanコンテナのスケールリング.....	50
binaryscannerコンテナのスケールリング.....	50
シングルサインオンのSAMLの設定.....	51
ソースファイルのアップロード.....	53
シールキーと未処理のマスターキーのバックアップ.....	54
シールキーの置き換え.....	54
追加の構成オプション.....	54
起動または停止 Black Duck.....	55
起動 Black Duck.....	55
上書きファイル使用時のBlack Duckの起動.....	55
シャットダウン Black Duck.....	56
ユーザーセッションタイムアウトの構成.....	56
カスタマサポートへのお客様のBlack Duckシステム情報の提供.....	58
デフォルトのsysadminユーザーについて.....	58
Black Duckレポート遅延の構成.....	58
コンテナのタイムゾーンの構成.....	58
デフォルトの使用法の変更.....	59
アップロードされたjsonld/bdioファイルbdio2のマッチタイプ.....	60
Black DuckコンテナのユーザーIDのカスタマイズ.....	61
Webサーバー設定の構成.....	62
ホスト名の構成.....	62
ホストポートの構成.....	62
IPv6の無効化.....	62
UTF8文字エンコードを使用した構成表レポートの作成.....	63
スキャン監視.....	63
HTMLレポートのダウンロードサイズの設定.....	63
 5. アンインストール: Black Duck.....	64
 6. のアップグレード Black Duck.....	65
インストールファイル.....	65
GitHubページからダウンロードする場合.....	65
wgetコマンドを使用してダウンロードする場合.....	65
監査イベントテーブルの未使用の行をパージする移行スクリプト.....	66
AppMgrアーキテクチャからのアップグレード.....	67
PostgreSQLデータベースの移行.....	67
のアップグレード Black Duck.....	68
シングルコンテナAppMgrアーキテクチャからのアップグレード.....	69
PostgreSQLデータベースの移行.....	69
Black Duckのアップグレード Black Duck.....	70
既存のDockerアーキテクチャからのアップグレード.....	71
PostgreSQLデータベースの移行.....	71
のアップグレード Black Duck.....	72
 7. Dockerコンテナ.....	74

Authenticationコンテナ	75
CAコンテナ	76
DBコンテナ	77
Documentationコンテナ	78
Jobrunnerコンテナ	78
Logstashコンテナ	79
Registrationコンテナ	80
Scanコンテナ	80
Uploadcacheコンテナ	81
Webappコンテナ	82
Webserverコンテナ	83
Redisコンテナ	84
Rabbitmqコンテナ	85
bomengineコンテナ	85
blackduck-matchengineコンテナ	86
webuiコンテナ	87
Black Duck - Binary Analysis コンテナ	87
Binaryscannerコンテナ	87

まえがき

Black Duck ドキュメント

Black Duckのドキュメントは、オンラインヘルプと次のドキュメントで構成されています。

タイトル	ファイル	説明
リリースノート	release_notes.pdf	新機能と改善された機能、解決された問題、現在のリリースおよび以前のリリースの既知の問題に関する情報が記載されています。
Docker Swarmを使用したBlack Duckのインストール	install_swarm.pdf	Docker Swarmを使用したBlack Duckのインストールとアップグレードに関する情報が記載されています。
使用する前に	getting_started.pdf	初めて使用するユーザーにBlack Duckの使用法に関する情報を提供します。
スキャンベストプラクティス	scanning_best_practices.pdf	スキャンのベストプラクティスについて説明します。
SDKを使用する前に	getting_started_sdk.pdf	概要およびサンプルのユースケースが記載されています。
レポートデータベース	report_db.pdf	レポートデータベースの使用に関する情報が含まれています。
ユーザーガイド	user_guide.pdf	Black DuckのUI使用に関する情報が含まれています。

KubernetesまたはOpenShift環境にBlack Duckソフトウェアをインストールするためのインストール方法は、SynopsysctlとHelmです。次のリンクをクリックすると、マニュアルが表示されます。

- ・ [Helm](#)は、Black Duckのインストールに使用できるKubernetesのパッケージマネージャです。
- ・ [Synopsysctl](#)は、KubernetesおよびRed Hat [OpenShift](#)にBlack Duckソフトウェアを展開するためのクラウドネイティブの管理コマンドラインツールです。

Black Duck 統合に関するドキュメントは[Confluence](#)で入手できます。

カスタマサポート

ソフトウェアまたはドキュメントについて問題がある場合は、Synopsysカスタマサポートに問い合わせてください。

Synopsysサポートには、複数の方法で問い合わせできます。

- ・ オンライン: <https://www.synopsys.com/software-integrity/support.html>
- ・ 電話: お住まいの地域の電話番号については、[サポートページ](#)の下段にあるお問い合わせのセクションを参照してください。

サポートケースを開くには、Synopsys Software Integrityコミュニティサイト(<https://community.synopsys.com/s/contactsupport>)にログインしてください。

常時対応している便利なリソースとして、[オンラインカスタマポータル](#)を利用できます。

Synopsys Software Integrityコミュニティ

Synopsys Software Integrityコミュニティは、カスタマサポート、ソリューション、および情報を提供する主要なオンラインリソースです。コミュニティでは、サポートケースをすばやく簡単に開いて進捗状況を監視したり、重要な製品情報を確認したり、ナレッジベースを検索したり、他のSoftware Integrityグループ(SIG)のお客様から情報を得ることができます。コミュニティセンターには、共同作業に関する次の機能があります。

- ・ つながる – サポートケースを開いて進行状況を監視するとともに、エンジニアリング担当や製品管理担当の支援が必要になる問題を監視します。
- ・ 学ぶ – 他のSIG製品ユーザーの知見とベストプラクティスを通じて、業界をリードするさまざまな企業から貴重な教訓を学ぶことができます。さらにCustomer Hubでは、最新の製品ニュースやSynopsysの最新情報をすべて指先の操作で確認できます。これは、オープンソースの価値を組織内で最大限に高めるように当社の製品やサービスをより上手に活用するのに役立ちます。
- ・ 解決する – SIGの専門家やナレッジベースが提供する豊富なコンテンツや製品知識にアクセスして、探している回答をすばやく簡単に得ることができます。
- ・ 共有する – Software Integrityグループのスタッフや他のお客様とのコラボレーションを通じて、クラウドソースソリューションに接続し、製品の方向性について考えを共有できます。

[Customer Successコミュニティにアクセスしましょう](#)。アカウントをお持ちでない場合や、システムへのアクセスに問題がある場合は、[こちら](#)をクリックして開始するか、community.manager@synopsys.comにメールを送信してください。

トレーニング

Synopsys Software Integrity, Customer Education (SIG Edu) は、すべてのBlack Duck教育ニーズに対応するワンストップリソースです。ここでは、オンライントレーニングコースやハウツービデオへの24時間365日のアクセスを利用できます。

新しいビデオやコースが毎月追加されます。

Synopsys Software Integrity, Customer Education (SIG Edu) では、次のことができます。

- ・ 自分のペースで学習する。
- ・ 希望する頻度でコースを復習する。
- ・ 試験を受けて自分のスキルをテストする。
- ・ 終了証明書を印刷して、成績を示す。

詳細については、<https://community.synopsys.com/s/education>を参照してください。また、Black Duckのヘルプに

ついては、Black Duck UIの[ヘルプ]メニュー()から、[Black Duckチュートリアル]を選択します。

包括性と多様性に関するSynopsysの声明

Synopsysは、すべての従業員、お客様、パートナーが歓迎されていると感じられる包括的な環境の構築に取り組んでいます。当社では、製品およびお客様向けのサポート資料から排他的な言葉を確認して削除しています。また、当社の取り組みには、設計および作業環境から偏見のある言葉を取り除く社内イニシアチブも含まれ、これはソフトウェアやIPに組み込まれている言葉も対象になっています。同時に、当社は、能力の異なるさまざまな人々が当社のWebコンテンツおよびソフトウェアアプリケーションを利用できるように取り組んでいます。なお、当社のIPは、排他

的な言葉を削除するための現在検討中である業界標準仕様を実装しているため、当社のソフトウェアまたはドキュメントには、非包括的な言葉の例がまだ見つかる場合があります。

1. 概要

このドキュメントでは、Docker環境にBlack Duckをインストールする手順について説明します。

Black Duck アーキテクチャ

Black Duck は、Dockerコンテナのセットとして導入されます。さまざまなコンポーネントがコンテナ化されるようにBlack DuckをDocker化することで、Swarmなどのサードパーティのオーケストレーションツールで個々のコンテナをすべて管理できるようになります。

Dockerアーキテクチャは、次のような大幅な改善をBlack Duckにもたらしめます。

- ・ パフォーマンスの向上
- ・ より簡単なインストールと更新
- ・ スケーラビリティ
- ・ 製品コンポーネントのオーケストレーションと安定性

Black Duckアプリケーションを構成するDockerコンテナの詳細については、「[Dockerコンテナ](#)」を参照してください。

Dockerの詳細については、Docker Webサイト(<https://www.docker.com/>)を参照してください。

Dockerのインストール情報を取得するには、<https://docs.docker.com/engine/installation/>にアクセスしてください。

Black Duckサーバーでホストされているコンポーネント

次のリモートBlack Duckサービスは、Black Duckによって使用されます。

- ・ 登録サーバー: Black Duckのライセンスを検証するのに使用されます。
- ・ Black Duckナレッジベースサーバー: Black Duckナレッジベース(KB)は、業界で最も包括的な、オープンソースプロジェクト、ライセンス、およびセキュリティ情報のデータベースです。クラウドのBlack Duck KBを使用することで、Black Duckは、Black Duckインストールを定期的に更新することなく、オープンソースソフトウェア(OSS)に関する最新情報を表示できます。

2. インストールの計画

この章では、Black Duckをインストールする前に実行する必要があるインストール前の計画と構成について説明します。

使用する前に

Black Duckのインストールのプロセスは、Black Duckを初めてインストールするか、以前のバージョンのBlack Duckからアップグレードするか（AppMgrアーキテクチャに基づくか、Dockerアーキテクチャに基づくか）によって異なります。

新規インストール

Black Duckの新規インストールの場合は、次の手順を実行します。

1. この計画に関する章を読んで、すべての要件を確認します。
2. すべての要件を満たしていることを確認したら、インストール手順について[インストール: Black Duck](#)を確認します。
3. [管理タスク](#)を確認します。

以前のバージョンのからのアップグレード: Black Duck

1. この計画に関する章を読んで、すべての要件を確認します。
2. すべての要件を満たしていることを確認したら、アップグレード手順についての[アップグレード Black Duck](#)を確認します。
3. 管理タスクについて[管理タスク](#)を確認します。


ハードウェア要件

バージョン2022.4.0の時点で、ハードウェアの推奨最小要件は、7 CPUコアおよび28.5 GBのRAMから14 CPUコアおよび38 GBのRAMに増加しました。提供されているHelmチャートまたはswarm構成ファイルを使用して、ハードウェア要件とリソース割り当てを手動で変更することもできます。ただし、現在のBlack Duckのデフォルトテンプレートは、新しい推奨事項と一致します。短期から中期的には、Black Duckはより低い要件で正常に実行できると思われますが、これは今後変わる可能性があります。将来的には、Black Duckは、古いハードウェア仕様の公式サポートまたは古いハードウェア仕様でのテストを終了します。新しい要件を満たすためのハードウェアがないお客様は、2022.4.0に更新する前に、ハードウェアを追加するか、オーケストレーションファイルを手動で変更する必要があります。

名前	スキャン/時間	Black Duckサービス	PostgreSQL	合計
10sph	10	CPU: 12コア メモリ: 30 GB	CPU: 2コア メモリ: 8 GB	CPU: 14コア メモリ: 38 GB
120sph	120	CPU: 13コア メモリ: 46 GB	CPU: 4コア メモリ: 16 GB	CPU: 17コア メモリ: 62 GB
250sph	250	CPU: 17コア メモリ: 118 GB	CPU: 6コア メモリ: 24 GB	CPU: 23コア メモリ: 142 GB
500sph	500	CPU: 28コア	CPU: 10コア	CPU: 38コア

		メモリ: 210 GB	メモリ: 40 GB	メモリ: 250 GB
1000sph	1000	CPU: 47コア メモリ: 411 GB	CPU: 18コア メモリ: 72 GB	CPU: 65コア メモリ: 483 GB
1500sph	1500	CPU: 66コア メモリ: 597 GB	CPU: 26コア メモリ: 104 GB	CPU: 92コア メモリ: 701 GB
2000sph	2000	CPU: 66コア メモリ: 597 GB	CPU: 34コア メモリ: 136 GB	CPU: 100コア メモリ: 733 GB

この新しいガイダンスは、現在のBlack Duck 2022.2.0アーキテクチャに基づいています。このガイダンスは、以降のリリースでさらに改良される可能性があります。ご質問やご不明な点がある場合は、製品管理までお問い合わせください。

 注：必要なディスク容量は、管理するプロジェクトの数によって異なります。したがって、個々の要件が異なる場合があります。各プロジェクトには約200 MBが必要であることを考慮してください。

Black Duck Softwareでは、Black Duckサーバーのディスク使用率を監視して、ディスクが最大容量に達しないようにすることを推奨しています。最大容量に達すると、Black Duckで問題が発生する可能性があります。

BDBAのスケーリングは、1時間あたりに実行される予想バイナリスキャン数に基づいて、binaryscannerレプリカ数を調整し、PostgreSQLリソースを追加することによって行われます。1時間あたり15回のバイナリスキャンごとに、次を追加します。

- ・ 1つのbinaryscannerレプリカ
- ・ PostgreSQL用の1つのCPU
- ・ PostgreSQL用の4 GBのメモリ

予想されるスキャンレートが15の倍数でない場合は、切り上げます。たとえば、1時間あたり24回のバイナリスキャンでは、次のものが必要です。

- ・ 2つのbinaryscannerレプリカ
- ・ PostgreSQL用の2つの追加CPU、および
- ・ PostgreSQL用の8 GBの追加メモリ。

このガイダンスは、バイナリスキャンが合計スキャンボリューム（スキャン数）の20%以下である場合に有効です。

バイナリスキャン

バイナリスキャンのライセンスがある場合、uploadcacheコンテナ/ポッドのメモリを増やす必要がある場合があります。これは、バイナリスキャナがバイナリを抽出して処理する場所であるためです。デフォルトでは、メモリは512MBに設定されていますが、これは大規模なスキャンには適切ではありません。大規模なバイナリをスキャンする場合は、uploadcacheコンテナ/ポッドのメモリを4 GB以上に増やすことをお勧めします。これを実行するには、YAMLを上書きして、メモリ制限を4096MBに更新します。


Swarmインストールの場合：

```
uploadcache:
  deploy:
    resources:
      limits:
        cpus: ".200"
        memory: "4096M"
      reservations:
        cpus: ".100"
        memory: "4096M"
    replicas: 1
```

Kubernetesインストールの場合：


2. インストールの計画・Dockerの要件

```
uploadcache:
  replicas: 1
  resources:
    limits:
      cpu: "200m"
      memory: "4096Mi"
    requests:
      cpu: "100m"
      memory: "4096Mi"
```


 注：Black Duck Alertをインストールするには、1 GBの追加メモリが必要です。

Dockerの要件

Black Duckをインストールするための推奨される方法であるDocker Swarmは、Dockerコンテナのクラスタリングおよびスケジューリングツールです。Docker Swarmを使用すると、Dockerノードのクラスタを1つの仮想システムとして管理できます。

 注：スケーラビリティの観点から、Black Duck Softwareでは、単一ノードのSwarm導入でBlack Duckを実行することを推奨しています。スケーラビリティのサイジングの詳細なガイドラインについては、『Black Duckリリースノート』の「コンテナのスケーラビリティ」セクションを参照してください。

Docker SwarmでBlack Duckを使用する場合、次の制限があります。

-  注意：PostgreSQLデータディレクトリでウイルス対策スキャンを実行しないでください。ウイルス対策ソフトウェアは、大量のファイルを開いたり、ファイルをロックしたりします。これらはPostgreSQLの操作を妨げます。特定のエラーは製品によって異なりますが、通常、PostgreSQLがデータファイルにアクセスできなくなります。たとえば、PostgreSQLが「システムで開かれているファイルが多すぎます」というエラーを伴って失敗することがあります。
- ・ データが失われないように、PostgreSQLデータベースは常にクラスタ内の同じノードで実行する必要があります (blackduck-databaseサービス)。
これは、外部PostgreSQLインスタンスを使用したインストールには該当しません。
- ・ blackhack-webappサービスとblackhack-logstashサービスは、同じホスト上で実行する必要があります。
これは、ダウンロードする必要があるログにblackduck-webappサービスがアクセスできるようにするのに必要です。
- ・ 登録データが失われないように、blackduck-registrationサービスは常にクラスタ内の同じノードで実行するか、NFSボリュームまたは同様のシステムによってバックアップされる必要があります。
blackduck-databaseサービスまたはblackduck-webappサービスに使用されるノードと同じノードである必要はありません。
- ・ データが失われないように、blackduck-upload-cacheサービスは常にクラスタ内の同じノードで実行するか、NFSボリュームまたは同様のシステムによってバックアップされる必要があります。
他のサービスに使用されるノードと同じノードである必要はありません。

Dockerバージョン

Black Duck のインストールでは、Dockerバージョン18.09.x、19.03.x、および20.10.x (CEまたはEE) がサポートされています。

RHELへのDockerエンジンのインストール


現在Dockerは、s390x (IBM Z) 上のRHEL用のパッケージのみを提供しています。他のアーキテクチャはRHELではまだサポートされていませんが、CentOSパッケージをRHELにインストールできる場合があります。詳細については、「[CentOSへのDockerエンジンのインストール](#)」ページを参照してください。

オペレーティングシステム

Docker環境にBlack Duckをインストールするのに推奨されるオペレーティングシステムは次のとおりです。

- ・ CentOS 7.3
- ・ Red Hat Enterprise Linux Server 7.3
- ・ Ubuntu 18.04.x
- ・ SUSE Linux Enterprise Serverバージョン12.x (64ビット)
- ・ Oracle Enterprise Linux 7.3

さらに、Black Duckは、サポートされているDockerバージョンに対応している他のLinuxオペレーティングシステムをサポートしています。

 注： Docker CEは、Red Hat Enterprise Linux、Oracle Linux、またはSUSE Linux Enterprise Server (SLES) をサポートしていません。詳細については、[ここ](#)をクリックしてください。


Windowsオペレーティングシステムは現在サポートされていません。

ソフトウェア要件

Black Duck は、HTMLインターフェイスを備えたWebアプリケーションです。アプリケーションにはWebブラウザを介してアクセスします。Black Duckでは、次のWebブラウザバージョンがテストされています。

- ・ Safariバージョン15.4 (16613.1.17.1.13、16613)
 - ・ Safariバージョン13.0以前はサポートされなくなりました
- ・ Chromeバージョン100.0.4896.75 (公式ビルド) (x86_64)
 - ・ Chromeバージョン71以前はサポートされなくなりました
- ・ Firefoxバージョン99.0 (64ビット)
 - ・ Firefoxバージョン71以前はサポートされなくなりました
- ・ Microsoft Edgeバージョン100.0.1185.36 (公式ビルド) (64ビット)
 - ・ Microsoft Edgeバージョン78以前はサポートされなくなりました

Black Duckは互換モードをサポートしていません。

 注： これらのブラウザバージョンは、Black Duck SoftwareがBlack Duckをテストした現在リリースされているバージョンです。Black Duckのリリース後に新しいブラウザバージョンが利用可能になる場合があります、期待どおりに機能する場合と機能しない場合があります。古いバージョンのブラウザも期待どおりに機能する可能性があります。ただし、テストされておらず、サポートされていない可能性があります。

ネットワーク要件


Black Duck では、次のポートが外部からアクセスできる必要があります。

2. インストールの計画・データベース要件

- ・ ポート443 - NGiNXを介したBlack DuckのWebサーバーHTTPSポート
- ・ ポート55436 - PostgreSQLからの読み取り専用データベースポート(レポート用)

企業のセキュリティポリシーで特定のURLの登録が必要な場合、Black DuckインストールからBlack Duck Softwareホストサーバーへの接続は、ポート443でのHTTPS/TCPを介した次のサーバーとの通信に制限されます。

- ・ updates.suite.blackducksoftware.com (ソフトウェア登録用)
- ・ kb.blackducksoftware.com (Black Duck KBデータへのアクセス)
- ・ https://auth.docker.io/token?scope=repository/blackducksoftware/blackduckregistration/pull&service=registry.docker.io (Dockerレジストリへのアクセス)


 注：ネットワークプロキシを使用している場合は、これらのURLをプロキシ構成で宛先として設定する必要があります。

許可リストに次のアドレスがあることを確認します。

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


接続を確認するには、次の例に示すようにcURLコマンドを使用します。

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


 ヒント：Dockerホストで接続を確認することもできますが、Dockerネットワーク内から接続を確認することをお勧めします。

データベース要件

Black Duck では、PostgreSQLオブジェクトリレーショナルデータベースを使用してデータを格納します。

 注意：Synopsysのテクニカルサポート担当者から特に要請がない限り、Black Duckデータベース(bds_hub)からデータを削除しないでください。必ず適切なバックアップ手順に従ってください。データを削除すると、UIの問題からBlack Duckが完全に起動しなくなるという障害に至る、いくつかのエラーが発生する可能性があります。Synopsysのテクニカルサポートは、削除されたデータを再作成することはできません。利用可能なバックアップがない場合、Synopsysは可能な範囲で最善のサポートを提供します。


Black Duckをインストールする前に、自動的にインストールされるデータベースコンテナを使用するか、外部のPostgreSQLインスタンスを使用するかを判断します。

 重要：Black Duck 2022.7.0の時点で、Synopsysでは、外部PostgreSQLを使用する新規インストールにはPostgreSQL 14を使用することを推奨しています。PostgreSQL 11.xは、外部PostgreSQLインスタンスではサポートされなくなりました。内部PostgreSQLコンテナを使用している場合、PostgreSQL 13は引き続きサポート対象バージョンです。


外部PostgreSQLインスタンスの場合、Black Duckは以下をサポートしています。

- ・ Amazon Relational Database Service (RDS)を介したPostgreSQL 13.x、14.x
- ・ Google Cloud SQLを介したPostgreSQL 13.x、14.x
- ・ PostgreSQL 13.x、14.x (Community Edition)

- ・ Microsoft Azureを介したPostgreSQL 13.x、14.x

 注： PostgreSQL 10.xまたはPostgreSQL 12.xはサポートされていません。


詳細については、「[外部PostgreSQLインスタンスの構成](#)」を参照してください。

 注： PostgreSQLのサイジングのガイドラインについては、『[Black Duck Hardware Scaling Guidelines](#)』を参照してください。


PostgreSQLのバージョン

Black Duck 2022.10.0は、新しいPostgreSQLの機能をサポートし、Black Duckサービスのパフォーマンスと信頼性を向上させます。Black Duck 2022.10.0の時点で、内部PostgreSQLコンテナの現在サポートされているPostgreSQLのバージョンは、PostgreSQLコンテナ13です。

以前のバージョンのBlack Duck(2022.10.0より前)からアップグレードする場合は、PostgreSQL 13に移行する必要があります。Black Duck 2022.10.0アップデートは、内部Black Duck PostgreSQLデータベースコンテナをPostgreSQLのバージョン13に移行します。データベースコンテナを使用してOpenShiftに導入する場合は、Black Duckのリリースノートとインストールガイドに記載されているように、1回限りの移行ジョブを実行する必要があります。

 注： PostgreSQLのサイジングのガイドラインについては、『[Black Duck Hardware Scaling Guidelines](#)』を参照してください。

独自の外部PostgreSQLインスタンスを実行する場合は、新規インストールにPostgreSQL 14を使用することをお勧めします。PostgreSQL 14.0から14.3には、インデックスが破損するというバグがあるため、サポートされているPostgreSQL 14の最小バージョンは14.4です。

 注意： PostgreSQLデータディレクトリでウイルス対策スキャンを実行しないでください。ウイルス対策ソフトウェアは、大量のファイルを開いたり、ファイルをロックしたりします。これらはPostgreSQLの操作を妨げます。特定のエラーは製品によって異なりますが、通常、PostgreSQLがデータファイルにアクセスできなくなります。たとえば、PostgreSQLが「システムで開かれているファイルが多すぎます」というエラーを伴って失敗することがあります。

一般的な移行プロセス

このガイドは、任意のPG 9.6ベースのHub(2022.2.0より前のリリース)から2022.10.0以降にアップグレードする場合に該当します。

1. 移行は、blackadue-postgres-upgraderコンテナによって実行されます。
2. PostgreSQL 9.6ベースのBlack Duckバージョンからアップグレードする場合：
 - ・ 将来のPostgreSQLバージョンのアップグレードがより簡単になるように、PostgreSQLデータボリュームのフォルダレイアウトが再構成されます。
 - ・ データボリュームの所有者のUIDが変更されます。新しいデフォルトUIDは1001です。ただし、導入固有の説明を参照してください。
3. pg_upgradeスクリプトを実行して、データベースをPostgreSQL 13に移行します。
4. クエリプランナ統計情報を初期化するために、PostgreSQL 13データベース上でプレーンなANALYZEが実行されます。
5. blackduck-postgres-upgraderが終了します。

Swarmでの移行の概要

- ・ 移行は完全に自動化されているため、Black Duckの標準アップグレードの操作以外に追加の操作は必要ありません。
- ・ 上記のレイアウトとUIDの変更を行うには、blackduck-postgres-upgraderコンテナをルートとして実行する必要があります。
- ・ その後のBlack Duckの再起動時に、blackadu-postgres-upgraderは移行が不要であると判断し、すぐに終了します。
- ・ オプション: 移行が成功した後は、blackduck-postgres-upgraderコンテナをルートとして実行する必要はありません。

4.2.0より前のバージョンのBlack Duckからアップグレードする場合は、データベースの移行手順について、第6章「Black Duckのアップグレード」を参照してください。

プロキシサーバーの要件

Black Duck は、次のものをサポートしています。


- ・ 認証なし
- ・ Digest
- ・ Basic
- ・ NTLM

Black Duckにプロキシリクエストを行う場合は、プロキシサーバー管理者と協力して、次の必要な情報を取得します。

- ・ プロキシサーバーホストで使用されるプロトコル(httpまたはhttps)。
- ・ プロキシサーバーホストの名前
- ・ プロキシサーバーホストがリスンするポート。

連携するNGINXサーバーの設定: Black Duck

Black Duckの前にHTTPSサーバー/プロキシとして機能するNGINXサーバーがある場合は、NGINXサーバーが正しいヘッダーをBlack Duckに渡すように、NGINX構成ファイルを変更する必要があります。Black Duck により、HTTPSを使用するURLが生成されます。

 注: NGINXサーバー上の1つのサービスのみがhttpsポート443を使用できます。


Black Duckに正しいヘッダーを渡すには、nginx.config構成ファイルのlocationブロックを次のように編集します。

```
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;
}
```

プロキシサーバー/ロードバランサー構成でX-Forwarded-Prefixヘッダーが指定されている場合は、nginx.conf構成ファイルのlocationブロックを次のように編集します。


```
location/prefixPath {
    proxy_set_header X-Forwarded-Prefix "/prefixPath";
}
```

ファイルを正常にスキャンするには、contextパラメータを使用するか(コマンドライン使用時)、Black Duck Scannerの[Black DuckサーバーURL]フィールドにそれを含める必要があります。


 注：これらの手順はNGINXサーバーに該当しますが、どのタイプのプロキシサーバーでも同様の構成変更を行う必要があります。

プロキシサーバーがBlack Duckへのリクエストを書き換える場合は、次のHTTPヘッダーを使用して元のリクエスト元ホストの詳細を保持できることをプロキシサーバー管理者に知らせてください。

HTTPヘッダー	説明
X-Forwarded-Host	リクエストを行うために書き換えられたまたはルーティングされたホストのリストを追跡します。元のホストは、カンマ区切りリストの最初のホストです。 例: X-Forwarded-Host: "10.20.30.40,my.example, 10.1.20.20"
X-Forwarded-Port	元のリクエストに使用されたポートを表す1つの値が含まれます。 例: X-Forwarded-Port: "9876"
X-Forwarded-Proto	元のリクエストに使用されたプロトコルスキームを表す1つの値が含まれます。 例: X-Forwarded-Proto: "https"
X-Forwarded-Prefix	元のリクエストに使用されたプレフィックスパスが含まれます。 例: X-Forwarded-Prefix: "prefixPath" ファイルを正常にスキャンするには、contextパラメータを使用する必要があります

Amazonサービス

次の操作を実行できます。

- Black DuckをAmazon Web Services(AWS)にインストールする
AWSの詳細については、[AWSドキュメント](#)と[AMIドキュメント](#)を参照してください。
 - Black Duckで使用するPostgreSQLデータベースにAmazon Relational Database Service (RDS)を使用する。
Amazon RDSの詳細については、[Amazon Relational Database Serviceドキュメント](#)を参照してください。
-  **重要：** Synopsysでは、PostgreSQLバージョン13(外部PostgreSQLデータベース)を使用することを推奨しています。
外部PostgreSQLインスタンスでは、数字のみで構成されるユーザー名がサポートされるようになりました。

ポートに関するその他の情報

次のリストのポートは、ファイアウォールルールまたはDocker構成でブロックすることはできません。これらのポートがどのようにブロックされるかの例を次に示します。

- ホストマシン上のiptables構成。

2. インストールの計画・キープアライブ設定の構成

- ・ ホストマシン上のfirewalld構成。
- ・ ネットワーク上の別のルータ/サーバーでの外部ファイアウォール構成。
- ・ Dockerがデフォルトで作成するもの、およびBlack Duckがデフォルトで作成するもの以外の適用される特別なDockerネットワークルール。

ブロック解除されたままにする必要があるポートの完全なリストは次のとおりです。

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

キープアライブ設定の構成

net.ipv4.tcp_keepalive_time/パラメータは、確立されているTCP接続をアプリケーションがアイドル状態にしておく時間を制御します。デフォルトでは、この値は7,200秒(2時間)です。

最適なBlack Duckパフォーマンスを実現するには、このパラメータの値を600～800秒にする必要があります。

この設定は、Black Duckのインストール前またはインストール後に設定できます。

値を編集するには、次の手順を実行します。

1. /etc/sysctl.confファイルを編集します。以下に例を示します。

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

sysctlコマンドを使用してこのファイルを変更することもできます。

2. net.ipv4.tcp_keepalive_timeを追加するか(パラメータがファイルにない場合)、既存の値を編集します(パラメータがファイルにある場合)。

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

3. ファイルを保存して終了します。
4. 次のコマンドを入力して、新しい設定を読み込みます。

```
sysctl -p
```


5. Black Duckがインストールされている場合は、それを再起動します。

ナレッジベースフィードバックサービス

ナレッジベースフィードバックは、Black Duckナレッジベース(KB)機能を強化するのに使用されます。

- ・ KBによって実行されたマッチのコンポーネント、バージョン、取得元、取得元ID、またはライセンスに構成表の調整が加えられた場合、フィードバックが送信されます。

- ・ コンポーネントにマッチしないファイルが特定された場合にも、フィードバックが送信されます。手動で追加されたコンポーネントにファイルが関連付けられていない場合、フィードバックは送信されません。
- ・ フィードバックは、将来のマッチの精度を向上させるのに使用されます。この情報は、Synopsysがリソースに優先順位を付けて、お客様にとって重要なコンポーネントをより詳細に検査できるようにするのに役立ちます。

 **重要：** お客様を特定する情報はKBに送信されません。

ユーザーエージェント解析


ナレッジベースは、発信元ユーザーエージェント解析を使用して、ナレッジベースサービスのスケーラビリティを向上させ、ユーザーに対するサービス品質を向上させます。

追加のヘッダー情報により、ナレッジベースへの送信HTTPリクエストのヘッダーサイズが増加します。追加のヘッダーサイズをサポートするために、一部の中間エグレスプロキシ(顧客管理)で再構成が必要になる場合がありますが、このようになることはほとんどありません。

フィードバックサービスの無効化

ナレッジベースフィードバックサービスはデフォルトで有効です。構成表に対して実行された調整はナレッジベースに送信されます。

- ・ BLACKDUCK_KBFEEDBACK_ENABLED環境変数を使用して、フィードバックサービスを上書きできます。falseの値を指定すると、フィードバックサービスが上書きされ、構成表調整はナレッジベースに送信されません。
- ・ フィードバックサービスを無効にするには、blackduck-config.env fileにBLACKDUCK_KBFEEDBACK_ENABLED=falseを追加します。

 **注：** フィードバックサービスを再度有効にするには、値を[真]に設定します。

3. インストール: Black Duck

Black Duckをインストールする前に、次の要件を満たしていることを確認してください。

Black Duck インストール要件	
ハードウェア要件	
√	ハードウェアが最小 ハードウェア要件 を満たしていることを確認しました。
Dockerの要件	
√	システムが dockerの要件 を満たしていることを確認しました。
ソフトウェア要件	
√	システムと潜在的なクライアントが ソフトウェア要件 を満たしていることを確認しました。
ネットワーク要件	
√	ネットワークが ネットワーク要件 を満たしていることを確認しました。具体的な内容: <ul style="list-style-type: none">・ ポート443とポート55436は外部からアクセス可能です。・ サーバーは、Black Duckライセンスの検証に使用されるupdates.suite.blackducksoftware.comへのアクセス権を有しています。
データベース要件	
√	データベース構成 を選択しました。 具体的な内容: データベース設定を構成 しました(外部PostgreSQLインスタンスを使用している場合)。
プロキシ要件	
√	ネットワークが プロキシ要件 を満たしていることを確認しました。 Black Duckのインストール前またはインストール後に プロキシ設定 を構成します。
Webサーバーの要件	
√	Black Duckのインストール前またはインストール後に Webサーバーの設定 を構成します。

インストールファイル

インストールファイルはGitHubで入手できます。

オーケストレーションファイルをダウンロードします。インストール/アップグレードプロセスの一環として、これらのオーケストレーションファイルは必要なDockerイメージをプルダウンします。

tar.gzファイルのファイル名はアクセス方法によって異なりますが、内容は同じです。

GitHubページからダウンロードする場合

1. リンクを選択して、GitHubページからtar.gzファイルをダウンロードします: <https://github.com/blackducksoftware/hub>。
2. Black Duck .gzファイルを解凍します。
`gunzip hub-2022.10.0.tar.gz`

3. Black Duck.tarファイルを展開します。

```
tar xvf hub-2022.10.0.tar
```

wgetコマンドを使用してダウンロードする場合

1. 次のコマンドを実行します。

```
wget https://github.com/blackducksoftware/hub/archive/v2022.10.0.tar.gz
```

2. Black Duck .gzファイルを解凍します。

```
gunzip v2022.10.0.tar.gz
```

3. Black Duck.tarファイルを展開します。

```
tar xvf v2022.10.0.tar
```

配布

docker-swarmディレクトリは、Black Duckをインストールまたはアップグレードするのに必要な次のファイルで構成されています。

- blackduck-config.env: Black Duckの設定を構成するための環境ファイル。
- docker-compose.bdba.yml: Black DuckをBlack Duck – Binary Analysisとともにインストールし、Black Duckによって提供されたデータベースコンテナを使用する場合に使用されるDocker Composeファイル。
- docker-compose.dbmigrate.yml: Black Duckによって提供されたデータベースコンテナを使用する場合に、PostgreSQLデータベースを移行するのに使用されるDocker Composeファイル。
- docker-compose.externaldb.yml: 外部PostgreSQLデータベースで使われるDocker Composeファイル。
- docker-compose.local-overrides.yml: .ymlファイルのデフォルト設定を上書きするのに使用されるDocker Composeファイル。
- docker-compose.readonly.yml: Swarmサービスに対してファイルシステムを読み取り専用として宣言するDocker Composeファイル。
- docker-compose.yml: Black Duckによって提供されたデータベースコンテナを使用する場合のDocker Composeファイル。
- external-postgres-init.pgsql: 外部PostgreSQLデータベースの構成に使用されるPostgreSQL.sqlファイル。
- hub-bdba.env: Black Duck – Binary Analysisの追加設定が含まれている環境ファイル。このファイルを変更する必要はありません。
- hub-postgres.env: [外部PostgreSQLデータベース](#)を設定するための環境ファイル。
- hub-webserver.env: [Webサーバー設定を構成](#)するための環境ファイル。

binディレクトリは、次のファイルで構成されています。

- bd_get_source_upload_master_key.sh: [ソースファイルのアップロード](#)時にマスターとシールキーをバックアップするのに使用されるスクリプト。
- hub_create_data_dump.sh: Black Duckによって提供されたデータベースコンテナを使用する場合に、PostgreSQLデータベースをバックアップするのに使用されるスクリプト。
- hub_db_migrate.sh: Black Duckによって提供されたデータベースコンテナを使用する場合に、PostgreSQLデータベースを移行するのに使用されるスクリプト。
- hub_reportdb_changepassword.sh: [レポートデータベースのパスワードを変更](#)および設定するのに使用されるスクリプト。

3. インストール: Black Duck・インストール: Black Duck

- ・ `recover_master_key.sh`: [ソースファイルのアップロード](#)に使用する新しいシールキーを作成するスクリプト。
- ・ `system_check.sh`: [Black Duckシステム情報を収集](#)してカスタマサポートに送信するのに使用されるスクリプト。

`sizes-gen02`ディレクトリは、次のファイルで構成されています。


- ・ `resources.yaml`: これらは、拡張スキャン用のリソース定義ファイルです。

`sizes-gen03`ディレクトリは、次のファイルで構成されています。

- ・ `10sph.yaml`, `120sph.yaml`, `250sph.yaml`, `500sph.yaml`, `1000sph.yaml`, `1500sph.yaml`, `2000sph.yaml`: これらは、さまざまなスキャン/時間サイズに関するリソース定義ファイルです。詳細については、「[ハードウェア要件](#)」セクションを参照してください。

インストール: Black Duck

Black Duckをインストールする前に、構成する必要がある[設定](#)があるかどうかを確認してください。

 注: これらの手順は、Black Duckの新規インストール用です。[Black Duckのアップグレード](#)の詳細については、第6章を参照してください。

次のBlack Duckのインストール手順では、`docker`グループのユーザーまたはルートユーザーであるか、`sudo`アクセス権を持っている必要があります。ルート以外のユーザーとしてBlack Duckをインストールするには、次のセクションを参照してください。

- ・ PostgreSQLデータベースコンテナを使用してBlack Duckをインストールするには、次の手順を実行します。

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

`docker swarm init`コマンドは、単一ノードのswarmを作成します。

Black Duck 2022.4.0以降では、コンテナリソースを直接`docker-compose.yml`に割り当てなくなりました。代わりに、リソースは別の上書きファイルで指定されます。Black Duck 2022.4.0より前に使用されていたリソース割り当ては、`sizes-gen02/resources.yaml`にあります。Black Duck 2022.4.0以降では、`sizes-gen03`フォルダで複数の割り当てが可能です。1時間あたりの平均スキャン数で測定された負荷に基づいて、[7つの割り当て](#)があります。予想される負荷が事前定義された割り当てのいずれにも一致しない場合は切り上げます。

たとえば、1時間あたり約100回のスキャンが予想される場合は、次のコマンドで`sizes-gen03/120sph.yaml`を使用します。

```
docker stack deploy -c docker-compose.yml -c sizes-gen03/120sph.yaml -c docker-
compose.local-overrides.yml hub
```

ここで、

`docker-compose.yml`: ストック導入。このファイルは編集してはいけません。

`sizes-gen03/120sph.yaml`: リソース定義ファイル。この例では、`sizes-gen03/120sph.yaml`が目的のリソース定義として使用されていますが、これはスキャンパターンと使用法に合わせて変更できます。使用可能なすべてのオプションのリストについては、[配布](#)セクションを参照してください。リソース定義が大きくなるに従ってシステム要件も高度になるため、各オプションの[システム要件](#)に注意してください。

`docker-compose.local-overrides.yml`: インストール環境にシークレット、メモリ制限などのカスタム構成がある場合のオプションファイル。

- ・ PostgreSQLデータベースコンテナを使用してBlack DuckをBlack Duck - Binary Analysisとともにインストールするには、次の手順を実行します。

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

`docker swarm init`コマンドは、単一ノードのswarmを作成します。


- 外部PostgreSQLインスタンスを使用してBlack Duckをインストールするには、次の手順を実行します。

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

docker swarm initコマンドは、単一ノードのswarmを作成します。

- 外部PostgreSQLインスタンスを使用してBlack DuckをBlack Duck - Binary Analysisとともにインストールするには、次の手順を実行します。

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


 注：新しいガイドスファイル(導入サイジング用)を外部データベースで使用する場合は、導入前に、使用するtshirtサイズファイルからpostgresおよびpostgres-upgraderサービスを削除します。

docker swarm initコマンドは、単一ノードのswarmを作成します。

- Swarmサービスに対してファイルシステムが読み取り専用の状態でBlack Duckをインストールするには、前の手順にdocker-compose.readonly.ymlファイルを追加します。

たとえば、PostgreSQLデータベースコンテナを使用してBlack Duckをインストールするには、次のコマンドを入力します。

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

 注：Dockerの一部のバージョンでは、イメージがプライベートリポジトリに存在する場合に、上記のコマンドに次のフラグを追加しない限り、dockerスタックはそれらをプルしません。--with-registry-auth。


docker psコマンドを実行して各コンテナのステータスを表示することで、インストールが成功したことを確認できます。「正常」ステータスは、インストールが正常に完了したことを示します。インストール後、数分間にわたってコンテナが「開始」状態になることがあります。

Black Duckのすべてのコンテナが起動すると、Black DuckのWebアプリケーションがポート443上でDockerホストに公開されます。[ホスト名](#)を構成したことを確認します。これで、次のように入力してBlack Duckにアクセスすることができます。

<https://hub.example.com>

Black Duckに初めてアクセスすると、登録とエンドユーザーライセンス契約が表示されます。Black Duckを使用するには、条件に同意する必要があります。

提供された登録キーを入力して、Black Duckにアクセスします。

 注：再登録する必要がある場合は、エンドユーザーライセンス契約の条件に再度同意する必要があります。

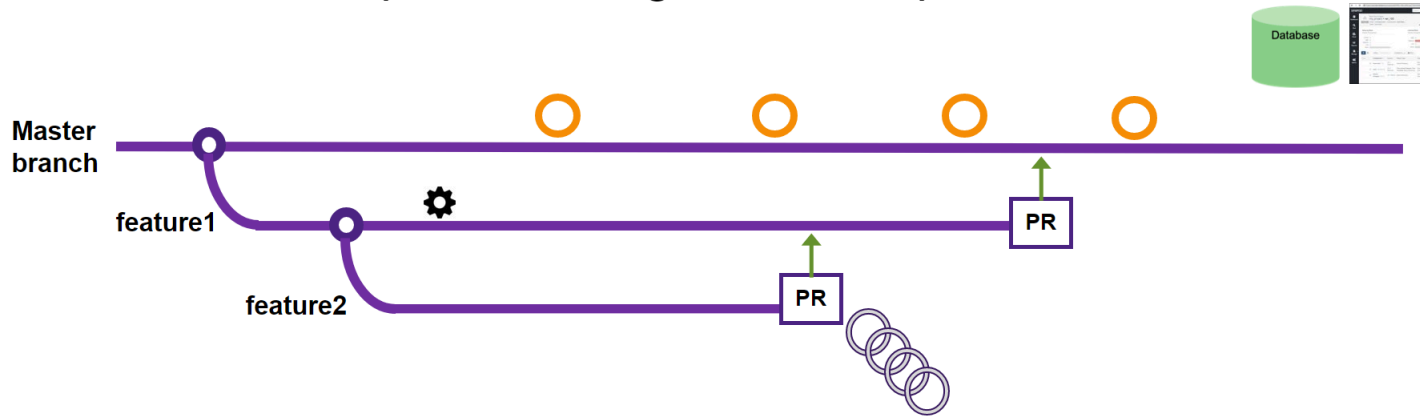
高速スキャン: Black Duck

Black Duck 高速スキャンは、Black Duck 2021.6.0でデフォルトで利用できます。

高速スキャン(依存関係スキャン)は、Black Duckで構成表を作成することなく、マージ前にコードの脆弱性やポリシー違反をチェックする効率的な方法を開発者に提供します。

高速スキャンを使用するには、Synopsys Detect 7.0.0以降を使用する必要があります。

Black Duck rapid scanning for development use cases



- Rapid Scan**
Early check for vulns or policy violations at code commit / pull request
- Full SCA Scan**
Complete SCA scan to determine all License & Security Risk

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

4. 管理タスク

この章では、次の管理タスクについて説明します。

A

- ・ [KBMATCH_SENDFPATHパラメータについて。](#)
- ・ [APIドキュメント。プロキシサーバー経由でのAPIドキュメントへのアクセス。](#)
- ・ [API、Black Duckサーバー以外からのREST APIへのアクセス。](#)

B

- ・ [バイナリスキャンファイル、サイズ値の増大](#)

C

- ・ [証明書、既存の自己署名証明書の置き換え](#)
- ・ [logstashのホスト名の変更](#)
- ・ [マップされていないコードの場所のクリーンアップ](#)
- ・ [Bearerトークンの有効期限の変更](#)
- ・ [スタックしている構成表イベントのクリア](#)
- ・ [HTMLレポートのダウンロードサイズの設定](#)

D

- ・ [重複している構成表の検出](#)

H

- ・ [階層構成表、有効化](#)

I

- ・ [無視されたコンポーネント、無視されたコンポーネントをレポートに含める](#)

L

- ・ [LDAP、セキュリティ保護されたLDAPの設定。](#)
- ・ [ログファイル、アクセス](#)

4. 管理タスク・M

M

- ・ [メモリ制限、デフォルトのメモリ制限の変更](#)
- ・ [ナレッジベースからの移行データの取得](#)

O

- ・ [上書きファイル、使用](#)

P

- ・ [PostgreSQL、外部PostgreSQLインスタンスの構成。](#)
- ・ [プロキシ設定、構成](#)

R

- ・ [レポートデータベース、レポートデータベースのパスワードの設定。](#)

S

- ・ [job runner、scanおよびbinaryscannerコンテナのスケーリング。](#)
- ・ [スキャン監視の設定](#)
- ・ [シングルサインオン\(SSO\)、設定](#)
- ・ [ソースファイル、アップロード](#)
- ・ [起動または停止 Black Duck](#)
- ・ [サポート、サポートへのBlack Duckシステム情報の提供。](#)
- ・ [sysadminユーザー、デフォルトのsysadminユーザーについて。](#)

T

- ・ [タイムゾーン、コンテナのタイムゾーンの設定](#)

U

- ・ [使用法、デフォルトの使用法の変更](#)
- ・ [ユーザーID、Black DuckコンテナのユーザーIDのカスタマイズ](#)

W

- ・ [Webサーバー設定、Webサーバー設定の構成、ホスト名、ホストポート、IPv6の無効化など。](#)

環境ファイルの使用

一部の構成では環境ファイルを使用します(例: Webサーバー設定、プロキシ設定、または外部PostgreSQL設定の構成)。これらの設定を構成する環境ファイルは、docker-swarmディレクトリにあります。

環境ファイルを使用する設定を構成するには、次の手順を実行します。

- ・ Black Duckをインストールする前に構成設定を行うには、以下の説明に従ってファイルを編集し、変更を保存します。
- ・ Black Duckをインストールした後に既存の設定を変更するには、設定を変更してから、スタック内の[サービスを再導入](#)します。

環境変数とバイナリのスキャン

Black Duck – Binary Analysis (BDBA)を使用してバイナリをスキャンする場合は、構成表にバージョンのないコンポーネントが表示されるように、Job Runnerコンテナ環境変数にHUB_SCAN_ALLOW_PARTIAL='true'パラメータが追加されていることを確認する必要があります。BDBAスキャナは、Black Duckのスキャンとは異なり、バージョン文字列情報がバイナリで識別できない場合にバージョンのないコンポーネントを表示します。構成表では、コンポーネ

ントの名前の横に疑問符(?)が表示され、コンポーネントにセキュリティ脆弱性を割り当てる前に、このコンポーネントを確認する必要があることが示されます。これは、Black Duckでは、セキュリティ脆弱性をコンポーネントにマッピングするにはバージョンが必要であるためです。


重複している構成表の検出

スキャンパフォーマンスを向上させるため、重複している構成表の検出機能はデフォルトで有効になっています。

この機能により、スキャンによって既存の構成表と同一の構成表が生成されると判断された場合、構成表の計算がスキップされます。無効にするには、次の設定を使用します。

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


この設定は、blackduck-config.envファイルで変更できます。

 注: Black Duck 2021.4.0では、Detectによって検出された一連の依存関係が前のスキャンのセットと同一の場合にのみ、この機能がパッケージマネージャ(依存関係)スキャンに影響を与えます。この機能は今後のリリースで拡張される予定です。

Bearerトークンの有効期限の変更

REST APIで使用されるBearerトークンの有効期限を延長するには、docker-compose.local-overrides.ymlファイルを使用して、HUB_AUTHENTICATION_ACCESS_TOKEN_EXPIRE環境変数に新しい有効期限値(秒単位)を構成して、デフォルト設定を上書きします。

HUB_AUTHENTICATION_ACCESS_TOKEN_EXPIREプロパティは、アクセストークンの有効期限が切れるまでの秒数です。

 注: 有効期限の変更は、docker-compose.local-overrides.ymlファイルで設定を変更した後に作成されたAPIトークンに対してのみ機能します。設定を変更してサービスを再起動したときに、既存のデータベースレコード/APIトークンは設定した有効期限に更新されません。

KBMATCH_SENDFPATHパラメータについて

KBMATCH_SENDFPATH: このパラメータは、ナレッジベースで、マッチング目的および精度でファイルパスとファイル名が使用されないようにします。マッチング結果に何らかの影響を与える可能性があるため、Synopsysではこれを変更することを推奨していません。

プロキシサーバー経由でのAPIドキュメントへのアクセス

リバースプロキシを使用していて、そのリバースプロキシのサブパスの下にBlack Duckがある場合は、APIドキュメントにアクセスできるようにBLACKDUCK_SWAGGER_PROXY_PREFIXプロパティを構成します。BLACKDUCK_SWAGGER_PROXY_PREFIXの値はBlack Duckのパスです。たとえば、「https://customer.companyname.com/hub」にあるBlack Duckにアクセスした場合、BLACKDUCK_SWAGGER_PROXY_PREFIXの値は「hub」になります。

このプロパティを構成するには、docker-swarmディレクトリにあるblackduck-config.envファイルを編集します。

Black Duckサーバー以外からのREST APIへのアクセス

Black Duckサーバー以外から提供されたWebページからBlack DuckREST APIにアクセスしたい場合があります。Black Duckサーバー以外からREST APIにアクセスできるようにするには、Cross Origin Resource Sharing (CORS) を有効にする必要があります。

Black DuckインストールのCORSを有効にして構成するためのプロパティは次のとおりです。

プロパティ	説明
BLACKDUCK_HUB_CORS_ENABLED	必須。CORSを有効にするかどうかを定義します。「true」はCORSが有効であることを示します。
BLACKDUCK_CORS_ALLOWED_ORIGINS_PROP_NAME	必須。CORSの許可されているオリジン。ブラウザは、クロスオリジンリクエストを行うときにオリジンヘッダーを送信します。このオリジンは、blackduck.hub.cors.allowedOriginsプロパティにリストされている必要があります。たとえば、http://123.34.5.67:8080からページを提供するサーバーを実行している場合、ブラウザはこれをオリジンとして設定し、この

プロパティ	説明
	<p>値をプロパティに追加する必要があります。</p> <p>プロトコル、ホスト、およびポートが一致している必要があります。複数のベースオリジンURLを指定するには、カンマ区切りリストを使用します。</p>
BLACKDUCK_CORS_ALLOWED_HEADERS_PROP_NAME	オプション。リクエストの作成に使用できるヘッダー。
BLACKDUCK_CORS_EXPOSED_HEADERS_PROP_NAME	オプション。CORSをリクエストするブラウザがアクセスできるヘッダー。
BLACKDUCK_CORS_ALLOWED_ORIGIN_PATTERNS_PROP_NAME	<p>ワイルドカードパターンで宣言されたオリジンをサポートする</p> <p>BLACKDUCK_CORS_ALLOWED_ORIGIN_PATTERNS_PROP_NAME*,*の代替。</p> <p>BLACKDUCK_CORS_ALLOWED_ORIGIN_PATTERNS_PROP_NAMEは、BLACKDUCK_CORS_ALLOWED_ORIGIN_PATTERNS_PROP_NAMEを上書きします。</p>
BLACKDUCK_CORS_ALLOW_CREDENTIALS_PROP_NAME	<p>ブラウザが、クロスドメインリクエストとともにcookieなどの資格情報をアノテーション付きエンドポイントに送信する必要があるかどうかを指定します。構成された値は、プリフライトリクエストのAccess-Control-Allow-Credentials応答ヘッダーに設定されます。ALLOW_CREDENTIALS=trueおよびALLOWED_ORIGIN=*に構成することは無効です。許可されたオリジンに「ALL」構成値が必要な場合は、今後は代わりに</p>

4. 管理タスク・バイナリスキャンファイルのサイズ値の増大

プロパティ	説明
	ALLOWED_ORIGIN_PATTERNS 構成プロパティを使用して構成する必要があります。

これらのプロパティを構成するには、docker-swarmディレクトリにあるblackduck-config.envファイルを編集します。

バイナリスキャンファイルのサイズ値の増大

Black Duck - Binary Analysisを使用する場合、スキャンできるバイナリの最大サイズは6 GBです。docker-swarmディレクトリのhub_webserver.envファイルに環境変数BINARY_UPLOAD_MAX_SIZEを追加して、メガバイト単位で値を指定することにより、この制限値を大きくすることができます。

たとえば、最大バイナリスキャンを7 GBにするには、次の環境変数を追加します。

```
BINARY_UPLOAD_MAX_SIZE=7168m
```

ダッシュボードの更新間隔の構成

blackduck-config.envファイルでSEARCH_DASHBOARD_REFRESH_JOB_DUTY_CYCLE環境変数を構成して、ダッシュボードの更新間隔をスケジュールします。

許可されているSEARCH_DASHBOARD_REFRESH_JOB_DUTY_CYCLE値は次のとおりです。

- ・ デフォルト値は20です。
- ・ 最小値は10です。
- ・ 最大値は50です。

値が、許可されている値と一致しない場合は、最も近い許可されている値にリセットされます。

例:

SEARCH_DASHBOARD_REFRESH_JOB_DUTY_CYCLE=100 構成された値は、最も近い許可されている値(10、20、または50)である50にリセットされます。

SEARCH_DASHBOARD_REFRESH_JOB_DUTY_CYCLE=10 構成された値は、許可されている値である10のままになります。

SEARCH_DASHBOARD_REFRESH_JOB_DUTY_CYCLE=5 構成された値は、最も近い許可されている値(10、20、または50)である10にリセットされます。

証明書の管理

デフォルトでは、Black DuckはHTTPS接続を使用します。HTTPSの実行に使用されるデフォルトの証明書は自己署名証明書です。これは、ローカルで作成され、承認された認証局(CA)によって署名されていないことを意味します。

このデフォルトの証明書を使用する場合、Black DuckのUIにログインするにはセキュリティ例外を作成する必要があります。ブラウザは証明書の発行元を認識しないため、デフォルトでは受け入れられません。

また、スキャン時にBlack Duckサーバーに接続するときに証明書に関するメッセージを受け取ります。これは、証明書は自己署名であり、CAによって発行されたものではないため、スキャナが証明書を検証できないためです。

目的の認証局から署名付きSSL証明書を取得できます。署名付きSSL証明書を取得するには、証明書署名要求(CSR)を作成します。CAはこれを使用して、Black Duckインスタンスを実行しているサーバーを「安全」として識別する証明書を作成します。署名付きSSL証明書をCAから受信したら、自己署名証明書を置き換えることができます。


SSL証明書キーストアを作成するには、次の手順を実行します。

1. SSLキーとCSRを生成するには、コマンドラインで次のように入力します。

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

ここで、

- ・ <keyfile>は、<会社のサーバー名>.key
- ・ <keystrength>は、サイトの公開暗号化キーのサイズ
- ・ <CSRfile>は、<会社のサーバー名>.csr

 注：会社のサーバーとして入力する名前は、SSLサーバーが存在する完全なホスト名であり、組織名はドメインの「whois」レコードにあるものと同じであることが重要です。

以下に例を示します。

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

この例では、server.company.comのCSRを作成して、CAから証明書を取得します。

2. CSRを希望する方法でCAに送信します(通常はWebポータル経由)。
3. Apache Webサーバーの証明書が必要であることを示します。
4. 会社に関する要求された情報をCAに提供します。この情報は、ドメインレジストリ情報と一致する必要があります。
5. CAから証明書を受け取ったら、次のセクションの手順を実行して、証明書をBlack Duckインスタンスにアップロードします。

カスタム証明書の使用

webserverコンテナには、Dockerから取得した自己署名証明書があります。この証明書をカスタム証明書-キーペアに置き換えることができます。

1. docker secretコマンドでWEBSERVER_CUSTOM_CERT_FILEとWEBSERVER_CUSTOM_KEY_FILEを使用して、Docker Swarmに証明書とキーを通知します。シークレットの名前にスタック名を含める必要があります。次の例で、スタック名は「hub」です。

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

2. docker-compose.local-overrides.ymlファイルのwebserverサービスにシークレットを追加します。

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

3. docker-swarmディレクトリにあるdocker-compose.local-overrides.ymlファイルの末尾にあるsecretsセクションからコメント文字(#)を削除します。

```
secrets:
  WEBSERVER_CUSTOM_CERT_FILE:
    external: true
```

4. 管理タスク・ナレッジベースからのコンポーネント移行データの取得

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

4. `docker-compose.local-overrides.yml`ファイルのwebserverサービスのhealthcheckプロパティは、シークレットからの新しい証明書をポイントしている必要があります。

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

5. 次のコマンドを実行して、スタックを再導入します。

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

証明書認証でのカスタム認証局の使用

証明書認証に独自の認証局を使用できます。

カスタム認証局を使用するには、次の手順を実行します。

1. カスタム認証局証明書ファイルであるAUTH_CUSTOM_CAという名前のdocker secretを、docker-swarmディレクトリにある`docker-compose.local-overrides.yml`ファイルのwebserverおよびauthenticationサービスに追加します。

```
webserver:
  secrets:
    - AUTH_CUSTOM_CA
authentication:
  secrets:
    - AUTH_CUSTOM_CA
```

2. docker-swarmディレクトリにある`docker-compose.local-overrides.yml`ファイルの末尾に次のテキストを追加します。

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

3. webserverコンテナとauthenticationサービスを開始します。
4. Black Duckサービスが起動したら、信頼できる認証局で署名された証明書キーペアを含むJson Web Token (JWT)を返すAPIリクエストを行います。以下に例を示します。

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

 注：認証に使用される証明書のユーザー名は、Black DuckシステムにCommon Name (CN)として存在する必要があります。

ナレッジベースからのコンポーネント移行データの取得

移行APIを使用して、Black Duckナレッジベースから新規または変更されたコンポーネントIDを取得します。これらのAPIは、ナレッジベース内の移行されたコンポーネントの生データと新しいIDを返します。

APIを使用して、古いコンポーネントID (例: コンポーネントID: bf368a1d-ef4f-422c-baca-a138737595e7)を送信し、ナレッジベースから新しいコンポーネントIDを取得できます。

移行追跡APIは、特定のコンポーネントまたはコンポーネントバージョンの移行情報を取得したり、特定の日付に発生した移行の詳細を取得したりできます。

移行の記録の有効化

移行の記録を有効にするには、次で以下のプロパティを設定します: `blackduck-config.env` file.

`RECORD_MIGRATIONS = true` (デフォルトは`false`)

これにより、移行が検出されたときにレコードを書き込むことができます。

移行データの保持

レコードが返される日数を設定するには、`blackduck-config.env`ファイルで次のプロパティを設定します。

`MIGRATED_OBJECT_RETENTION_DAYS = <number_of_days>` (デフォルトは30日)

APIエンドポイント

APIドキュメントに移動して、次のAPIの使用を開始します。

特定の日付以降に発生した移行の場合:

`GET /api/component-migrations`

特定のコンポーネントまたはコンポーネントバージョンの移行の場合:

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

詳細については、『Black Duck APIドキュメント』(https://<blackduck_server>/api-doc/public.html#_component_component_version_migration_endpoints)を参照してください。

階層構成表の有効化

デフォルトでは、階層構成表は無効になっています。この機能を有効にするには、`HUB_HIERARCHICAL_BOM`環境変数を`.env`ファイルに追加します。値を「`true`」に設定します。例: `HUB_HIERARCHICAL_BOM=true`。値を「`false`」にリセットすると、この機能が無効になります。

`docker-swarm`ディレクトリにある `docker-compose.local-overrides.yml`ファイルでwebappサービスを編集することもできます。

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

環境変数が複数ある場合は、カンマ(,)で区切ります。以下に例を示します。

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

無視されたコンポーネントをレポートに含める

デフォルトでは、無視されたコンポーネントとそれらの無視されたコンポーネントに関連付けられた脆弱性は、脆弱性ステータスレポート、脆弱性更新レポート、脆弱性修正レポート、およびプロジェクトバージョンレポートから除外されます。無視されたコンポーネントを含めるには、`docker-swarm`ディレクトリにある`blackduck-config.env`ファイルで`BLACKDUCK_REPORT_IGNORED_COMPONENTS`環境変数の値を「`true`」に設定します。


`BLACKDUCK_REPORT_IGNORED_COMPONENTS`の値を「`false`」にリセットすると、無視されたコンポーネントが除外されます。

セキュリティ保護されたLDAPの設定

セキュリティで保護されたLDAPサーバーをBlack Duckに接続するときに証明書の問題が発生する場合は、セキュリティで保護されたLDAPサーバーへの信頼関係接続をBlack Duckサーバーが設定していないことが考えられます。これは通常、自己署名証明書を使用している場合に発生します。

セキュリティで保護されたLDAPサーバーの信頼関係接続を設定するには、次の方法でローカルBlack Duck LDAPトラストストアにサーバー証明書をインポートします。

1. LDAP情報を取得します。
2. Black DuckのUIを使用してサーバー証明書をインポートします。

 注：ホストされるすべてのお客様は、SAMLまたはLDAPを介したシングルサインオン(SSO)の既成サポートを活用して、Black Duckアプリケーションへのアクセスのセキュリティを確保する必要があります。これらのセキュリティ機能を有効にして設定する方法については、インストールガイドを参照してください。さらに、2ファクタ認証を提供しているSAML SSOプロバイダを使用しているお客様は、そのテクノロジーを有効にして活用し、Black Duckアプリケーションへのアクセスのセキュリティをさらに高めることをお勧めします。

LDAP情報の取得

LDAP管理者に連絡し、次の情報を収集します。

LDAPサーバーの詳細

Black Duckがディレクトリサーバーに接続するために使用する情報です。

- ・ (必須) インスタンスがリスンするディレクトリサーバーのホスト名またはIPアドレス(プロトコルスキームとポートを含む)です。
例: `ldaps://<server_name>.<domain_name>.com:339`
- ・ (オプション) 組織で匿名認証を使用せず、LDAPアクセスに認証情報が必要な場合は、ディレクトリサーバーの読み取り権限を持っているユーザーのパスワードと、LDAP名または絶対LDAP識別名(DN)のいずれかです。
絶対LDAP DNの例: `uid=ldapmanager,ou=employees,dc=company,dc=com`
LDAP名の例: `jdoe`
- ・ (オプション) LDAPアクセスに認証情報が必須である場合は、使用する認証タイプ(simpleまたはdigest-MD5)です。

LDAPユーザー属性

ディレクトリサーバー内のユーザーを見つけるためにBlack Duckで使われる情報です。

- ・ (必須) ユーザーを見つけるために使用できる絶対ベースDNです。
例: `dc=example,dc=com`
- ・ (必須) 特定の一意ユーザーとのマッチングに使用される属性です。この属性の値により、その名前のユーザーのユーザープロフィールアイコンがパーソナライズされます。
例: `uid={0}`


テスト用ユーザー名とパスワード

- ・ (必須) ディレクトリサーバーへの接続をテストするためのユーザーの認証情報

サーバー証明書のインポート

サーバー証明書をインポートするには、次の手順を実行します。

1. システム管理者としてBlack Duckにログインします。

2.  をクリックします。
3. [システム設定]を選択します。
4. [ユーザー認証]をクリックします。
5. [LDAP構成を有効にする]チェックボックスをオンにして、上で説明したように[LDAPサーバーの詳細]セクションに情報を入力します。[サーバーURL]フィールドで、セキュリティで保護されたLDAPサーバー（プロトコルスキームはldaps://）が構成されていることを確認します。
6. 上記の説明に従って、[LDAPユーザー属性]セクションに情報を入力します。
必要に応じて、[Black Duckで自動的にユーザーアカウントを作成する]チェックボックスをオフにして、LDAPで認証されたユーザーの自動作成をオフにします。このチェックボックスはデフォルトでオンになっているため、Black Duckに存在しないユーザーがLDAPを使用してBlack Duckにログインすると、ユーザーが自動的に作成されます。これは、新規インストールとアップグレードに適用されます。
7. [テスト接続、ユーザー認証およびフィールドマッピング]セクションにユーザーの認証情報を入力し、[接続のテスト]をクリックします。
8. 証明書に問題がない場合は、自動的にインポートされ、「接続テストに成功しました」というメッセージが表示されます。

Test Connection, User Authentication and Field Mapping

Tests ability to connect and authenticate test-user. Note: test-user credentials are not saved.

✓ Connection Test Succeeded

Test Username *

Denis Bors

Test Password *

.....

Reset

Test Connection

9. 証明書に問題がある場合、ダイアログボックスに証明書に関する詳細が表示されます。次のいずれかを実行します。
 - ・ [キャンセル]をクリックして、証明書の問題を解決します。
解決したら、接続を再度テストし、証明書の問題が解決して証明書がインポートされたことを確認します。成功した場合は、「接続テストに成功しました」というメッセージが表示されます。
 - ・ [保存]をクリックしてこの証明書をインポートします。
[接続のテスト]をクリックして、証明書がインポートされたことを確認します。成功した場合は、「接続テストに成功しました」というメッセージが表示されます。

LDAP信頼ストアのパスワード

カスタムのBlack Duck Webアプリケーション信頼ストアを追加する場合は、これらの方法を使用してLDAP信頼ストアのパスワードを指定します。

Docker Swarmを使用する場合は、次の方法を使用してLDAP信頼ストアのパスワードを指定します。

- ・ `docker secret`コマンドでLDAP_TRUST_STORE_PASSWORD_FILEを使用してDocker Swarmにパスワードを通知します。シークレットの名前にスタック名を含める必要があります。この例では、「HUB」がスタック名です。

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

`docker-swarm`ディレクトリにある`docker-compose.local-overrides.yml`ファイルで、パスワードシークレットをwebappサービスに追加します。


```
secrets:
  - LDAP_TRUST_STORE_PASSWORD_FILE
```

`docker-compose.local-overrides.yml`ファイルの末尾にある`secrets` セクションに、次のようなテキストを追加します。

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

- ・ `docker-swarm`ディレクトリにある`docker-compose.local-overrides.yml`ファイルにwebappサービスのボリュームセクションを追加して、LDAP_TRUST_STORE_PASSWORD_FILEというファイルが含まれているディレクトリを`/run/secrets`にマウントします。

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

 注： 信頼ストアが完全に置き換えられ、別のパスワードで保護されている場合にのみ、LDAP_trust_store_password_fileが含まれているディレクトリをマウントする必要があります。

ログファイルとヒートマップデータのダウンロード

問題のトラブルシューティングやカスタマサポートへのログファイルの提供が必要になる場合があります。システム管理者の役割のあるユーザーは、現在のログファイルまたはシステムのヒートマップデータが格納されている.zipファイルをダウンロードできます。

ログファイルまたはヒートマップファイルの準備には数分かかる場合があることに注意してください。ログ取得とヒートマップデータの設定の詳細については、『インストールガイド』を参照してください。

ログファイルへのアクセス

Black DuckのUIからログファイルをダウンロードするには、次の手順を実行します。

1. システム管理者の役割でBlack Duckにログインします。


2.  をクリックします。

3. [診断]を選択します。
4. [システム情報]タブを選択します。
5. [ログのダウンロード(.zip)]をクリックします。

ヒートマップデータへのアクセス

ヒートマップを圧縮CSVとしてダウンロードして端末スキャンの傾向を確認および分析し、スプレッドシートプログラムのピボットとしてヒートマップを作成することができます。

システムのヒートマップデータをダウンロードするには:

1. システム管理者の役割でBlack Duckにログインします。
2.  をクリックします。
3. [診断]を選択します。
4. [システム情報]タブを選択します。
5. [ヒートマップのダウンロード(.zip)]をクリックします。

ログファイルの表示

ログの取得

コンテナからログを取得するには、次の手順を実行します。

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

ここで、「logs/」は、ログのコピー先のローカルディレクトリです。

コンテナのログファイルの表示

docker-compose logsコマンドを使用して、すべてのログを表示します。

```
docker-compose logs
```

Dockerコマンドの詳細については、DockerドキュメントWebサイト(<https://docs.docker.com/>)を参照してください。

ログのページ

デフォルトでは、ログファイルは14日後に自動的にページされます。この値を変更するには、次の手順を実行します。


1. コンテナを停止します。
2. docker-swarmディレクトリにあるdocker-compose.local-overrides.ymlファイルを編集します。
 - a. logstashサービスを追加します。
 - b. 新しい値でDAYS_TO_KEEP_LOGS環境変数を追加します。次の例では、10日後にログファイルがページされます。

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

3. コンテナを再起動します。

デフォルトのメモリ制限の変更

Black Duckにかかる負荷によっては、一部のコンテナでデフォルトのメモリ制限よりも高いメモリ制限が必要になることがあります。

 注: デフォルトのメモリ制限は、絶対に引き下げないでください。引き下げると、Black Duckが正しく機能しなくなります。

4. 管理タスク・デフォルトのメモリ制限の変更

次のコンテナのデフォルトのメモリ制限を変更できます。

- webapp
- jobrunner
- scan
- binaryscanner
- bomengine

webappコンテナのデフォルトのメモリ制限の変更

webappコンテナには、次の3つのメモリ設定があります。

- HUB_MAX_MEMORY環境変数は、最大Javaヒープサイズを制御します。
- limits memoryとreservations memory設定は、Dockerがwebappコンテナの全体的なメモリをスケジュールおよび制限するのに使用する制限を制御します。
 - limits memory設定は、コンテナが使用できるメモリ容量です。
 - Dockerはreservations memory設定を使用して、コンテナをマシンに導入(スケジュール)できるかどうかを判断します。この値を使用して、Dockerは、すべてのコンテナが同じメモリを求めて競合するのではなく、マシンに導入されたすべてのコンテナに十分なメモリがあることを確認します。

これらの各設定の値は、最大Javaヒープサイズよりも大きくする必要があります。Black Duck Softwareでは、Javaヒープサイズを更新する場合は、limits memoryとreservations memoryの値をそれぞれ最大Javaヒープサイズよりも少なくとも1 GB大きく設定することを推奨しています。

docker-compose.local-overrides.ymlファイルのwebappセクションを使用し、必要に応じてコメント文字(#)を削除して、新しい値を入力します。

次の例では、Web Appコンテナの最大Javaヒープサイズが8 GBに変更され、limit memoryとreservations memory設定の値がそれぞれ9 GBに変更されます。

元の値:

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

更新された値:

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


jobrunnerコンテナのデフォルトのメモリ制限の変更

jobrunnerコンテナには、次の3つのメモリ設定があります。

- HUB_MAX_MEMORY環境変数は、最大Javaヒープサイズを制御します。

- limits memoryとreservations memory設定は、Dockerがjobrunnerコンテナの全体的なメモリをスケジュールおよび制限するのに使用する制限を制御します。
 - limits memory設定は、コンテナが使用できるメモリ容量です。
 - Dockerはreservations memory設定を使用して、コンテナをマシンに導入（スケジュール）できるかどうかを判断します。この値を使用して、Dockerは、すべてのコンテナが同じメモリを求めて競合するのではなく、マシンに導入されたすべてのコンテナに十分なメモリがあることを確認します。

これらの各設定の値は、最大Javaヒープサイズよりも大きくする必要があります。Black Duck Softwareでは、Javaヒープサイズを更新する場合は、limits memoryとreservations memoryの値をそれぞれ最大Javaヒープサイズよりも少なくとも1 GB大きく設定することを推奨しています。

 注：これらの設定は、スケーリングされたJob Runnerコンテナを含む、すべてのJob Runnerコンテナに適用されます。

docker-compose.local-overrides.ymlファイルのjobrunnerセクションを使用し、必要に応じてコメント文字(#)を削除して、新しい値を入力します。

次の例では、jobrunnerコンテナの最大Javaヒープサイズが8 GBに変更され、limit memoryとreservations memory設定の値がそれぞれ9 GBに変更されます。

元の値:

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

更新された値:

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

デフォルトのメモリ制限の変更

HUB_MAX_MEMORY値を変更するときにコンテナにメモリを追加した場合、次の値が変更されることがあります。

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

ガイドンスの例


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


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

Large/X-Large
HUB_MAX_MEMORY: 12288m
```

4. 管理タスク・デフォルトのメモリ制限の変更

```
org.quartz.scheduler.periodic.maxthreads: 8
org.quartz.scheduler.periodic.prefetch: 4
org.quartz.scheduler.ondemand.maxthreads: 24
org.quartz.scheduler.ondemand.prefetch: 16
```

 注: LargeとX-Largeの違いは、jobrunnerインスタンスの数だけです。メモリとスレッドの数は同じです。


 注: 両方のプールの最大スレッド数は32を超えてはいけません

scanコンテナのデフォルトのメモリ制限の変更

scanコンテナには、次の3つのメモリ設定があります。

- HUB_MAX_MEMORY環境変数は、最大Javaヒープサイズを制御します。
- limits memoryとreservations memory設定は、DockerがScanコンテナの全体的なメモリをスケジュールおよび制限するのに使用する制限を制御します。
 - limits memory設定は、コンテナが使用できるメモリ容量です。
 - Dockerはreservations memory設定を使用して、コンテナをマシンに導入(スケジュール)できるかどうかを判断します。この値を使用して、Dockerは、すべてのコンテナが同じメモリを求めて競合するのではなく、マシンに導入されたすべてのコンテナに十分なメモリがあることを確認します。

これらの各設定の値は、最大Javaヒープサイズよりも大きくする必要があります。Black Duck Softwareでは、Javaヒープサイズを更新する場合は、limits memoryとreservations memoryの値をそれぞれ最大Javaヒープサイズよりも少なくとも1 GB大きく設定することを推奨しています。

 注: これらの設定は、スケーリングされたScanコンテナを含む、すべてのScanコンテナに適用されます。

docker-compose.local-overrides.ymlファイルのscanセクションを使用し、必要に応じてコメント文字(#)を削除して、新しい値を入力します。

次の例では、scanコンテナの最大Javaヒープサイズが4 GBに増やされ、limits memoryとreservations memory設定の値がそれぞれ5 GBに増やされます。

元の値:


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

更新された値:

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

binaryscannerコンテナのデフォルトのメモリ制限の変更

binaryscannerコンテナの唯一のデフォルトのメモリサイズは、コンテナの実際のメモリ制限です。

 注: これらの設定は、スケーリングされたbinaryscannerコンテナを含む、すべてのbinaryscannerコンテナに適用されます。

docker-compose.local-overrides.ymlファイルにbinaryscannerセクションを追加します。

次の例では、コンテナのメモリ制限が4 GBに変更されます。

更新された値:

```
binaryscanner:
```




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

bomengineコンテナのデフォルトのメモリ制限の変更

bomengineコンテナには、次の3つのメモリ設定があります。

- ・ HUB_MAX_MEMORY環境変数は、最大Javaヒープサイズを制御します。
- ・ limits memoryとreservations memory設定は、Dockerがbomengineコンテナの全体的なメモリをスケジュールおよび制限するのに使用する制限を制御します。
 - ・ limits memory設定は、コンテナが使用できるメモリ容量です。
 - ・ Dockerはreservations memory設定を使用して、コンテナをマシンに導入（スケジュール）できるかどうかを判断します。この値を使用して、Dockerは、すべてのコンテナが同じメモリを求めて競合するのではなく、マシンに導入されたすべてのコンテナに十分なメモリがあることを確認します。

 注：これらの設定は、スケーリングされたbomengineコンテナを含む、すべてのbomengineコンテナに適用されます。

docker-compose.local-overrides.ymlファイルのbomengineセクションを使用し、必要に応じてコメント文字(#)を削除して、新しい値を入力します。


次の例では、bomengineコンテナの最大Javaヒープサイズが8 GBに変更され、limits memoryとreservations memory設定の値がそれぞれ9 GBに変更されます。

元の値:

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

更新された値:

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

 重要：bomengineコンテナのHUB_MAX_MEMORYとlimits memoryには、jobrunnerコンテナと同じ値を割り当てることをお勧めします。

logstashのホスト名の変更

logstashのホスト名とサービス名を変更しても、内部PostgreSQLコンテナにはプッシュされません。たとえば、logstashを別の目的で使用していて、PostgreSQLログをbdlogstashに書き込むために、logstashホスト名をbdlogstashに変更したい場合は、次の手順を実行します。

1. blackduck-config.envで、logstashをbdlogstashに変更します: HUB_LOGSTASH_HOST=bdlogstash。
2. docker-compose.ymlを編集して、logstashをbdlogstashに変更します。

bdlogstash:

```
image: blackducksoftware/blackduck-logstash:1.0.9
volumes: ['log-volume:/var/lib/logstash/data']
env_file: [blackduck-config.env]
```

4. 管理タスク・マップされていないコードの場所のクリーンアップ

3. `docker-compose.yml`の`postgres`コンテナ定義に`env_file: [blackduck-config.env]`を追加して、ホスト名の変更が読み取られるようにします。

`env_file: [blackduck-config.env]`

マップされていないコードの場所のクリーンアップ

マップされていないコードの場所は、プロジェクトバージョンにマップされていないコードの場所です。Black Duckの内部では、コードの場所は1つまたは複数のスキャンによって表されます。

- ・ 場合によっては、コードスキャンが、コードの場所をプロジェクトバージョンにマッピングせずにコードの場所を作成することがあり、この結果、マップされていないコードの場所になります。
- ・ プロジェクトバージョンを削除すると、コードの場所/スキャンデータが孤立する可能性があります。

マップされていないコードの場所のデータをパージしたくない場合は、`blackduck-config.env`ファイルでこの機能を無効にすることができます。これはデフォルトで`true`に設定されており、30日ごとにパージするように設定されています。

定期的にスキャンし、頻繁にデータを破棄したい場合は、保持期間を14日間などの短い日数に設定することをお勧めします。


マップされていないコードの場所のクリーンアップをスケジュールするには、`blackduck-config.env`ファイルで次のプロパティを設定します。

- ・ `BLACKDUCK_HUB_UNMAPPED_CODE_LOCATION_CLEANUP=true`

デフォルト設定は`true`です。

- ・ `BLACKDUCK_HUB_UNMAPPED_CODE_LOCATION_RETENTION_DAYS=<number of days between 1-365, for example, 14>`

デフォルト設定は30日です。


 注：マップされていないコードの場所のクリーンアップを構成してシステムを再起動すると、スキャンパージが開始され、保持条件（構成された保持日数よりも古い）を満たすマップされていないコードの場所が削除されます。デフォルトでは、スキャンパージジョブは15分ごとに実行されます。

cron文字列を使用したスキャンパージジョブのスケジュール

`docker swarm`実装の`blackduck-config.env`ファイルに変数を設定することで、スキャンパージを構成できます。

次のいずれかの方法を使用して、`blackduck-config.env`で変数を設定することにより、スキャンパージジョブをスケジュールできます。

- ・ cronジョブを使用する場合：`blackduck.scan.processor.scanpurge.cronstring`
次のcron形式を使用します。second minute hour dayOfMonth month daysOfWeek
- ・ 固定遅延を使用する場合：`blackduck.scan.processor.scanpurge.fixeddelay`（`scanpurgejob`を実行する頻度をミリ秒単位で設定します。デフォルトは15分です）

 注：`blackduck.scan.processor.scanpurge.cronstring`を指定した場合、代わりにcron文字列が使用されるため、`blackduck.scan.processor.scanpurge.fixeddelay`設定は無視されます。

スタックしている構成表イベントのクリア

構成表イベントクリーンアップジョブは、処理エラーのためにスタックしている可能性がある構成表イベントをクリアします。デフォルトでは、ジョブは毎日午前0時に実行され、過去24時間より前にスタックしたイベントを削除しま

す。cronスケジュールは、システムの使用率が低い時間帯に応じて変更できます。また、過去何時間のイベントを保持するかの設定を1〜48時間の間で変更できます。

- 過去何時間の構成表イベントを保持するかを設定します。これは、処理エラーまたはトポロジ変更によってスタックしている可能性のある構成表イベントをパージするのに役立ちます。デフォルトは24時間です。有効な範囲は1〜48時間です。たとえば、過去12時間より前にスタックしたすべてのイベントを削除する場合は、次のように指定します。

```
BLACKDUCK_BOM_EVENT_CLEANUP_BEFORE_HOURS=12
```

- Cron式を介して、構成表イベントをクリアするジョブの実行スケジュールを設定します。システムの使用率が低い時間帯に実行することをお勧めします。デフォルトでは、午前0時に実行され、値は0 0 * * * ?です。たとえば、構成表イベントクリーンアップジョブを毎日午前2時に実行する必要がある場合は、次のように指定します。

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

VersionBomEventCleanupJobはデフォルトで有効になっており、このジョブを無効にすることはできません。

上書きファイルの使用

Black Duckで使用されるデフォルト設定の一部を上書きすることができます。.yamlファイルを直接編集する代わりに、docker-swarmディレクトリにあるdocker-compose.local-overrides.yamlを使用します。

このファイルを使用してデフォルト設定を変更することで、アップグレード時に変更内容が保持され、Black Duckをアップグレードするたびに.yamlファイルを変更する必要がなくなります。

docker-composeコマンドで、docker-compose.local-overrides.yamlファイルは最後に使用される.yamlファイルである必要があります。たとえば、次のコマンドは、外部データベースを使用してBlack Duckを起動します。

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


分析の構成: Black Duck

Black Duckでは、blackduck-config.envファイルで解析をオフにすることで、Synopsis Detectの自動実行をグローバルに無効にできます。

- blackduck-config.envファイルで、ANALYTICS=falseのように構成します。
- Black Duckを再起動します。

外部PostgreSQLインスタンスの構成

Black Duck は、外部PostgreSQLインスタンスの使用をサポートしています。

 注: azure.extensions値をPGCRYPTOに設定すると、initスクリプトがAzureデータベースにアクセスできるようになります。


Synopsysでは、外部PostgreSQLを使用する新規インストールには、PostgreSQL 13.4(またはバージョン13.x以降)を推奨しています。PostgreSQL 9.6外部データベースのフルサポートは、2021.6.0で削除されました。PostgreSQL 11.xは、外部データベースユーザーに対してもサポートされています。

- Community PostgreSQLのユーザーは、PostgreSQLのアップグレードについてPostgreSQL 11の[手順](#)に従う必要があります。
- サードパーティのPostgreSQL(RDSなど)のユーザーは、プロバイダの指示に従う必要があります。
- PostgreSQL 9.6を使用する内部PostgreSQLコンテナのユーザーは影響を受けず、変更を行う必要はありません。PostgreSQL 11.12は、内部PostgreSQLコンテナではまだサポートされていません。

4. 管理タスク・外部PostgreSQLインスタンスの構成

- ・ Black Duckバージョン4.0.0より前のBlack Duckを使用している場合は、2020.6.0以降にアップグレードする前に2020.4にアップグレードする必要があります。
- ・ 数字のみで構成される数値のユーザー名は、外部PostgreSQLインスタンスのPostgreSQLユーザー名に使用できません。

更新されたexternal-postgres-init.pgsqlスクリプトでは、数値のPostgreSQLユーザー名がスクリプトで正常に実行されるように、ユーザー名を二重引用符で囲みます。external-postgres-init.pgsqlスクリプトで、HUB_Postgre_USERおよびPostgreSQL_USERのデフォルトの名前値を検索して、数値のユーザー名に置き換えます。

 **重要：** PostgreSQL 10.xまたはPostgreSQL 12.xはサポートされていません。

外部PostgreSQLデータベースを構成するには、次の手順を実行します。

1. 外部PostgreSQLクラスタをUTF8エンコードで初期化します。これを行う方法は、外部PostgreSQLプロバイダによって異なる場合があります。たとえば、PostgreSQL initdbツールを使用する場合は、次のコマンドを実行します。

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

2. データベースのユーザー名とパスワードを作成および設定します。PostgreSQLデータベースには、管理者（デフォルトでは、blackduckがユーザー名）、ユーザー（デフォルトでは、blackduck_userがユーザー名）、Black Duckレポートデータベースのユーザー（デフォルトでは、blackduck_reporterがユーザー名）の3名のユーザーが存在します。次の操作を実行できます。

- ・ [デフォルトのユーザー名でアカウントを作成する。](#)
- ・ [カスタムユーザー名でアカウントを作成する。](#)

3. [PostgreSQLインスタンスを構成します。](#)

デフォルトのユーザー名を使用したアカウントの作成と構成：

デフォルトのblackduck、blackduck_user、およびblackduck_reporterのユーザー名を使用する場合は、次の手順を実行します。

これらの手順を完了したら、「[PostgreSQLインスタンスの構成](#)」に進みます。

1. 管理者権限を持つblackduckという名前のデータベースユーザーを作成します。

Amazon RDSの場合は、データベースインスタンスの作成時に「マスターユーザー」をblackduckに設定します。

その他の特定の値は必要ありません。


2. ディレクトリにあるexternal-postgres-init.pgsqlファイルで以下を docker-swarm

置き換えます。

POSTGRES_USERを次と置き換えます： blackduck

HUB_POSTGRES_USERを次と置き換えます： blackduck_user

BLACKDUCK_USER_PASSWORDを次に使用するパスワードに置き換えます： blackduck_user

 **重要：** この手順は必須です。


3. docker-swarmディレクトリにあるexternal-postgres-init.pgsqlスクリプトを実行して、ユーザー、データベース、およびその他の必要なアイテムを作成します。以下に例を示します。

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

4. 任意のPostgreSQL管理ツールを使用して、blackduck、blackduck_user、およびblackduck_reporterデータベースユーザーのパスワードを構成します。

これらのユーザーは、前の手順でexternal-postgres-init.pgsqlスクリプトによって作成されています。

5. 「PostgreSQLインスタンスの構成」に進みます。

 注：Black Duckはpgcrypto拡張機能を使用し、それが使用可能であると想定します。CloudSQL、RDS、およびAzureは、デフォルトでpgcrypto拡張機能を提供しているため、さらなる構成は必要ありません。Community PostgreSQLのユーザーは、postgresql-contribパッケージがまだインストールされていない場合は、それをインストールする必要があります。パッケージ名はディストリビューションによって異なります。

カスタムユーザー名とパスワードを使用したアカウントの作成と構成：

カスタムデータベースのユーザー名を作成する場合は、次の手順を実行します。

次の手順で、各ユーザー名は次のとおりです。

- ・ DBAdminNameは、新しいカスタム管理者のユーザー名。
- ・ DBUserNameは、新しいカスタムデータベースユーザーのユーザー名。
- ・ DBReporterNameは、新しいカスタムデータベースレポーターのユーザー名。

これらの手順を完了したら、次のセクション「PostgreSQLインスタンスの構成」に進みます。

1. 管理者権限を持つDBAdminNameという名前のデータベースユーザーを作成します。

Amazon RDSの場合は、データベースインスタンスの作成時に「マスターユーザー」をDBAdminNameに設定します。

その他の特定の値は必要ありません。

2. docker-swarmディレクトリにあるexternal-postgres-init.pgsqlスクリプトを編集し、DBAdminName、DBUserName、およびDBReporterNameに使用するアカウント名を指定します。
3. docker-swarmディレクトリにある編集済みのexternal-postgres-init.pgsqlスクリプトを実行して、ユーザー、データベース、およびその他の必要なアイテムを作成します。以下に例を示します。

```
psql -U DBAdminName -h <hostname> -p <port> -f external-postgres-init.pgsql postgres
```
4. 任意のPostgreSQL管理ツールを使用して、DBAdminName、DBUserName、およびDBReporterNameデータベースユーザーのパスワードを構成します。

これらのユーザーは、前の手順でexternal-postgres-init.pgsqlスクリプトによって作成されています。

5. hub-postgres.env環境ファイルを編集します。このファイルには、HUB_Postgre_USERおよびHUB_Postgre_ADMINのデフォルトのユーザー名がリストされています。これらのデフォルト値を、データベースユーザーおよび管理者のカスタムユーザー名に置き換えます。
6. 次のセクション「PostgreSQLインスタンスの構成」に進みます。

PostgreSQLインスタンスの構成：

ユーザーを作成してパスワードを構成したら、次の手順を実行します。

1. docker-swarmディレクトリにあるhub-postgres.env環境ファイルを編集して、データベース接続パラメータを指定します。次のことを行えます。
 - ・ データベース接続でSSLを有効にする。
 認証に、証明書またはユーザー名とパスワード、あるいはその両方を使用することを選択できます。
 - ・ データベース接続でSSLを無効にする。
 SSLが無効の場合は、ユーザー名とパスワードによる認証を使用する必要があります。

パラメータ	説明
HUB_Postgre_ENABLE_SSL	データベース接続でのSSLの使用を定義します。

4. 管理タスク・外部PostgreSQLインスタンスの構成

パラメータ	説明
	<ul style="list-style-type: none"> データベース接続でSSLを使用しないように設定するには、値を「false」に設定します。 データベース接続でSSLを使用するように設定するには、値を「true」に設定します。
HUB_Postgre_ENABLE_SSL_CERT	<p>認証に証明書が必要かどうかを定義します。</p> <ul style="list-style-type: none"> クライアント証明書認証を無効にするには、値を「false」に設定します。 データベース接続でSSLを使用するときにクライアント証明書を使用するには、値を「true」に設定します。
HUB_Postgre_HOST	PostgreSQLインスタンスを含むサーバーのホスト名。
HUB_Postgre_PORT	PostgreSQLインスタンスの接続先のデータベースポート。

2. ユーザー名とパスワードによる認証を使用している場合は、PostgreSQL管理者とユーザーのパスワードをBlack Duckに提供します。

- a. データベースユーザーのパスワードが含まれたHUB_Postgre_USER_PASSWORD_FILEという名前のファイルを作成します。デフォルトのユーザー名を使用している場合、これはblackduck_userのユーザー名で、前の例ではDBUserNameです。
- b. データベース管理者ユーザーのパスワードが含まれたHUB_Postgre_ADMIN_PASSWORD_FILEという名前のファイルを作成します。デフォルトのユーザー名を使用している場合、これはblackduckのユーザー名で、前の例ではDBAdminNameです。
- c. 両方のファイルが含まれているディレクトリを/run/secretsにマウントします。docker-swarmディレクトリにあるdocker-compose.local-overrides.ymlファイルを使用します。各サービス（webapp、jobrunner、authentication、bomengine、およびscan）で、次の手順を実行します。

1. 必要に応じて、サービス名の前にあるコメント文字(#)を削除します。
2. サービスにマウントするボリュームを追加します。

次の例では、webappサービスにボリュームが追加されます。

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

また、authentication、jobrunner、bomengine、およびscanサービスにこのテキストを追加する必要があります。

手順2a～cの代わりに、docker secretコマンドを使用して、HUB_Postgre_USER_PASSWORD_FILEという名前のシークレットとHUB_Postgre_ADMIN_PASSWORD_FILEという名前のシークレットを作成できます。

- a. docker secretコマンドを使用して、Docker Swarmにシークレットを通知します。シークレットの名前にスタック名を含める必要があります。次の例で、スタック名は「hub」です。

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

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

- b. docker-compose.local-overrides.ymlファイルで、webapp、jobrunner、authentication、bomengine、およびscanサービスにパスワードシークレットを追加します。webappサービスの例を次に示します。

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



```
- HUB_POSTGRES_ADMIN_PASSWORD_FILE
```

必要に応じて、コメント文字(#)を削除します。

コメント文字を削除し、必要に応じてスタック名をdocker-compose.local-overrides.ymlファイルの末尾のテキストに変更します。

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


3. 証明書認証を使用している場合は、docker-swarmディレクトリにあるdocker-compose.local-overrides.ymlファイルを編集して、webapp、jobrunner、authentication、およびscanサービスで、すべての証明書ファイル(HUB_Postgre_CA(サービスCAファイル)、HUB_Postgre_CERT(クライアント証明書ファイル)、HUB_Postgre_KEY(クライアントキーファイル))が含まれているディレクトリを/run/secretsにマウントします。手順2cの例を参照してください。
4. 証明書および/またはユーザー名/パスワード認証でSSLを使用できるようにするには、HUB_Postgre_ENABLE_SSL_CERTを「true」に設定し、手順2と3を完了します。
5. Black Duckをインストールまたはアップグレードします。

既存の外部データベースのPostgreSQLユーザー名の変更

デフォルトでは、PostgreSQLデータベースユーザーのユーザー名はblackduck_userで、PostgreSQL管理者のユーザー名はblackduckです。

外部PostgreSQLデータベースを使用している場合は、これらのユーザー名を変更できます。

これらの手順は、外部データベースが現在blackduckおよびblackduck_userユーザー名を使用している既存のBlack Duckインスタンス用の手順です。外部データベースの新しい構成のユーザー名を変更するには、前のセクションの手順に従います。

 **重要：** 管理者権限を持っていないBlack Duckデータベースユーザー(これはGCPやRDSなどのホストされたプロバイダで一般的)の場合は、bds_hubデータベースに接続して次を実行します GRANT blackduck_user TO blackduck;

既存のPostgreSQLアカウント名を変更するには、次の手順を実行します。

1. Black Duckを停止します。
2. bds_hubデータベースで、ユーザーの名前を変更し、パスワードをリセットします。


```
alter user blackduck_user rename to NewName1 ;
alter user blackduck rename to NewName2 ;
alter user NewName1 password 'NewName1Password' ;
alter user NewName2 password 'NewName2Password' ;
```
3. docker-swarmディレクトリにあるhub-postgres.envファイルで、HUB_Postgre_USERおよびHUB_Postgre_ADMINの値を編集します。HUB_Postgre_USERの値は、blackduck_userの新しいユーザー名です。HUB_Postgre_ADMINの値は、blackduckの新しいユーザー名です。以下に例を示します。


```
HUB_POSTGRES_USER=NewName1
HUB_POSTGRES_ADMIN=NewName2
```
4. Black Duckを再起動します。

 **注：** 2020.4.0リリースでは、BDIOデータベースが削除されました。

プロキシ設定の構成

blackduck-config.envファイルを編集して、プロキシ設定を構成します。外部インターネットアクセスにプロキシが必要な場合は、これらの設定を構成する必要があります。

Black Duck Softwareがホストするサービスへのアクセスが必要なコンテナは次のとおりです。

- ・ Authentication
- ・ Registration
- ・ Job runner
- ・ Web app
- ・ スキャン
- ・ Bomengine

プロキシ環境変数は次のとおりです。

- ・ HUB_PROXY_HOST。プロキシサーバーホストの名前。
- ・ HUB_PROXY_PORT。プロキシサーバーホストがリスンするポート。
- ・ HUB_PROXY_SCHEME。プロキシサーバーへの接続に使用するプロトコル。
- ・ HUB_PROXY_USER。プロキシサーバーにアクセスためのユーザー名。

NTLMプロキシの環境変数は次のとおりです。

- ・ HUB_PROXY_WORKSTATION。認証要求発信元のワークステーション。基本的には、このマシンのコンピュータ名。
- ・ HUB_PROXY_DOMAIN。内部で認証するドメイン。

プロキシパスワード

認証がプロキシ経由で使用される場合、次のサービスではプロキシパスワードが必要です。

- ・ Authentication
- ・ Bomengine
- ・ Web App
- ・ Registration
- ・ Job Runner
- ・ スキャン

プロキシパスワードを指定する方法として、次の3つの方法があります。

- ・ HUB_PROXY_PASSWORD_FILEというテキストファイルが含まれているディレクトリを/run/secretsにマウントします。これは最も安全なオプションです。
- ・ プロキシパスワードを含むHUB_PROXY_PASSWORDという名前の環境変数を指定します。

- 次の手順に従って、docker secretコマンドを使用して、HUB_PROXY_PASSWORD_FILEという名前のシークレットを作成します。

1. docker secretコマンドを使用して、Docker Swarmにシークレットを通知します。シークレットの名前をスタック名に含める必要があります。次の例で、スタック名は「hub」です。

```
docker secret create hub_HUB_PROXY_PASSWORD_FILE <パスワードを含むファイル>
```

2. docker-swarmディレクトリにあるdocker-compose.local-overrides.ymlファイルで、各サービス（authentication、webapp、registration、jobrunner、Match engine、Bom engine、およびscan）に対してシークレットへのアクセスを提供します。scanサービスの例を次に示します。

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

必要に応じて、コメント文字(#)を削除します。


3. docker-compose.local-overrides.ymlファイルの末尾にあるsecretsセクションに、次を追加します。

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

必要に応じて、コメント文字(#)を削除します。

環境変数が個別のマウント済みのファイルまたはシークレットに指定されていない場合は、blackduck-config.envファイルを使用して環境変数を指定できます。

1. HUB_PROXY_PASSWORDの前にあるシャープ記号(#)を削除して、コメントアウトを解除します。
2. プロキシパスワードを入力します。
3. ファイルを保存します。

 注：Docker Swarm導入でdocker secret HUB_PROXY_PASSWORD_FILEを使用してプロキシパスワードを提供したときにKB呼び出しが失敗した場合は、blackduck-config.envファイルでパスワードを提供して問題を解決します。

プロキシ証明書のインポート

プロキシ証明書をインポートして、プロキシを操作できます。

1. プロキシ証明書ファイルを使用して、<スタック名>_HUB_PROXY_CERT_FILEという名前のdocker secretを作成します。次に例を示します。

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

2. docker-swarmディレクトリにあるdocker-compose.local-overrides.ymlファイルで、次のサービスにシークレットへのアクセスを提供します：authentication、webapp、registration、jobrunner、およびscan。scanサービスの例を次に示します。


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

レポートデータベースのパスワードの設定

このセクションでは、レポートデータベースのパスワードを構成する手順について説明します。

docker-swarm/binディレクトリにあるhub_reportdb_changepassword.shスクリプトを使用して、レポートデータベースのパスワードを設定または変更します。

4. 管理タスク・job runner、scan、bomengine、およびbinaryscannerコンテナのスケーリング

 注：このスクリプトは、Black Duckによって自動的にインストールされるデータベースコンテナを使用している場合に、レポートデータベースのパスワードを設定または変更します。外部のPostgreSQLデータベースを使用している場合は、任意のPostgreSQL管理ツールを使用してパスワードを設定します。

スクリプトを実行してパスワードを設定または変更する場合は、次の点に注意してください。

- ・ dockerグループのユーザーまたはルートユーザーであるか、sudoアクセス権を持っている必要があります。
- ・ PostgreSQLデータベースコンテナを実行しているDockerホストで操作を実行する必要があります。

次の例では、レポートデータベースのパスワードは「blackduck」に設定されます。

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

job runner、scan、bomengine、およびbinaryscannerコンテナのスケーリング

job runner、scan、bomengine、およびbinaryscannerコンテナをスケーリングできます。

次のコマンドを実行するには、dockerグループのユーザーまたはルートユーザーであるか、sudoアクセス権を持っている必要があります。

bomengineコンテナのスケーリング

この例では、2つ目のbomengineコンテナが追加されます。

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

bomengineコンテナの現在の数よりも小さい数を指定することで、bomengineコンテナを削除できます。次の例では、bomengineコンテナが減らされて1つになります。

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

 注：bomengineコンテナをjobrunnerコンテナと同じレベルにスケーリングすることをお勧めします。

job runnerコンテナのスケーリング

この例では、2つ目のJob Runnerコンテナが追加されます。

```
docker service scale hub_jobrunner=2
```

job runnerコンテナの現在の数よりも小さい数を指定することで、job runnerコンテナを削除できます。次の例では、job runnerコンテナが減らされて1つになります。

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

scanコンテナのスケーリング

この例では、2つ目のScanコンテナが追加されます。

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

scanコンテナの現在の数よりも小さい数を指定することで、scanコンテナを削除できます。次の例では、scanコンテナが減らされて1つになります。


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

binaryscannerコンテナのスケーリング

binaryscannerコンテナはBlack Duck – Binary Analysisで使用されます。

この例では、2つ目のbinaryscannerコンテナが追加されます。

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


 注：Binaryscannerコンテナを1個追加するごとに、1 CPU、2 GB RAM、100 GBの空きディスク容量の追加が必要です。

binaryscannerコンテナの現在の数よりも小さい数を指定することで、binaryscannerコンテナを削除できます。次の例では、binaryscannerコンテナが減らされて1つになります。

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

シングルサインオンのSAMLの設定

Security Assertion Markup Language (SAML) は、XMLベースのオープンな標準データ形式で、パーティ間で認証および認証データを交換します。たとえば、アイデンティティプロバイダとサービスプロバイダ間での交換です。Black DuckのSAML実装ではシングルサインオン(SSO)機能が提供され、SAMLが有効になっているとき、Black DuckユーザーはBlack Duckに自動的にサインインできます。SAMLの有効化はBlack Duckユーザー全員に適用され、個別のユーザーに選択的に適用することはできません。

 注：ホストされるすべてのお客様は、SAMLまたはLDAPを介したシングルサインオン(SSO)の既成サポートを活用して、Black Duckアプリケーションへのアクセスのセキュリティを確保する必要があります。これらのセキュリティ機能を有効にして設定する方法については、インストールガイドを参照してください。さらに、2ファクタ認証を提供しているSAML SSOプロバイダを使用しているお客様は、そのテクノロジーを有効にして活用し、Black Duckアプリケーションへのアクセスのセキュリティをさらに高めることをお勧めします。

SAML機能の有効/無効を切り替えるには、システム管理者の役割を持つユーザーである必要があります。

追加のSAML情報：

- ・ Assertion Consumer Service (ACS) : <https://<host>/saml/SSO>
- ・ 推奨されるサービスプロバイダのエンティティID : <https://<host>>。hostはBlack Duckサーバーの場所です。

次の点に注意してください。

- ・ Black Duck は属性ステートメントで情報が提供されている場合、外部ユーザーの情報(氏名、名、姓、電子メール)を同期して取得できます。姓と名の値は大文字と小文字が区別されるため注意してください。
Black Duck は、Black Duckでグループ同期を有効にした場合、外部ユーザーのグループ情報を同期することもできます。
- ・ SAMLを有効にしてログインすると、Black Duckのログインページではなく、IDプロバイダのログインページにリダイレクトされます。
- ・ SSOユーザーがBlack Duckからログアウトしたときに、Black Duckから正常にログアウトしたことを通知するログアウトページが表示されるようになりました。このログアウトページには、Black Duckに再ログインするためのリンクが含まれ、ユーザーがBlack Duckに正常に再ログインするために資格情報を提供する必要がありません。
- ・ SSOシステムに問題があり、SSO設定を無効にする必要がある場合は、Black Duck `servername/sso/login`を入力して、Black Duckにログインします。

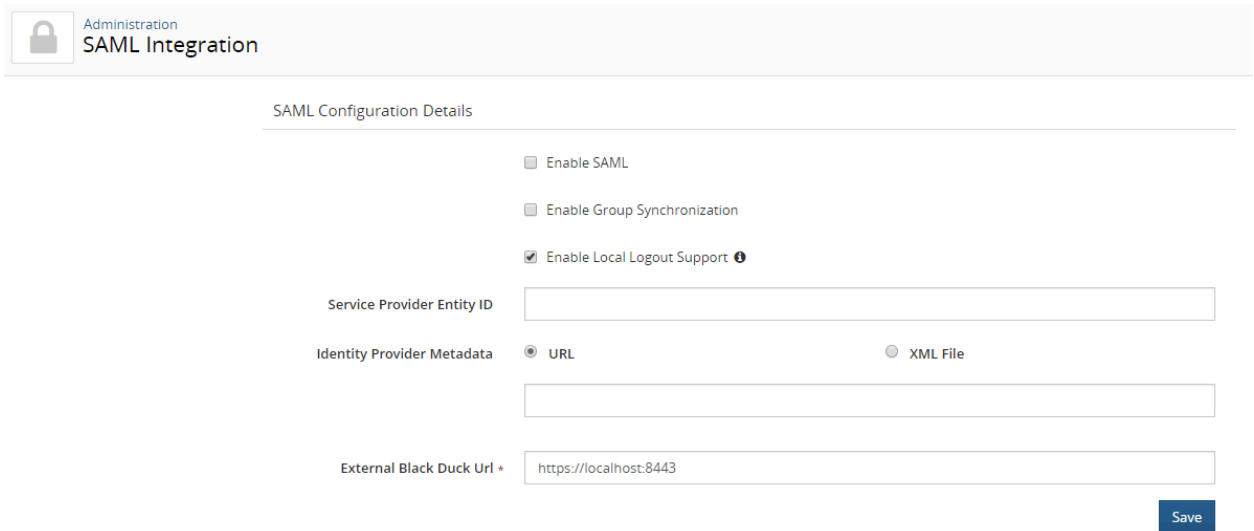
SAMLを使用するシングルサインオンを有効にするには

1.  をクリックします。


[管理]ページが表示されます。

4. 管理タスク・シングルサインオンのSAMLの設定

2. [SAML統合]を選択して[SAML統合]ページを開きます。



3. [SAML設定の詳細]設定で、次の操作を完了します。
 - a. [SAMLの有効化]チェックボックスをオンにします。
 - b. 必要に応じて、[グループ同期の有効化]チェックボックスをオンにします。このオプションが有効な場合、ログイン時にIDプロバイダ (IDP) のグループがBlack Duckで作成され、ユーザーはこれらのグループに割り当てられます。「Groups」という属性名を使用して属性ステートメントでグループを送信するように、IDPを設定する必要があります。
 - c. 必要に応じて、[ローカルログアウトのサポートの有効化]チェックボックスをオンにします。このオプションが有効な場合、Black Duckからログアウトすると、IDPのログインページが表示されます。


 注：ローカルログアウトのサポートが有効な場合、SAML要求はForceAuthn="true"で送信されます。IDPをチェックして、この機能がサポートされていることを確認します。
 - d. [サービスプロバイダのエンティティID]フィールド。ご使用の環境のBlack Duckサーバーの情報を、https://host形式で入力します。ここで、hostはBlack Duckサーバーです。
 - e. IDプロバイダのメタデータ。次のいずれかを選択します。
 - ・ [URL]を選択して、IDプロバイダのURLを入力します。
 - ・ [XMLファイル]を選択して、ファイルをドロップするか、表示されている領域をクリックして、XMLファイルを選択できるダイアログボックスを開きます。
 - f. [外部Black Duck URL]フィールド。Black DuckサーバーのパブリックURLです。

例: https://blackduck-docker01.dc1.lan

4. [保存]をクリックします。

[保存]をクリックすると、[BlackDuckメタデータURL]フィールドが表示されます。リンクをコピーするか、SAML XML構成情報を直接ダウンロードできます。

SAMLを使用するシングルサインオンを無効にするには

1.  をクリックします。
2. [SAML統合]を選択して[SAML統合]ページを開きます。

3. [SAML設定の詳細]設定で、[SAMLの有効化]チェックボックスをオフにします。
4. [保存]をクリックします。

ソースファイルのアップロード

構成表レビュー担当者は、マッチを確認し、偽陰性を調査することで、スキャンの結果を簡単に確認できる必要があります。スニペットマッチをレビューする場合、ソースファイルとマッチの並列比較を確認することはマッチの評価とレビューに役立ちます。

Black Duck では、ソースファイルをアップロードすることができます。これにより、構成表レビュー担当者はBlack Duck UI内からファイルの内容を確認することができます。

スキャン中にディープライセンスデータの検出または著作権テキストの検索を有効にした場合、ソースファイルをアップロードすると、構成表レビュー担当者は、検出されたライセンスまたは著作権テキストをBlack Duck UI内で確認することができます。ファイルがアップロードされると、Black Duckは埋め込みライセンスと著作権テキストのリストを提供し、ファイル内のテキストが強調表示で表示されます。

構成表レビュー担当者がBlack Duck UI内からファイルコンテンツを表示するには、次の手順を実行します。

1. 管理者は、ソースファイルをアップロードできるようにする必要があります。
 - a. 管理者は、環境変数を使用して機能を有効にし、シールキーと呼ばれる「ユーザーキー」で構成されるDocker関連のシークレットを提供します。
 - b. Black Duckはシールキーを取得し、ソースファイルの暗号化と復号化に使用される「マスターキー」を生成します。
 「マスターキー」は、「ユーザーキー」(AES-GCM-256、キー長32バイト)によって暗号化され、マウントされたボリュームに格納されます。
2. グローバルまたはプロジェクトコードスキャナ役割を持っているユーザーは、署名スキャナを使用して、`--upload-source`パラメータを有効にする必要があります。詳細については、オンラインヘルプまたはユーザーガイドを参照してください。

スキャンクライアントは、SSL/TLSで保護されたエンドポイントと適切な認証トークンを使用して、ソースファイルの内容をBlack Duckインスタンスに送信します。

「マスターキー」はファイルを暗号化します。アップロードされたファイルは、ファイル名ではなく、関連するスキャンIDとファイル署名を使用して保存されます。

Black Duck UIで、ソースファイルは、HTTPSを介してネットワーク経由で送信されます。

次の点に注意してください。

- ・ ファイルをアップロードするのに必要な十分なディスク容量があることを確認してください。
- ・ 一度にアップロードできるソースの最大合計サイズは4 GB (4,000 MB) です。この値は構成可能です。
- ・ アップロードされたファイルは180日後に削除されます。この値は構成可能です。
- ・ アップロードサービスが最大ディスク設定の95%に達すると、ファイルは削除されます。
 サービスは、ディスク容量が最大ディスク設定の90%になるまで、古いファイルから順に削除します。

ファイルのアップロードを有効にするには、次の手順を実行します。

1. ホストシステムでSEAL_KEYという名前のファイルを作成して、「ユーザーキー」を作成します。このファイルは安全な場所に保存する必要があります。32バイトまたは32文字の長さにする必要があります。以下に例を示します。

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

4. 管理タスク・ソースファイルのアップロード

2. docker-swarmディレクトリにあるblackduck-config.envファイルのENABLE_SOURCE_UPLOADS環境変数をtrueに設定して、この機能を有効にします。

```
ENABLE_SOURCE_UPLOADS=true
```

3. Docker secretコマンドでSEAL_KEYを使用してDocker Swarmにキーを通知します。シークレットの名前にスタック名を含める必要があります。次の例で、スタック名は「hub」です。

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

4. docker-swarmディレクトリにあるdocker-compose.local-overrides.ymlファイルのuploadcacheセクションからコメント文字(#)を削除して、アップロードキャッシュサービスがシークレットにアクセスできるようにします。

```
uploadcache:
```

```
secrets:
  - SEAL_KEY
```

5. docker-swarmディレクトリにあるdocker-compose.local-overrides.ymlファイルの次のテキストから末尾までのコメント文字(#)を削除します。

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

6. [コンテナを再起動します](#)。

これで、署名スキャナを使用してソースファイルをアップロードすることができます。

シールキーと未処理のマスターキーのバックアップ

シールキーと未処理のマスターキーをバックアップします。これらのキーがバックアップされていない場合、シールキーを紛失するとデータが失われる可能性があります。

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

シールキーの置き換え

前のセクションで説明したように、既存のマスターキーをバックアップしておけば、シールキーを紛失した場合にそれを置き換えることができます。

シールキーを置き換えるには、次のスクリプトを使用します。

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

追加の構成オプション

Black Duck は、次の追加の構成を提供しています。

- ・ データ保持。デフォルトでは、ファイルデータは180日間保持されます。
この設定を変更するには、DATA_RETENTION_IN_DAYS環境変数を使用します。
- ・ ソースの合計サイズ。デフォルトでは、ソースの最大合計サイズは4 GB (4,000 MB) です。
この設定を変更するには、MAX_TOTAL_SOURCE_SIZE_MB環境変数を使用します。

これらの設定を変更するには、docker-swarmディレクトリにあるdocker-compose.local-overrides.ymlファイルのupload cacheサービスに環境変数を追加し、新しい設定を定義します。以下に例を示します。

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

アップロードされるファイルのサイズを制限することもできます。デフォルトでは、最大サイズは5 MBです。この設定を変更するには、コマンドラインで次のように入力します。

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

ここで、#はMB単位の新しい値です。たとえば、最大サイズを6 MBに増やすには、次のように入力します。

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

起動または停止 Black Duck

これらのコマンドを使用して、Black Duckを起動またはシャットダウンします。

起動 Black Duck

上書きファイルを使用してデフォルトの構成設定を変更していない場合は、これらのコマンドを使用します。

- PostgreSQLデータベースコンテナを使用してBlack Duckを起動するには、次のコマンドを実行します。

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

- PostgreSQLデータベースコンテナを使用してBlack DuckをBlack Duck - Binary Analysisとともに起動するには、次のコマンドを実行します。

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

- 外部データベースを使用してBlack Duckを起動するには、次のコマンドを実行します。

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

- 外部データベースを使用してBlack DuckをBlack Duck - Binary Analysisとともに起動するには、次のコマンドを実行します。

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


- Swarmサービスに対してファイルシステムが読み取り専用の状態でBlack Duckを実行している場合は、前の手順にdocker-compose.readonly.ymlファイルを追加します。

たとえば、PostgreSQLデータベースコンテナを使用してBlack Duckをインストールするには、次のコマンドを入力します。

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

上書きファイル使用時のBlack Duckの起動

上書きファイルを使用してデフォルトの構成設定を変更した場合は、これらのコマンドを使用します。

 注： docker-compose.local-overrides.ymlファイルは、docker-composeコマンドで最後に使用される.ymlファイルである必要があります。

- PostgreSQLデータベースコンテナを使用してBlack Duckを起動するには、次のコマンドを実行します。

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

- PostgreSQLデータベースコンテナを使用してBlack DuckをBlack Duck - Binary Analysisとともに起動するには、次のコマンドを実行します。

```
docker swarm init
```


4. 管理タスク・ユーザーセッションタイムアウトの構成

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

- ・ 外部データベースを使用してBlack Duckを起動するには、次のコマンドを実行します。

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

- ・ 外部データベースを使用してBlack DuckをBlack Duck - Binary Analysisとともに起動するには、次のコマンドを実行します。

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

- ・ Swarmサービスに対してファイルシステムが読み取り専用の状態でBlack Duckを実行している場合は、前の手順にdocker-compose.readonly.ymlファイルを追加します。

たとえば、PostgreSQLデータベースコンテナを使用してBlack Duckをインストールするには、次のコマンドを入力します。

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

シャットダウン Black Duck

- ・ 次のコマンドを実行してBlack Duckをシャットダウンします。


```
docker stack rm hub
```

ユーザーセッションタイムアウトの構成

Black Duckサーバーからユーザーを自動的にログアウトし、企業のセキュリティポリシーに合わせるには、ユーザーセッションタイムアウト値を構成します。

1. 現在のタイムアウト値を表示するには、次のGETリクエストを行います。


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

 注：GETメソッドを使用するには、OAuthクライアントの読み取り権限を持っている必要があります。

2. 現在のタイムアウト値を変更するには、PUTリクエスト本文で次のPUTリクエストを行います。

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

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

 注：このタスクでPUTメソッドを使用するには、OAuthクライアントを更新する権限を持っている必要があります。システム管理者の役割には、必要な権限が含まれています。

PUTリクエスト本文に入力する値は、新しいタイムアウト値です。

30分(1,800秒)から24時間(86,400秒)までのタイムアウト値を使用できます。

次のメディアタイプを使用できます。

```
application/vnd.blackducksoftware.user-4+json  
application/json
```

Postmanの例を次に示します。

PUT		https://localhost/api/system-oauth-client
Params	Authorization ●	Headers (10) Body ● Pre-request Script Tests
▼ Headers (2)		
	KEY	VALUE
<input checked="" type="checkbox"/>	Accept	application/vnd.blackducksoftware.user-4+json
<input checked="" type="checkbox"/>	Content-Type	application/vnd.blackducksoftware.user-4+json

次の例では、タイムアウト値が7,200秒から8,000秒に変更されます。

PUT		https://<hub-server>/api/system-oauth-client...
Params	Authorization ●	Headers (12) Body ● Pre-request Script Tests Settings
● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL Text ▼		
1	{	
2	"accessTokenValiditySeconds": 8000	
3	}	
Body Cookies Headers (17) Test Results		
Pretty Raw Preview Visualize BETA JSON ▼		
1	{	
2	"clientId": "00000000-0000-4000-0000-000000000001",	
3	"scopes": [
4	"READ",	
5	"CLIENT_MANAGEMENT",	
6	"WRITE"	
7],	
8	"grantTypes": [
9	"IMPLICIT"	
10],	
11	"registeredRedirectUri": [],	
12	"accessTokenValiditySeconds": 7200,	
13	"refreshTokenValiditySeconds": 14400,	
14	"_meta": {	
15	"allow": [
16	"DELETE",	
17	"GET",	
18	"PUT"	
19],	
20	"href": "https://<hub-server>/api/oauthclients/00000000-0000-4000-0000-000000000001",	
21	"links": []	
22	}	
23	}	

Change 7200 to 8000 seconds by using PUT request with the new value.

カスタマサポートへのお客様のBlack Duckシステム情報の提供

カスタマサポートは、システム統計と環境またはネットワーク情報など、Black Duckのインストールに関する情報の提供を求める場合があります。この情報をすばやく簡単に取得できるようにするために、Black Duckでは、この情報を収集するためのsystem_check.shスクリプトを提供しています。スクリプトは、この情報を作業ディレクトリにあるsystem_check.txtファイルに出力します。その後、このファイルをカスタマサポートに送信できます。


system_check.shスクリプトはdocker-swarm/binディレクトリにあります。

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

このスクリプトを実行するには、dockerグループのユーザーまたはルートユーザーであるか、sudoアクセス権を持っている必要があります。

デフォルトのsysadminユーザーについて

Black Duckをインストールすると、デフォルトのシステム管理者(sysadmin)アカウントが構成されます。デフォルトのsysadminユーザーには、すべての役割と権限が関連付けられています。

 ヒント：ベストプラクティスとして、最初のログインにはデフォルトのsysadminアカウントを使用し、すぐにデフォルトのパスワード(blackduck)を変更して、サーバーのセキュリティを確保する必要があります。パスワードを変更するには、Black DuckのUIの右上隅にあるユーザー名/ユーザープロフィールアイコンから[マイプロフィール]を選択します。

sysadminユーザーに関連付けられているデフォルトの電子メールアドレスを編集するには、Black DuckのUIの[ユーザーの管理]ページに移動し、sysadminユーザー名を選択し、電子メールアドレスを変更して保存します。変更を確認するには、ログアウトしてから再度ログインする必要があります。

[ユーザーの管理]ページにアクセスするには、使用するユーザーアカウントにスーパーユーザーの役割が必要です。この役割は、デフォルトでsysadminアカウントに割り当てられています。電子メールアドレスの主な目的は、ユーザーアカウントの連絡先参照情報として使用することです。

Black Duckレポート遅延の構成

Black Duck 2022.10.0で、レポートデータベースジョブプロセスは480分ごとに実行されます。これは構成可能です。

別のレポート遅延を設定するには、次の手順を実行します。

1. docker-swarmディレクトリのblackduck-config.envファイルを編集し、を構成します。
BLACKDUCK_REPORTING_DELAY_MINUTES=<value in minutes>

例：BLACKDUCK_REPORTING_DELAY_MINUTES=360

2. コンテナを再起動します。

コンテナのタイムゾーンの構成

デフォルトでは、Black DuckコンテナのタイムゾーンはUTCです。監視目的で、ログに表示されるタイムスタンプにローカルタイムゾーンが反映されるようにこの値を変更することができます。

別のタイムゾーンを設定するには、次の手順を実行します。

1. `docker-swarm`ディレクトリにある`blackduck-config.env`ファイルのTZ環境変数の値を新しいタイムゾーンに設定します。[ここ](#)に示すように、Wikipediaに示されている値を使用します。

たとえば、コロラド州デンバーで使用されているタイムゾーンに変更するには、次のように入力します。

```
TZ=America/Denver
```

2. コンテナを再起動します。

デフォルトの使用法の変更

使用法は、このバージョンがリリースされるときにコンポーネントをプロジェクトにどのように含めるかについて示します。

使用法の値は次のとおりです。

- ・ 静的にリンク。静的にリンクされ、プロジェクトとともに配布される、密接に統合されているコンポーネント。
- ・ 動的にリンク。DLLやjarファイルなど、動的にリンクされ、中程度に統合されているコンポーネント。
- ・ ソースコード。javaまたは.cppファイルなどのソースコード。
- ・ 開発ツール/除外。コンポーネントは、リリースされたプロジェクトに含まれません。たとえば、構築、開発、またはテストのため社内的に使用されるコンポーネントです。例として、ユニットテスト、IDEファイル、コンパイラがあります。
- ・ 分離した製品。統合の弱いコンポーネントを対象としています。作業内容はコンポーネントから取得されていません。分離した製品と見なされるようにするには、アプリケーションに独自の実行可能ファイルが含まれていて、コンポーネントとアプリケーションがリンクされていない必要があります。例としては、配布メディアに無償のAcrobat PDFビューアを含めている場合があります。
- ・ 標準の実装。標準に従って実装している場合を対象としています。たとえば、プロジェクトとともに出荷されるJava Spec Requestなどがあります。
- ・ 単純集合。プロジェクトで使用されていない、または依存していないコンポーネントを対象としています。ただし、これらは同じメディア上にある場合があります。たとえば、関連付けられていない製品のサンプル版を配布に含めている場合があります。
- ・ 前提条件。必要ですが配布で提供されないコンポーネントを対象としています。
- ・ 未指定。このコンポーネントの使用法はまだ明確にされていません。

Black Duckバージョン2020.10.0で、SynopsysはUNSPECIFIED使用法値を導入しました。既存のデフォルトの代わりに、これをデフォルトオプションとして使用することで、コンポーネントに正しい使用法値またはデフォルトの使用法値のどちらかが割り当てられているかどうかの混乱を解消することができます。たとえば、デフォルトの使用法値としてDYNAMICALLY_LINKEDを使用した場合、DYNAMICALLY_LINKEDの正しい使用法値を持つコンポーネントと、デフォルトで使用方法値が割り当てられたコンポーネントを区別できない場合があります。使用法のデフォルトとしてUNSPECIFIEDを使用することで、使用方法値としてUNSPECIFIEDが割り当てられていることが表示された場合に、有効な使用方法値を割り当てるためのアクションを実行する必要があることがわかります。

デフォルトの使用法は、マッチタイプによって決まります。スニペットの使用法はソースコードですが、その他のすべてのマッチタイプの使用法は動的にリンクです。

Black Duck には次の変数があり、これらを使用して類似するマッチタイプのデフォルトの使用法を変更できます。

4. 管理タスク・デフォルトの使用法の変更

- ・ BLACKDUCK_HUB_FILE_USAGE_DEFAULT。この変数の使用法を定義すると、次のマッチタイプのデフォルト値が設定されます。
 - ・ バイナリ
 - ・ 完全ディレクトリ
 - ・ 完全ファイル
 - ・ 追加/削除されたファイル
 - ・ 変更したファイル
 - ・ 部分
- ・ BLACKDUCK_HUB_DEPENDENCY_USAGE_DEFAULT。この変数の使用法を定義すると、次のマッチタイプのデフォルト値が設定されます。
 - ・ ファイルの依存関係
 - ・ 直接的な依存関係
 - ・ 推移的な依存関係
- ・ BLACKDUCK_HUB_SOURCE_USAGE_DEFAULT。この変数の使用法を定義すると、次のマッチタイプのデフォルト値が設定されます。
 - ・ スニペット
- ・ BLACKDUCK_HUB_MANUAL_USAGE_DEFAULT。この変数の使用法を定義すると、次のマッチタイプのデフォルト値が設定されます。
 - ・ 手動で追加
 - ・ 手動で判定
- ・ BLACKDUCK_HUB_SHOW_UNMATCHED: マッチしなかったコンポーネント数を表示するかどうかを決定します。デフォルトはfalseです(表示されません)。

アップロードされたjsonld/bdioファイルbdio2のマッチタイプ

これらのマッチのデフォルトの使用法は、動的にリンクです。

- ・ 直接的な依存関係バイナリ
- ・ 推移的な依存関係バイナリ


これらは、既存のバイナリマッチと同じフルファイルマッチです。

1つのファイルに複数の推移的な依存関係バイナリマッチを含めることができますが、直接的な依存関係バイナリは1つのみです。

別の使用法値を構成するには、次の手順を実行します。

1. docker-swarmディレクトリにあるblackduck-config.envファイルで、コメントアイコン(#)を削除し、値を入力して、新しい使用法値にします。ファイルに示されている使用法値のいずれかを使用します: SOURCE_CODE、STATICALLY_LINKED、DYNAMICALLY_LINKED、SEPARATE_WORK、IMPLEMENTATION_OF_STANDARD、
たとえば、ファイルのデフォルトの使用法を未指定に変更するには、次のようにします。

```
BLACKDUCK_HUB_FILE_USAGE_DEFAULT=UNSPECIFIED
```

 注: 正しくない使用法テキストを入力した場合は、元のデフォルト値が適用されます。jobrunnerコンテナのログファイルに警告メッセージが表示されます。

変更された使用法値は、新しいスキャンまたは再スキャンに適用されます。

Black DuckコンテナのユーザーIDのカスタマイズ

コンテナを実行するユーザーID (UID) の変更が必要になる場合があります。

各コンテナの現在のUIDは次のとおりです。

- ・ Authentication (blackduck-authentication) : 100
- ・ Binaryscanner (bdba-worker) : 0
- ・ CA (blackduck-cfssl) : 100
- ・ DB (blackduck-postgres) : 70
- ・ Documentation (blackduck-documentation) : 8080
- ・ Job Runner (blackduck-jobrunner) : 100
- ・ Logstash (blackduck-logstash) : 100
- ・ RabbitMQ (rabbitmq) : 100
- ・ Registration (blackduck-registration) : 8080
- ・ Scan (blackduck-scan) : 8080
- ・ Web App (blackduck-webapp) : 8080
- ・ Webserver (blackduck-nginx) : 100
- ・ Uploadcache (blackduck-uploadcache) : 100
- ・ Redis (blackduck-redis) : 任意のユーザー/グループとして実行
- ・ Bomengine (blackduck-bomengine) : 100
- ・ Matchengine (blackduck-matchengine) : 100

UIDを変更するには、docker-swarmディレクトリにあるdocker-compose.local-overrides.ymlにコンテナの新しい値を追加します。コンテナのセクションに

user:UID_NewValue:root 行を追加します。

次の例では、webappコンテナのUIDを1001に変更します。

webapp:

user: 1001:root

次の点に注意してください。

- ・ postgresコンテナとbinaryscannerコンテナのUIDは変更できません。
 - ・ postgresコンテナのUIDは70でなければなりません。
 - ・ binaryscannerコンテナのUIDは0 (ルート) でなければなりません。
- ・ 一部のコンテナのUID値は同じですが (たとえば、Documentation、Registration、Scan、Web AppコンテナのUIDは8080)、1つのコンテナのUID値を変更しても、同じUID値のコンテナのUID値は変更されません。たとえば、Web Appコンテナの値を8080から1001に変更しても、Documentation、Scan、またはRegistrationコンテナの値は変更されません。これらのコンテナのUID値は8080のままになります。
- ・ コンテナは、どのユーザーとしてコンテナが実行されるかを問わず、ユーザーはルートグループに属するものとして指定される必要があると想定します。

UIDをカスタマイズするには、次の手順を実行します。

4. 管理タスク・Webサーバー設定の構成

1. Black Duckを**停止**します。
2. 上記のように値を編集します。
3. Black Duckを**起動**します。

Webサーバー設定の構成

hub-webserver.envファイルを次のように編集します。

- ・ ホスト名を構成します。
- ・ ホストポートを構成します。
- ・ IPv6を無効にします。

ホスト名の構成

hub-webserver.envファイルを編集して、証明書ホスト名が一致するようにホスト名を設定します。環境変数には、デフォルト値としてサービス名が設定されています。

証明書が構成されていない場合、Webサーバーの起動時にHTTPS証明書が生成されます。ホスト名が一致するように、PUBLIC_HUB_WEBSERVER_HOST環境変数の値を指定して、リッスンするホスト名をWebサーバーに通知する必要があります。そうしないと、証明書にはホスト名として使用するサービス名のみが含まれるようになります。この値は、ユーザーがBlack Duckにアクセスするためにブラウザに入力する公開ホスト名に変更する必要があります。以下に例を示します。

```
PUBLIC_HUB_WEBSERVER_HOST=blackduck-docker01.dcl.lan
```

ホストポートの構成

ホストポートには、異なる値を設定できます。デフォルトは443です。

ホストポートを設定するには、次の手順を実行します。

1. docker-swarmディレクトリにあるdocker-compose.local-overrides.ymlファイルに、新しいホストポート値を追加します。

webserverセクションを使用して、次の形式でポート情報を追加します。ports: ['NewValue:8443'] たとえば、ポートを8443に変更するには、次のように指定します。

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

2. hub-webserver.envファイルのPUBLIC_HUB_WEBSERVER_PORT値を編集して、新しいポート値を追加します。以下に例を示します。

```
PUBLIC_HUB_WEBSERVER_PORT=8443
```

3. docker-compose.ymlファイルの次の部分をコメントアウトして、ホストのポート443へのアクセスを無効にします。そうしないと、ユーザーは引き続きポート443を使用してホストにアクセスできます。

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

IPv6の無効化

デフォルトでは、NGINXはIPv4とIPv6をリッスンします。ホストマシンでIPv6が無効になっている場合は、IPV4_ONLY環境変数の値を1に変更します。

UTF8文字エンコードを使用した構成表レポートの作成

アルファベット以外の文字を使用している場合に構成表レポートでUTF8文字エンコードのサポートを有効にするには、`blackduck-config.env`ファイルに以下を追加します。

```
USE_CSV_BOM=true
```

スキャン監視

スキャン監視エンドポイントの情報が収集されるデフォルトの時間は1時間です。この時間は、次のプロパティを使用して別の値に設定できます。

`HUB_SCAN_MONITOR_ROLLUP_WINDOW_SECONDS`

上記で設定した最後の時間（デフォルトの期間は1時間）に、システムが分析するスキャンが一定数以上ある場合のみAPIは情報を提供できます。スキャンがしきい値よりも小さい場合、レベル1リクエストはデフォルトでOK応答を返します。スキャン数は、次のプロパティを使用して別の値に設定できます。

`HUB_SCAN_MONITOR_MINIMUM_SCAN_COUNT`

レベル1リクエストは、失敗率のしきい値に基づいて「OK」または「NOT OK」の応答を返します（上限を超えると、「NOT OK」応答が返されます）。上限（デフォルトは30%）と下限（デフォルトは10%）はパーセンテージで設定され、次のプロパティで設定できます。

`HUB_SCAN_MONITOR_ERROR_RATE_LOWER_THRESHOLD_PERCENT`

`HUB_SCAN_MONITOR_ERROR_RATE_HIGHER_THRESHOLD_PERCENT`

スキャン監視APIリクエストの詳細については、Black Duckで入手可能な『REST API開発者ガイド』を参照してください。

HTMLレポートのダウンロードサイズの設定

任意のサイズのHTMLレポートをダウンロードできるように環境を設定できます。デフォルトのサイズは3000 KBです。この制限を超えるレポートは、エラーメッセージ「503サービスは使用できません」が返されます。

この制限を変更するには、`blackduck-config.env`ファイルの`HUB_MAX_HTML_REPORT_SIZE_KB`プロパティの値を変更します。

5. アンインストール: Black Duck

Black Duckをアンインストールするには、次の手順に従います。

次のいずれかの方法を使用して、Black Duckをアンインストールします。

- ・ コンテナを停止して削除し、ボリュームを削除する。

```
docker stack rm hub
```

- ・ コンテナを停止して削除し、ボリュームは保持する。以下に例を示します。

```
docker volume prune
```

⚠ 注意: このコマンドは、未使用のボリュームをすべて削除します。どのコンテナからも参照されていないボリュームが削除されます。これには、他のアプリケーションで使用されていない未使用のボリュームが含まれます。

PostgreSQLデータベースはバックアップされません。[データベースをバックアップする](#)には、これらの手順を使用します。

6. のアップグレード Black Duck

Black Duck は、利用可能な任意のバージョンへのアップグレードをサポートしているため、1回のアップグレードで複数のバージョンを飛び越えることができます。

❗ 重要： 4.0.0より前のバージョンのBlack Duckを使用している場合は、2020.6以降にアップグレードする前に2020.4にアップグレードする必要があります。

アップグレード手順は、以前のバージョンのBlack Duckによって異なります。

- ・ AppMgrアーキテクチャ
- ・ シングルコンテナAppMgrアーキテクチャ
- ・ マルチコンテナDocker

📖 注： 2019.8.0より前のバージョンからアップグレードする場合、VulnerabilityReprioritizationJobとVulnerabilitySummaryFetchJobの2つのジョブが起動時に実行され、脆弱性データが同期されます。

これらのジョブの実行には時間がかかる場合があります、既存の構成表の全体的な脆弱性スコアは、これらのジョブが完了するまで使用できません。システム管理者の役割を持っているユーザーは、Black Duckの[ジョブ]ページを使用してこれらのジョブを監視できます。

📖 注： 2018.12.0より前のバージョンからアップグレードする場合は、このリリースの新機能をサポートするためにデータ移行が必要になるため、通常よりもアップグレードに時間がかかります。アップグレード時間は、Black Duckデータベースのサイズによって異なります。アップグレードプロセスを監視したい場合は、Synopsisカスタマサポートにお問い合わせください。

インストールファイル

インストールファイルはGitHubで入手できます。

オーケストレーションファイルをダウンロードします。インストール/アップグレードプロセスの一環として、これらのオーケストレーションファイルは必要なDockerイメージをプルダウンします。

tar.gzファイルのファイル名はアクセス方法によって異なりますが、内容は同じです。

GitHubページからダウンロードする場合

1. リンクを選択して、GitHubページからtar.gzファイルをダウンロードします：<https://github.com/blackducksoftware/hub>。
2. Black Duck .gzファイルを解凍します。
`gunzip hub-2022.10.0.tar.gz`
3. Black Duck.tarファイルを展開します。
`tar xvf hub-2022.10.0.tar`

wgetコマンドを使用してダウンロードする場合

1. 次のコマンドを実行します。
`wget https://github.com/blackducksoftware/hub/archive/v2022.10.0.tar.gz`

6. のアップグレード Black Duck・監査イベントテーブルの未使用の行をパージする移行スクリプト

2. Black Duck .gzファイルを解凍します。

```
gunzip v2022.10.0.tar.gz
```

3. Black Duck.tarファイルを展開します。

```
tar xvf v2022.10.0.tar
```

監査イベントテーブルの未使用の行をパージする移行スクリプト

アップグレード中に、移行スクリプトが実行されて、レポートデータベースの変更によりaudit_eventテーブルで使用されなくなった行がパージされます。audit_eventテーブルのサイズによっては、このスクリプトの実行に時間がかかる場合があります。たとえば、350 GBのaudit_eventテーブルの場合、移行スクリプトの実行には約20分かかります。

- ❗ 重要：2019.12.0より前のバージョンのBlack Duckから2019.12.0以降のバージョンにアップグレードする場合は、レポートデータベースの変更により、1回のアップグレードでのみ移行スクリプトを実行する必要があります。


監査イベントテーブルのサイズを判断するには、次のいずれかのタスクを実行します。

- bds_hubデータベースで、次のコマンドを実行します。

```
SELECT pg_size_pretty( pg_total_relation_size('st.audit_event') );
```

- Black Duck UIにシステム管理者としてログインし、次の手順を実行します。

1.

展開メニューアイコン()をクリックし、[管理]を選択します。

2. [管理]ページで[システム情報]を選択します。

[システム情報]ページが表示されます。

3. ページの左側の列で[db]を選択します。

4. テーブルサイズテーブルで、audit_eventテーブル名のtotal_tbl_size値を見つけます。

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

移行スクリプトの実行が終了したら、audit_eventテーブルでVACUUMコマンドを実行してPostgreSQLのパフォーマンスを最適化することを強くお勧めします。

- システムの使用状況によっては、VACUUMコマンドを実行することで、Black Duckによって使用されなくなった大量のディスク容量を再利用できる可能性があります。
- このコマンドを実行すると、クエリのパフォーマンスが向上します。

📖 注：VACUUMコマンドを実行しないと、パフォーマンスが低下する可能性があります。

- ❗ 重要：VACUUMコマンドを実行するのに十分な容量があることを確認する必要があります。そうしないと、ディスク容量が不足して失敗し、データベース全体が破損する可能性があります。VACUUMコマンドは、audit_eventテーブルで現在使用されているディスク容量の2倍の容量を必要とします。

コンテナ化されたPostgreSQLデータベースの導入でVACUUMコマンドを実行するには、次の手順を実行します。

1. audit_eventテーブルのサイズを取得し、VACUUMコマンドを実行するのに十分な容量があることを確認します。
2. docker psコマンドを実行して、PostgreSQLコンテナのIDを取得します。
3. 次のコマンドを実行して、PostgreSQLコンテナにアクセスします。

```
docker exec -it <container_ID> psql bds_hub
```


4. 次のVACUUMコマンドを実行して、使用されなくなった容量を再利用します。

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

外部PostgreSQLデータベースを導入している場合は、audit_eventテーブルのサイズを判断し、VACUUMコマンドを実行し、完了したら導入を再起動する必要があります。

AppMgrアーキテクチャからのアップグレード

このセクションでは、AppMgrアーキテクチャをベースにしている以前のバージョンのBlack Duckから、マルチコンテナDockerアーキテクチャにアップグレードする方法について説明します。

 注：これらの手順は、AppMgr Amazon Web Services (AWS) AMIからアップグレードする場合にも該当します。

マルチコンテナDockerアーキテクチャへのアップグレードは、次の手順で構成されています。

1. PostgreSQLデータベースの移行。
既存のデータベースのデータを保持する場合、これはオプションの手順です。
2. Black Duckのアップグレード。


PostgreSQLデータベースの移行

既存のPostgreSQLデータを使用するには、データベースのデータを移行する必要があります。これは、次の手順で構成されています。

1. 元のPostgreSQLデータベースのバックアップ。
2. データの復元。


元のPostgreSQLデータベースをバックアップするには、次の手順を実行します。

1. Black Duckサーバーにblackduckユーザーとしてログインします。

 注：このユーザーは、Black Duckデータベースとインストールディレクトリを所有するユーザーです。


2. 次のコマンドを実行して、圧縮ファイルにダンプします。

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

 ヒント：十分な空き容量がある場所にデータベースをダンプしてください。この例では/tmpを使用しています。

このコマンドは、bds_hubデータベースからの情報を/tmpディレクトリのbds_hub.dump という名前のファイルに格納します。バックアップが不要ないくつかのスクラッチテーブルは無視されます。

3. bds_hub.dumpファイルを別のシステムに保存するか、オフラインで保存します。

 ヒント：データベースのダンプに時間がかかる場合は、非圧縮ファイルにダンプすることで速度を大幅に向上できます。ダンプの速度は3倍まで向上しますが、ファイルが4倍大きくなる可能性があります。使用しているシステムでこれを実行するには、pg_dumpコマンドに--compress=0パラメータを追加します。

PostgreSQLデータを復元するには、次の手順を実行します。

6. のアップグレード Black Duck・AppMgrアーキテクチャからのアップグレード

1. docker-swarmディレクトリにあるdocker-compose.dbmigrate.ymlファイルを使用します。データベースの移行に必要なコンテナとボリュームが起動します。

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

Dockerの一部のバージョンでは、イメージがプライベートリポジトリに存在する場合に、上記のコマンドに次のフラグを追加しない限り、dockerスタックはそれらをプルしません。

```
--with-registry-auth
```

2. DBコンテナが起動したら、docker-swarmディレクトリにある移行スクリプトを実行します。このスクリプトは、既存のデータベースダンプファイルからデータを復元します。

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

特定のデータベースを移行するには、次の構文を使用します。

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

<db name>に使用できる値は次のとおりです。

- bds_hub
- bds_hub_report

これで、Black DuckのマルチイメージDockerバージョンにアップグレードできます。

エラーメッセージ

Black DuckのAppMgrインストールからダンプファイルを復元すると、他のエラーメッセージとともに

```
"ERROR: role "blackduck" does not exist"
```

などのエラーメッセージが表示されることがあります。また、移行の最後に、次のような警告が表示される場合があります。

```
WARNING: errors ignored on restore: 7
```

これらのエラーメッセージと警告は無視できます。これらはデータの復元には影響しません。

のアップグレード Black Duck

1. Black Duckの以前のAppMgrバージョンでsetup-autostart.shスクリプトを実行した場合は、そのスクリプトによって作成された「iptables」エントリを削除する必要があります。ルートユーザーとして、Black Duckをインストールしたディレクトリ(/opt/blackduck/hub/appmgr/binなど)に移動し、deleteパラメータを指定してiptables-redirect.shスクリプトを実行します。

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

自動起動が構成されているかどうか不明な場合も、このスクリプトを安全に実行できます。これは、Black Duckの以前のAppMgrバージョンが自動起動用に構成されていない場合、このスクリプトは変更を行わないためです。

2. Black DuckのAppMgrバージョンがインストールされているのと同じサーバーにBlack Duckをインストールする場合は、次の手順を実行します。

- a. uninstall.shスクリプトを実行して、古いファイルを削除します。

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

- b. ルートユーザーまたはsudoアクセス権を持っているユーザーとして、autostartファイルを削除します。uninstall.shスクリプトは、スクリプトの実行の最後にファイルの場所を示します。以下に例を示します。

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

3. 新しいバージョンのBlack Duckのdocker-swarmディレクトリにあるファイルを使用して、次のいずれかのコマンドを実行します。

- ・ `docker stack deploy -c docker-compose.yml hub` (DBコンテナを使用している場合)
- ・ `docker stack deploy -c docker-compose.yml -c docker-compose.bdba.yml hub` (DBコンテナとBlack Duck - Binary Analysisを使用している場合)
- ・ `docker stack deploy -c docker-compose.externaldb.yml hub` (外部PostgreSQLデータベースを使用している場合)
- ・ `docker stack deploy -c docker-compose.externaldb.yml -c docker-compose.bdba.yml hub` (外部データベースとBlack Duck - Binary Analysisを使用している場合)

Swarmサービスに対してファイルシステムが読み取り専用の状態でBlack Duckをインストールする場合は、次のdocker-compose.readonly.ymlファイルを手順3に記載されている手順に追加します。

たとえば、PostgreSQLデータベースコンテナを使用してBlack Duckをインストールするには、次のコマンドを入力します。

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

シングルコンテナAppMgrアーキテクチャからのアップグレード

このセクションでは、シングルコンテナAppMgrアーキテクチャをベースにしている以前のバージョンのBlack Duckから、マルチコンテナDockerアーキテクチャにアップグレードする方法について説明します。

マルチコンテナDockerアーキテクチャへのアップグレードは、次の手順で構成されています。

1. PostgreSQLデータベースの移行。
既存のデータベースのデータを保持する場合、これはオプションの手順です。
2. Black Duckのアップグレード。

PostgreSQLデータベースの移行

既存のPostgreSQLデータを使用するには、データベースのデータを移行する必要があります。これは、次の手順で構成されています。

1. 元のPostgreSQLデータベースのバックアップ。
2. データの復元。

PostgreSQLデータベースをバックアップするには、次の手順を実行します。

1. 次のコマンドを実行して、PostgreSQLダンプファイルを作成します。

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

2. 次のコマンドを実行して、コンテナからダンプファイルをコピーします。

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

PostgreSQLデータを復元するには、次の手順を実行します。

6. のアップグレード Black Duck・シングルコンテナAppMgrアーキテクチャからのアップグレード

1. docker-swarmディレクトリにあるdocker-compose.dbmigrate.ymlファイルを使用します。データベースの移行に必要なコンテナとボリュームが起動します。

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

Dockerの一部のバージョンでは、イメージがプライベートリポジトリに存在する場合に、上記のコマンドに次のフラグを追加しない限り、dockerスタックはそれらをプルしません。

```
--with-registry-auth
```

2. DBコンテナが起動したら、docker-swarmディレクトリにある移行スクリプトを実行します。このスクリプトは、既存のデータベースダンプファイルからデータを復元します。

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

特定のデータベースを移行するには、次の構文を使用します。

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

<db name>に使用できる値は次のとおりです。

- bds_hub
- bds_hub_report

これで、Black DuckのマルチイメージDockerバージョンにアップグレードできます。

エラーメッセージ

Black DuckのAppMgrインストールからダンプファイルを復元すると、他のエラーメッセージとともに

「エラー: 役割「blkcdck」が存在しません」

などのエラーメッセージが表示されることがあります。また、移行の最後に、次のような警告が表示される場合があります。

警告: 復元時に無視したエラーの数: 7

これらのエラーメッセージと警告は無視できます。これらはデータの復元には影響しません。

Black Duckのアップグレード Black Duck

1. 新しいバージョンのBlack Duckのdocker-swarmディレクトリにあるファイルを使用して、次のいずれかのコマンドを実行します。

- docker stack deploy -c docker-compose.yml hub (DBコンテナを使用している場合)
- docker stack deploy -c docker-compose.yml -c docker-compose.bdba.yml hub (DBコンテナとBlack Duck - Binary Analysisを使用している場合)
- docker stack deploy -c docker-compose.externaldb.yml hub (外部PostgreSQLデータベースを使用している場合)
- docker stack deploy -c docker-compose.externaldb.yml -c docker-compose.bdba.yml hub (次で外部データベースを使用している場合: Black Duck - Binary Analysis)

Swarmサービスに対してファイルシステムが読み取り専用の状態でBlack Duckをインストールする場合は、次のdocker-compose.readonly.ymlファイルを上記の手順に追加します。

たとえば、PostgreSQLデータベースコンテナを使用してBlack Duckをインストールするには、次のコマンドを入力します。

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


既存のDockerアーキテクチャからのアップグレード

以前のバージョンのBlack Duckからアップグレードするには、次の手順を実行します。


1. PostgreSQLデータベースを移行します。

PostgreSQL 11.12は外部データベースに推奨され、サポートされていますが、PostgreSQL 9.6外部データベースのサポートはEOL(サポート終了)となりました。

内部PostgreSQLコンテナのユーザーの場合は、PostgreSQL 9.6が2020.8.xでサポートされるバージョンです。PostgreSQL 11.7は、内部PostgreSQLコンテナではまだサポートされていません。


 **重要：** Black Duck 2020.8.xへの移行では、外部データベース用にPostgreSQL 11.7に移行する必要はありませんが、PostgreSQL 11.7が推奨されます。PostgreSQL 9.6のフルサポートは維持されます。内部PostgreSQLコンテナのユーザーの場合は、PostgreSQL 9.6が引き続き2020.8.xでサポートされるバージョンです。

データ移行では、データベースダンプと新しいデータベースボリュームを保持するために、元のデータベースボリュームサイズの約2.5倍の追加の空きディスク容量が一時的に必要になります。経験則として、データベースが存在するボリュームの空き容量が60%以上あれば、ディスク容量は十分です。

 **注：** PostgreSQLデータベースのバージョンは、4.2.0でバージョン9.6.xにアップグレードされました。4.2.0より前のバージョンからアップグレードする場合は、Black Duckをアップグレードする前にデータベースを移行する必要があります。

バージョン4.2.0以降からアップグレードする場合、データベースの移行は任意です。


2. Black Duckのアップグレード。


 **注：** 4.1.0では、NGiNXのカスタムSSL証明書を構成する方法が変更されました。バージョン4.0.0または4.0.1からアップグレードし、NGiNXのカスタムSSL証明書を構成した場合は、[それらを再構成](#)する必要があります。

PostgreSQLデータベースの移行

既存のPostgreSQLデータを使用するには、データベースのデータを移行する必要があります。これは、次の手順で構成されています。

1. 元のPostgreSQLデータベースのバックアップ: Black Duckデータベース(bds_hub)、レポートデータベース(bds_hub_report)。
2. Black Duckコンテナの停止。
3. データの復元。

 **注：** Black Duckインスタンスが外部データベース(Amazon RDSなど)を使用するように構成されている場合、データをPostgreSQL 9.6からPostgreSQLの11.7インスタンスに移行し、そのインスタンスを指すようにシステムを構成できます。アップグレード前のシステムはPostgreSQL 9.6でのみ機能し、アップグレード後のシステムはPostgreSQL 9.6または11.7で機能します。

 **注：** Black Duck 2019.10.0リリースの時点で、bds_hub_reportデータベースは使用されなくなりました。VulnDBユーザーまたは以前のVulnDBユーザーは、bds_hub_reportデータベースをバックアップすることをお勧めします。他のすべてのユーザーの場合、bds_hub_reportデータベースに存在していたデータは、bds_hub databaseに存在するようになりました。

PostgreSQLデータベースをバックアップするには、次の手順を実行します。

6. のアップグレード Black Duck・既存のDockerアーキテクチャからのアップグレード

1. `hub_create_data_dump.sh`スクリプトを実行します。これにより、`blackduck-postgres`コンテナにPostgreSQLデータファイルが作成され、コンテナからローカルディレクトリにファイルがコピーされます。

❗ 重要: `docker-swarm/bin`ディレクトリにあるBlack Duckバージョン<version you're backing up/dumping from>の`hub_create_data_dump.sh`スクリプトを実行する必要があります。たとえば、2019.10.0 Docker-swarmデータベースをダンプする場合は、2019.10.0 `docker-swarm/bin`ディレクトリにあるダンプスクリプトを使用します。

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

このスクリプトは、複数のデータバックアップファイル(`globals.sql`、`bds_hub.dump`、`bds_hub_report.dump`)を作成します。

Black Duckコンテナを停止するには、次の手順を実行します。

1. 次のコマンドを実行してBlack Duckコンテナを停止します。これにより、以前のバージョンのBlack Duckが実行されている現在のスタックが削除されます。

```
docker stack rm hub
```

PostgreSQLデータを復元するには、次の手順を実行します。

1. `docker-swarm`ディレクトリにある`docker-compose.dbmigrate.yml`ファイルを使用します。データベースの移行に必要なコンテナとボリュームが起動します。

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

Dockerの一部のバージョンでは、イメージがプライベートリポジトリに存在する場合に、上記のコマンドに次のフラグを追加しない限り、dockerスタックはそれらをプルしません。

```
--with-registry-auth
```

2. DBコンテナが起動したら、移行スクリプトを実行します。これにより、既存のデータベースデータファイルからデータが復元されます。

❗ 重要: `docker-swarm/bin`ディレクトリにあるBlack Duckバージョン<version you're restoring from>の`hub_db_migrate.sh`スクリプトを実行する必要があります。たとえば、2019.10.0 Docker-swarmデータベースを復元する場合は、2019.10.0 `hub_db_migrate.sh`スクリプトを使用して、2019.10.0 Black Duckスタックに復元する必要があります。

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

DBコンテナが起動したら、`docker-swarm`ディレクトリにある移行スクリプトを実行します。このスクリプトは、既存のデータベースダンプファイルからデータを復元します。

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

特定のデータベースを移行するには、次の構文を使用します。

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

<db name>に使用できる値は次のとおりです。

- ・ `bds_hub`
- ・ `bds_hub_report`

これでBlack Duckをアップグレードできます。


のアップグレード Black Duck

Black Duckをアップグレードするには、次の手順を実行します。

1. 次のいずれかを実行します。

- ・ `docker-compose.local-overrides.yml`を使用して、`.yml`ファイルを変更しなかった場合は、新しいバージョンのBlack Duckの`docker-swarm`ディレクトリにあるファイルを使用して、次のいずれかのコマンドを実行します。
 - ・ `docker stack deploy -c docker-compose.yml hub` (DBコンテナを使用している場合)
 - ・ `docker stack deploy -c docker-compose.externaldb.yml hub` (外部PostgreSQLインスタンスを使用している場合)
- ・ Black Duck - Binary Analysisをアップグレードする場合は、新しいバージョンのBlack Duckの`docker-swarm`ディレクトリにあるファイルを使用して、次のコマンドのいずれかを実行します。
 - ・ `docker stack deploy -c docker-compose.yml -c docker-compose.bdba.yml hub` (DBコンテナとBlack Duck - Binary Analysisを使用している場合)
 - ・ `docker stack deploy -c docker-compose.externaldb.yml -c docker-compose.bdba.yml hub` (外部データベースとBlack Duck - Binary Analysisを使用している場合)
- ・ `docker-compose.local-overrides.yml`を使用して、`.yml`ファイルを変更した場合は、次のコマンドのいずれかを実行します。変更を含むバージョンの`docker-compose.local-overrides.yml`ファイルを使用します。他のすべてのファイルについては、新しいバージョンのBlack Duckの`docker-swarm`ディレクトリにあるファイルを使用します。
 - ・ `docker stack deploy -c docker-compose.yml -c docker-compose.local-overrides.yml hub` (DBコンテナを使用している場合)
 - ・ `docker stack deploy -c docker-compose.externaldb.yml -c docker-compose.local-overrides.yml hub` (外部PostgreSQLインスタンスを使用している場合)
 - ・ `docker stack deploy -c docker-compose.yml -c docker-compose.bdba.yml -c docker-compose.local-overrides.yml hub` (DBコンテナとBlack Duck - Binary Analysisを使用している場合)
 - ・ `docker stack deploy -c docker-compose.externaldb.yml -c docker-compose.bdba.yml -c docker-compose.local-overrides.yml hub` (外部データベースとBlack Duck - Binary Analysisを使用している場合)
- ・ Swarmサービスに対してファイルシステムが読み取り専用の状態でBlack Duckをアップグレードするには、`docker-compose.readonly.yml`ファイルを前の例に追加します。


たとえば、`docker-compose.local-overrides.yml`を使用して、`.yml`ファイルを変更し、DBコンテナを使用するBlack Duckインストールをアップグレードする場合は、次のコマンドを実行します。 `docker stack deploy -c docker-compose.yml -c docker-compose.readonly.yml -c docker-compose.local-overrides.yml hub`

 注：以前に`hub-proxy.env`ファイルを編集した場合、それらの編集内容を`blackduck-config.env`ファイルにコピーする必要があります。

7. Dockerコンテナ

Black Duckアプリケーションを構成するDockerネットワーク内のコンテナを以下に示します。

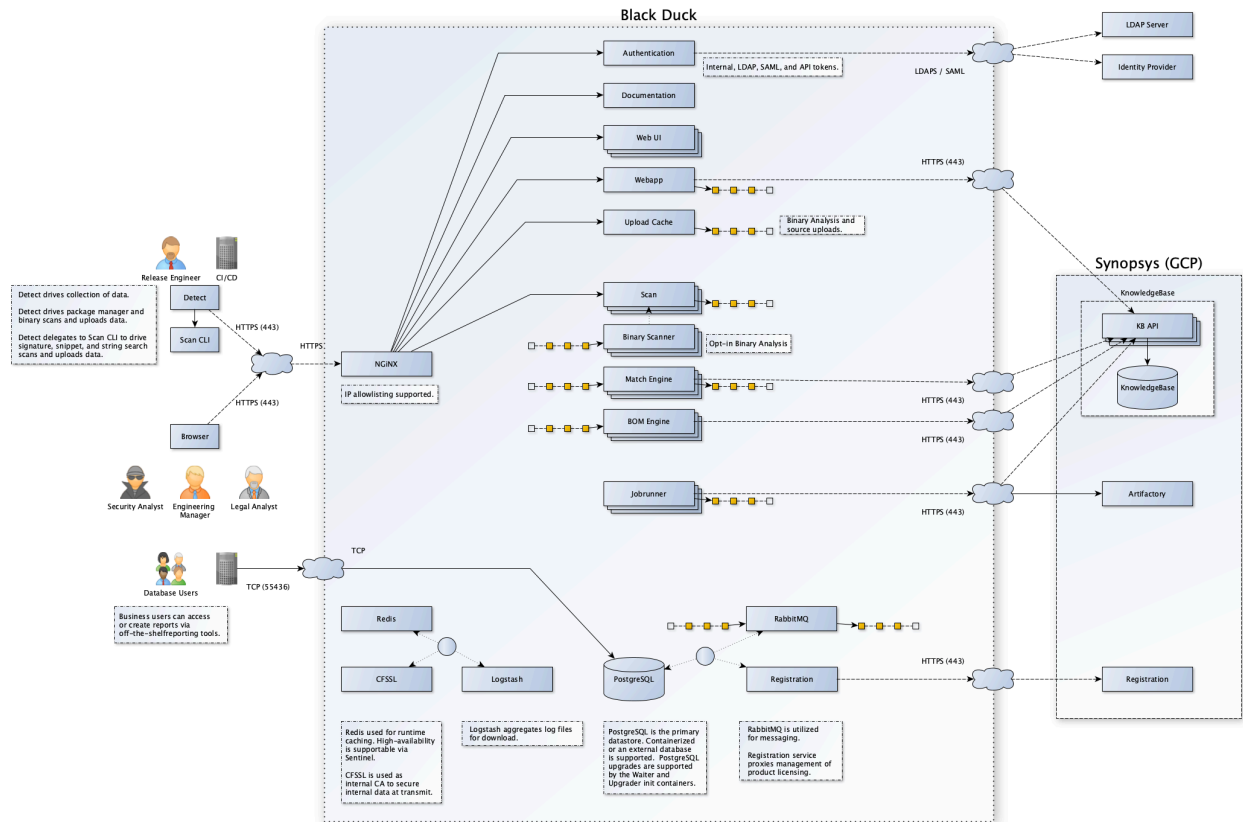
1. Authentication
2. CA
3. DB

 注：外部Postgreインスタンスを使用する場合、このコンテナはBlack Duckアプリケーションに含まれません。

4. Documentation
5. Jobrunner
6. Logstash
7. Registration
8. スキャン
9. Uploadcache – このコンテナはBlack Duck – Binary Analysisにも使用されます。
10. Webapp
11. Webserver
12. Redis
13. Rabbitmq
14. Bomengine
15. WebUI
16. Matchengine

Black Duck – Binary Analysisが有効になっている場合は、Binaryscannerコンテナが必要です。

次の図は、コンテナ間の基本的な関係と、Dockerネットワークの外部に公開されるポートを示しています。



ZookeeperコンテナはBlack Duckバージョン2020.4.0で削除されました。次のzookeeperボリュームは使用されなくなったため、手動で削除できます。

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

次の表に、各コンテナに関する詳細情報を示します。

Authenticationコンテナ


コンテナ名	blackduck-authentication
イメージ名	blackducksoftware/blackduck-authentication:2022.10.0
説明	authenticationサービスは、すべての認証関連リクエストの宛先となるコンテナです。
スケーラビリティ	このコンテナのインスタンスは1つだけである必要があります。現在、拡張することはできません。
リンク/ポート	<p>外部に公開されているものではありません(8443は内部に公開)。このコンテナは、次の他のコンテナ/サービスに接続する必要があります。</p> <ul style="list-style-type: none"> • postgres • cfssl • logstash • registration • webapp

コンテナ名 : blackduck-authentication	
	コンテナは、それにリンクする他のコンテナに8443を公開する必要があります。
代替ホスト名の環境変数	<p>他のタイプのオーケストレーションで実行する場合、Docker Swarmがデフォルトで使用するのではないホスト名をこれらのコンテナに設定すると便利な場合があります。これらの環境変数は、デフォルトのホスト名を上書きするように設定できます。</p> <ul style="list-style-type: none"> ・ postgres – \$HUB_Postgre_HOST ・ cfssl – \$HUB_CFSSL_HOST ・ logstash – \$HUB_LOGSTASH_HOST ・ registration – \$HUB_REGISTRATION_HOST ・ webapp – \$HUB_WEBAPP_HOST
リソース/制約	<ul style="list-style-type: none"> ・ デフォルトの最大Javaヒープサイズ: 512 MB ・ コンテナメモリ: 1 GB ・ コンテナCPU: 1 CPU
ユーザー/グループ	<p>このコンテナはUID 100として実行されます。コンテナがUID 0 (root)として起動された場合、メインプロセスを実行する前に、ユーザーはUID 100:rootに切り替えられます。</p> <p>このコンテナは、ルートグループ (GID/fsGroup 0) 内でも起動される場合、ランダムUIDとして起動することもできます。</p>
環境ファイル	blackduck-config.env

CAコンテナ

コンテナ名 : blackduck-cfssl	
イメージ名	blackducksoftware/blackduck-cfssl:1.0.2
説明	<p>このコンテナは、CFSSLを使用し、これはPostgreSQL、NGiNX、およびPostgreに対して認証する必要があるクライアントの証明書生成に使用されます。このコンテナは、アプリケーションを構成する内部コンテナのTLS証明書を生成するのにも使用されます。</p>
スケーラビリティ	このコンテナのインスタンスは1つだけである必要があります。スケーリングしてはいけません。
リンク/ポート	コンテナは、Dockerネットワーク内で、それにリンクする他のコンテナ/サービスにポート8888を公開する必要があります。
リソース/制約	<ul style="list-style-type: none"> ・ デフォルトの最大Javaヒープサイズ: 該当なし ・ コンテナメモリ: 512 MB ・ コンテナCPU: 未指定
ユーザー/グループ	<p>このコンテナはUID 100として実行されます。コンテナがUID 0 (root)として起動された場合、メインプロセスを実行する前に、ユーザーはUID 100:rootに切り替えられます。</p> <p>このコンテナは、ルートグループ (GID/fsGroup 0) 内でも起動される場合、ランダムUIDとして起動することもできます。</p>
環境ファイル	blackduck-config.env

DBコンテナ

 注：外部Postgreインスタンスを使用する場合、このコンテナはBlack Duckアプリケーションに含まれません。

コンテナ名 : blackduck-postgres	
イメージ名	blackducksoftware/blackduck-postgres:9.6-1.1
説明	<p>DBコンテナには、オープンソースのオブジェクトリレーショナルデータベースシステムであるPostgreSQLデータベースが格納されます。アプリケーションはPostgreSQLデータベースを使用してデータを格納します。</p> <p>このコンテナには1つのインスタンスがあります。これは、アプリケーションのすべてのデータが格納される場所です。Postgreのポートは2つあります。1つのポートはDockerネットワーク内のコンテナに公開されます。これはアプリケーションが使用する接続です。このポートは証明書認証によって保護されています。もう1つのポートはDockerネットワークの外部に公開されます。これにより、読み取り専用ユーザーは、hub_reportdb_changepassword.shスクリプトを使用して設定されたパスワードを介して接続できます。このポートとユーザーは、レポートとデータ抽出に使用できます。</p> <p>レポートデータベースの詳細については、『レポートデータベース』ガイドを参照してください。</p>
スケーラビリティ	このコンテナのインスタンスは1つだけである必要があります。スケーリングしてはいけません。
リンク/ポート	<p>DBコンテナは、次のコンテナ/サービスに接続する必要があります。</p> <ul style="list-style-type: none"> logstash cfssl <p>コンテナは、Dockerネットワーク内で、それにリンクする他のコンテナにポート5432を公開する必要があります。</p> <p>このコンテナは、データベースのレポート用にDockerネットワークの外部にポート55436を公開します。</p>
代替ホスト名の環境変数	<p>他のタイプのオーケストレーションで実行する場合、Docker Swarmがデフォルトで使用するのではないホスト名をこれらのコンテナに設定すると便利な場合があります。これらの環境変数は、デフォルトのホスト名を上書きするように設定できます。</p> <ul style="list-style-type: none"> logstash: \$HUB_LOGSTASH_HOST cfssl: \$HUB_CFSSL_HOST
リソース/制約	<ul style="list-style-type: none"> デフォルトの最大Javaヒープサイズ: 該当なし コンテナメモリ: 3 GB コンテナCPU: 1 CPU
ユーザー/グループ	<p>このコンテナはUID 70として実行されます。コンテナがUID 0(root)として起動された場合、メインプロセスを実行する前に、ユーザーはUID 70:rootに切り替えられます。</p> <p>このコンテナは、他のユーザーIDで開始することはできません。</p>
環境ファイル	該当なし

Documentationコンテナ

コンテナ名: blackduck-documentation	
イメージ名	blackducksoftware/blackduck-documentation: 2022.10.0
説明	Documentationコンテナは、アプリケーションのドキュメントを提供します。
スケーラビリティ	このコンテナには1つのインスタンスがあります。スケーリングしてはいけません。
リンク/ポート	<p>このコンテナは、次の他のコンテナ/サービスに接続する必要があります。</p> <ul style="list-style-type: none"> logstash cfssl <p>Documentationコンテナは、それにリンクする他のコンテナにポート8443を公開する必要があります。</p>
代替ホスト名の環境変数	<p>他のタイプのオーケストレーションで実行する場合、Docker Swarmがデフォルトで使用するのではないホスト名をこれらのコンテナに設定すると便利な場合があります。これらの環境変数は、デフォルトのホスト名を上書きするように設定できます。</p> <ul style="list-style-type: none"> logstash: \$HUB_LOGSTASH_HOST cfssl: \$HUB_CFSSL_HOST
リソース/制約	<ul style="list-style-type: none"> デフォルトの最大Javaヒープサイズ: 512 MB コンテナメモリ: 512 MB コンテナCPU: 未指定
ユーザー/グループ	<p>このコンテナはUID 8080として実行されます。コンテナがUID 0 (root)として起動された場合、メインプロセスを実行する前に、ユーザーはUID 8080:rootに切り替えられます。</p> <p>このコンテナは、ルートグループ(GID/fsGroup 0)内でも起動される場合、ランダムUIDとして起動することもできます。</p>
環境ファイル	blackduck-config.env

Jobrunnerコンテナ

コンテナ名: blackduck-jobrunner	
イメージ名	blackducksoftware/blackduck-jobrunner: 2022.10.0
説明	Job Runnerコンテナは、アプリケーションのすべてのジョブを実行するコンテナです。これには、マッチング、構成表の作成、レポート、データ更新などが含まれます。このコンテナには公開されているポートはありません。
スケーラビリティ	このコンテナはスケーリングできます。
リンク/ポート	<p>Job Runnerコンテナは、次のコンテナ/サービスに接続する必要があります。</p> <ul style="list-style-type: none"> postgres registration logstash

コンテナ名: blackduck-Jobrunner	
	<ul style="list-style-type: none"> cfssl
代替ホスト名の環境変数	<p>他のタイプのオーケストレーションで実行する場合、個々のサービス名が異なる場合があります。たとえば、別のサービス名を使用して解決された外部PostgreSQLエンドポイントがあるとして。このようなユースケースの場合は、これらの環境変数を設定して、デフォルトのホスト名を上書きすることができます。</p> <ul style="list-style-type: none"> postgres: \$HUB_Postgre_HOST registration: \$HUB_REGISTRATION_HOST logstash: \$HUB_LOGSTASH_HOST cfssl: \$HUB_CFSSL_HOST
リソース/制約	<ul style="list-style-type: none"> デフォルトの最大Javaヒープサイズ: 4 GB コンテナメモリ: 4.5 GB コンテナCPU: 1 CPU
ユーザー/グループ	<p>このコンテナはUID 100として実行されます。コンテナがUID 0(root)として起動された場合、メインプロセスを実行する前に、ユーザーはUID 100:rootに切り替えられます。</p> <p>このコンテナは、ルートグループ(GID/fsGroup 0)内でも起動される場合、ランダムUIDとして起動することもできます。</p>
環境ファイル	blackduck-config.env

Logstashコンテナ

コンテナ名: blackduck-logstash	
イメージ名	blackducksoftware/blackduck-logstash:1.0.10
説明	Logstashコンテナは、すべてのコンテナのログを収集して格納します。
スケーラビリティ	このコンテナのインスタンスは1つだけである必要があります。スケーリングしてはいけません。
リンク/ポート	コンテナは、Dockerネットワーク内で、それにリンクする他のコンテナ/サービスにポート5044を公開する必要があります。
リソース/制約	<ul style="list-style-type: none"> デフォルトの最大Javaヒープサイズ: 1 GB コンテナメモリ: 1 GB コンテナCPU: 未指定
ユーザー/グループ	<p>このコンテナはUID 100として実行されます。コンテナがUID 0(root)として起動された場合、メインプロセスを実行する前に、ユーザーはUID 100:rootに切り替えられます。</p> <p>このコンテナは、ルートグループ(GID/fsGroup 0)内でも起動される場合、ランダムUIDとして起動することもできます。</p>
環境ファイル	blackduck-config.env

Registrationコンテナ

コンテナ名: blackduck-registration	
イメージ名	blackducksoftware/blackduck-registration:2022.10.0
説明	コンテナは、他のコンテナからの登録リクエストを処理する小規模なサービスです。このコンテナは定期的にBlack Duck登録サービスに接続して、登録の更新を取得します。
スケーラビリティ	コンテナをスケーリングしてはいけません。
リンク/ポート	Registrationコンテナは、次のコンテナ/サービスに接続する必要があります。 <ul style="list-style-type: none"> logstash cfssl コンテナは、それにリンクする他のコンテナにポート8443を公開する必要があります。
代替ホスト名の環境変数	他のタイプのオーケストレーションで実行する場合、Docker Swarmがデフォルトで使用するのではないホスト名をこれらのコンテナに設定すると便利な場合があります。これらの環境変数は、デフォルトのホスト名を上書きするように設定できます。 <ul style="list-style-type: none"> logstash: \$HUB_LOGSTASH_HOST cfssl: \$HUB_CFSSL_HOST
リソース/制約	<ul style="list-style-type: none"> デフォルトの最大Javaヒープサイズ: 512 MB コンテナメモリ: 640 MB コンテナCPU: 未指定
ユーザー/グループ	このコンテナはUID 8080として実行されます。コンテナがUID 0 (root)として起動された場合、メインプロセスを実行する前に、ユーザーはUID 8080:rootに切り替えられます。このコンテナは、ルートグループ (GID/fsGroup 0) 内でも起動される場合、ランダムUIDとして起動することもできます。
環境ファイル	blackduck-config.env

Scanコンテナ

コンテナ名: blackduck-scan	
イメージ名	blackducksoftware/blackduck-scan:2022.10.0
説明	scanサービスは、すべてのスキャンデータリクエストの宛先となるコンテナです。
スケーラビリティ	このコンテナはスケーリングできます。
リンク/ポート	このコンテナは、次のコンテナ/サービスに接続する必要があります。 <ul style="list-style-type: none"> postgres registration logstash cfssl

コンテナ名: blackduck-scan	
	コンテナは、それにリンクする他のコンテナにポート8443を公開する必要があります。
代替ホスト名の環境変数	<p>他のタイプのオーケストレーションで実行する場合、Docker Swarmがデフォルトで使用するのではないホスト名をこれらのコンテナに設定すると便利な場合があります。これらの環境変数は、デフォルトのホスト名を上書きするように設定できます。</p> <ul style="list-style-type: none"> ・ postgres: \$HUB_Postgre_HOST ・ registration: \$HUB_REGISTRATION_HOST ・ logstash: \$HUB_LOGSTASH_HOST ・ cfssl: \$HUB_CFSSL_HOST
リソース/制約	<ul style="list-style-type: none"> ・ デフォルトの最大Javaヒープサイズ: 2 GB ・ コンテナメモリ: 2.5 GB ・ コンテナCPU: 1 CPU
ユーザー/グループ	<p>このコンテナはUID 8080として実行されます。コンテナがUID 0(root)として起動された場合、メインプロセスを実行する前に、ユーザーはUID 8080:rootに切り替えられます。</p> <p>このコンテナは、ルートグループ(GID/fsGroup 0)内でも起動される場合、ランダムUIDとして起動することもできます。</p>
環境ファイル	blackduck-config.env

Uploadcacheコンテナ

コンテナ名: Uploadcache	
イメージ名	blackducksoftware/blackduck-upload-cache: 1.0.17
説明	このコンテナは、バイナリ解析用のアップロードを一時的に格納するのに使用されます。また、スニペットマッチング用にソースファイルをアップロードするときに使用されます。
スケーラビリティ	このコンテナのインスタンスは1つだけである必要があります。スケーリングしてはいけません。
リンク/ポート	<p>このコンテナは、次のコンテナ/サービスに接続する必要があります。</p> <ul style="list-style-type: none"> ・ cfssl ・ logstash ・ rabbitmq (Black Duck - Binary Analysisが有効な場合) <p>コンテナは、それにリンクする他のコンテナにポート9443と9444を公開します。</p>
代替ホスト名の環境変数	<p>他のタイプのオーケストレーションで実行する場合、Docker Swarmがデフォルトで使用するのではないホスト名をこれらのコンテナに設定すると便利な場合があります。これらの環境変数は、デフォルトのホスト名を上書きするように設定できます。</p> <ul style="list-style-type: none"> ・ cfssl: \$HUB_CFSSL_HOST ・ logstash: \$HUB_LOGSTASH_HOST ・ rabbitmq: \$RABBIT_MQ_HOST

コンテナ名: Uploadcache	
制約	<ul style="list-style-type: none"> ・ デフォルトの最大Javaヒープサイズ: 該当なし ・ コンテナメモリ: 512 MB ・ コンテナCPU: 未指定
ユーザー/グループ	このコンテナはUID 100として実行されます。コンテナがUID 0 (root)として起動された場合、メインプロセスを実行する前に、ユーザーはUID 100:rootに切り替えられます。このコンテナは、ルートグループ (GID/fsGroup 0) 内でも起動される場合、ランダムUIDとして起動することもできます。
環境ファイル	hub-bdba.env blackduck-config.env

Webappコンテナ

コンテナ名: blackduck-webapp	
イメージ名	blackducksoftware/blackduck-webapp: 2022.10.0
説明	webappコンテナは、すべてのWeb/UI/APIリクエストの宛先となるコンテナです。また、あらゆるUIリクエストも処理します。図では、webappのポートはDockerネットワークの外部に公開されていません。代わりに、Dockerネットワークの外部に公開されているNGiNXリバースプロキシ (WebServerコンテナを参照) があります。
スケーラビリティ	このコンテナのインスタンスは1つだけである必要があります。スケーリングしてはいけません。
リンク/ポート	webappコンテナは、次のコンテナ/サービスに接続する必要があります。 <ul style="list-style-type: none"> ・ postgres ・ registration ・ logstash ・ cfssl コンテナは、それにリンクする他のコンテナにポート8443を公開する必要があります。
代替ホスト名の環境変数	他のタイプのオーケストレーションで実行する場合、Docker Swarmがデフォルトで使用するのではないホスト名をこれらのコンテナに設定すると便利な場合があります。これらの環境変数は、デフォルトのホスト名を上書きするように設定できます。 <ul style="list-style-type: none"> ・ postgres: \$HUB_Postgre_HOST ・ registration: \$HUB_REGISTRATION_HOST ・ logstash: \$HUB_LOGSTASH_HOST ・ cfssl: \$HUB_CFSSL_HOST
リソース/制約	<ul style="list-style-type: none"> ・ デフォルトの最大Javaヒープサイズ: 2 GB ・ コンテナメモリ: 2.5 GB ・ コンテナCPU: 1 CPU

コンテナ名 : blackduck-webapp	
ユーザー/グループ	このコンテナはUID 8080として実行されます。コンテナがUID 0(root)として起動された場合、メインプロセスを実行する前に、ユーザーはUID 8080:rootに切り替えられます。 このコンテナは、ルートグループ(GID/fsGroup 0)内でも起動される場合、ランダムUIDとして起動することもできます。
環境ファイル	blackduck-config.env

Webserverコンテナ

コンテナ名 : blackhack-webserver	
イメージ名	blackducksoftware/blackduck-nginx:1.0.32
説明	WebServerコンテナは、アプリケーションを含むコンテナのリバースプロキシです。Dockerネットワークの外部に公開されているポートがあります。これは、HTTPS用に構成されたコンテナです。ここにはHTTPSの構成を可能にする構成ボリュームがあります。 HTTP/2およびTLS 1.3がサポートされています
スケーラビリティ	コンテナをスケーリングしてはいけません。
リンク/ポート	Web Appコンテナは、次のコンテナ/サービスに接続する必要があります。 <ul style="list-style-type: none"> webapp cfssl documentation scan authentication upload cache (Black Duck – Binary Analysisが有効な場合) このコンテナは、Dockerネットワークの外部にポート443を公開します。
代替ホスト名の環境変数	他のタイプのオーケストレーションで実行する場合、Docker Swarmがデフォルトで使用するのではないホスト名をこれらのコンテナに設定すると便利な場合があります。これらの環境変数は、デフォルトのホスト名を上書きするように設定できます。 <ul style="list-style-type: none"> webapp: \$HUB_WEBAPP_HOST cfssl: \$HUB_CFSSL_HOST scan: \$HUB_SCAN_HOST documentation: \$HUB_DOC_HOST authentication: \$HUB_AUTHENTICATION_HOST upload cache: \$HUB_UPLOAD_CACHE_HOST
リソース/制約	<ul style="list-style-type: none"> デフォルトの最大Javaヒープサイズ: 該当なし コンテナメモリ: 512 MB コンテナCPU: 未指定
ユーザー/グループ	このコンテナはUID 100として実行されます。コンテナがUID 0(root)として起動された場合、メインプロセスを実行する前に、ユーザーはUID 100:rootに切り替えられます。 このコンテナは、ルートグループ(GID/fsGroup 0)内でも起動される場合、ランダムUIDとして起動することもできます。

コンテナ名 : blackhack-webserver	
環境ファイル	hub-webserver.env、blackduck-config.env

Redisコンテナ

コンテナ名 : blackduck-redis	
イメージ名	blackducksoftware/blackduck-redis:2022.10.0
説明	<p>このコンテナでは、Black Duckのキャッシュ機能の整合性が向上し、アプリケーションのパフォーマンスが改善します。</p> <p>Redisは、プライマリキャッシュメカニズムとしてデフォルトで有効になっています。</p>
構成	<p>Redisを構成するには、次の設定を使用します。</p> <ul style="list-style-type: none"> Redis設定を構成するためのblackduck-config.env環境ファイル <ul style="list-style-type: none"> Redis設定: <ul style="list-style-type: none"> BLACKDUCK_REDIS_MODE: <p>Redisモードはスタンドアロンまたはセンチネルにすることができます</p> BLACKDUCK_REDIS_TLS_ENABLED: <p>Redisクライアントとサーバーの間でTLS/SSL接続を強制するかどうかを指定できます。</p> Redisスタンドアロンモードにはdocker-compose.ymlを使用します。このモードは、1,024 MBの追加メモリを必要とします Redisセンチネルモードには、docker-compose.ymlとdocker-compose.redis.sentinel.ymlの両方を使用します。このモードは高可用性を提供しますが、3,168 MBの追加メモリを必要とします。 Redisコンテナは、他のサービスと同様にlogstashおよびfilebeatと統合されます。 Redisコンテナには、正常性状態チェックがあります。 Black Duck Hubシステム情報ページには、Redisデバッグ情報が表示されるredis-cacheタブがあります。
スケーラビリティ	コンテナをスケーリングしてはいけません。
リンク/ポート	<p>Redisコンテナは、次のコンテナ/サービスに接続する必要があります。</p> <ul style="list-style-type: none"> logstash cfssl
代替ホスト名の環境変数	該当なし
リソース/制約	<ul style="list-style-type: none"> デフォルトの最大Javaヒープサイズ: 該当なし コンテナメモリ: <ul style="list-style-type: none"> スタンドアロン1,024 MB センチネルモード3,168 MB コンテナCPU: 1
ユーザー/グループ	<p>任意のユーザー/グループとして実行できます。次に例を示します。</p> <pre>{ "runAsUser": 4567, "runAsGroup": 4567 }</pre>

コンテナ名: blackduck-redis	
環境ファイル	blackduck-config.env

Rabbitmqコンテナ

コンテナ名: rabbitmq	
イメージ名	blackducksoftware/rabbitmq: 1.2.2
説明	このコンテナにより、バイナリ解析ワーカーに情報を簡単にアップロードできます。また、これによりbomengineコンテナはBOM計算の開始通知を受信できるようになります。Dockerネットワーク内にはポートを公開しますが、Dockerネットワークの外部には公開しません。
スケーラビリティ	このコンテナのインスタンスは1つだけである必要があります。スケーリングしてはいけません。
リンク/ポート	このコンテナは、次の他のコンテナ/サービスに接続する必要があります。 <ul style="list-style-type: none"> cfssl コンテナは、それにリンクする他のコンテナにポート5671を公開する必要があります。
代替ホスト名の環境変数	他のタイプのオーケストレーションで実行する場合、Docker Swarmがデフォルトで使用するのではないホスト名をこれらのコンテナに設定すると便利な場合があります。これらの環境変数は、デフォルトのホスト名を上書きするように設定できます。 <ul style="list-style-type: none"> cfssl: \$HUB_CFSSL_HOST
制約	<ul style="list-style-type: none"> デフォルトの最大Javaヒープサイズ: 該当なし コンテナメモリ: 1 GB コンテナCPU: 未指定
ユーザー/グループ	このコンテナはUID 100として実行されます。コンテナがUID 0 (root)として起動された場合、メインプロセスを実行する前に、ユーザーはUID 100:rootに切り替えられます。 このコンテナは、ルートグループ (GID/fsGroup 0) 内でも起動される場合、ランダムUIDとして起動することもできます。
環境ファイル	blackduck-config.env

bomengineコンテナ

コンテナ名: bomengine	
イメージ名	blackducksoftware/blackduck-bomengine: 2022.10.0
説明	bomengineコンテナは、構成表を作成し、それらを最新の状態に保ちます。
スケーラビリティ	コンテナはスケーリングできます
リンク/ポート	bomengineコンテナは、次のコンテナ/サービスに接続する必要があります。

コンテナ名 : bomengine	
	<ul style="list-style-type: none"> ・ postgres ・ registration ・ logstash ・ cfssl
代替ホスト名の環境変数	<p>他のタイプのオーケストレーションで実行する場合、個々のサービス名が異なる場合があります。たとえば、別のサービス名を使用して解決された外部PostgreSQLエンドポイントがあるとします。このようなユースケースの場合は、これらの環境変数を設定して、デフォルトのホスト名を上書きすることができます。</p> <ul style="list-style-type: none"> ・ postgres: \$HUB_Postgre_HOST ・ registration: \$HUB_REGISTRATION_HOST ・ logstash: \$HUB_LOGSTASH_HOST ・ cfssl: \$HUB_CFSSL_HOST
リソース/制約	<ul style="list-style-type: none"> ・ デフォルトの最大Javaヒープサイズ: 4 GB ・ コンテナメモリ: 4.5 GB
ユーザー/グループ	<p>このコンテナはUID 100として実行されます コンテナがUID 0(root)として起動された場合、メインプロセスを実行する前に、ユーザーはUID 100:rootに切り替えられます。このコンテナは、ルートグループ(GID/fsGroup 0)内でも起動される場合、ランダムUIDとして起動することもできます。</p>
環境ファイル	blackduck-config.env

blackduck-matchengineコンテナ

コンテナ名 : matchengine	
イメージ名	blackducksoftware/blackduck-matchengine:2022.10.0
説明	クラウドナレッジベースからコンポーネントマッチ情報を取得します。
スケーラビリティ	このコンテナには複数のインスタンスが存在する場合があります。
リンク/ポート	<ul style="list-style-type: none"> ・ ポートを公開しません。 ・ 外部からクラウドナレッジベースサービスに接続します 次のサービスに内部的に接続します。 ・ cffsl ・ logstash ・ registration ・ postgres ・ rabbitmq
環境変数	<p>環境変数は次のとおりです。</p> <ul style="list-style-type: none"> ・ HUB_CFSSL_PORTHUB_MATCHENGINE_HOST ・ HUB_MAX_MEMORY ・ HUB_REGISTRATION_HOST ・ HUB_REGISTRATION_PORTHUB_Postgre_HOST

コンテナ名 : matchengine	
	<ul style="list-style-type: none"> ・ HUB_Postgre_PORT ・ RABBIT_MQ_HOST ・ RABBIT_MQ_PORT
リソース/制約	<ul style="list-style-type: none"> ・ デフォルトの最大Javaヒープサイズ: 1 GB ・ コンテナメモリ: 予約値: 1 GB、上限値: 1.5 GB ・ コンテナCPU: 予約値: 0.5、上限値: 1
ユーザー/グループ	このコンテナはUID 100として実行されます コンテナがUID 0(root)として起動された場合、メインプロセスを実行する前に、ユーザーはUID 100:rootに切り替えられます。このコンテナは、ルートグループ(GID/fsGroup 0)内でも起動される場合、ランダムUIDとして起動することもできます。
環境ファイル	blackduck-config.env

webuiコンテナ

コンテナ名 : webui	
イメージ名	イメージ名 : blackducksoftware/blackduck-webui:2021.6.0
説明	ユーザーインターフェイスのすべてのリソースを管理するNode.jsコンテナです。Node.jsは、ChromeのV8 JavaScriptエンジンをベースに構築されたオープンソースアプリケーションサーバーです。
スケーラビリティ	このコンテナのインスタンスは1つだけである必要があります。
リンク/ポート	<p>このコンテナは、次のコンテナ/サービスに接続する必要があります。</p> <ul style="list-style-type: none"> ・ cfssl ・ logstash <p>このコンテナは、nginxコンテナからのプロキシ処理されたリクエスト用にポート8443を公開します。Dockerネットワークの外部にはポートを公開しません。</p>
リソース/制約	<ul style="list-style-type: none"> ・ デフォルトの最大Javaヒープサイズ: 該当なし ・ コンテナメモリ: 640 MB ・ コンテナCPU: 未指定

Black Duck – Binary Analysis コンテナ

次のコンテナは、Binaryscannerコンテナがある場合にのみインストールされますBlack Duck – Binary Analysis

Binaryscannerコンテナ

コンテナ名 : bdba-worker	
イメージ名	イメージ : sigsynopsys/bdba-worker : 2021.03
説明	このコンテナはバイナリファイルを解析します。

コンテナ名 : bdba-worker	
	このコンテナは現在、Black Duck – Binary Analysisが有効な場合にのみ使用されます。
スケーラビリティ	このコンテナはスケーリングできます。
リンク/ポート	<p>このコンテナは、次のコンテナ/サービスに接続する必要があります。</p> <ul style="list-style-type: none"> ・ cfssl ・ logstash ・ rabbitmq ・ webserver <p>コンテナは、それにリンクする他のコンテナにポート5671を公開する必要があります。</p>
代替ホスト名の環境変数	<p>他のタイプのオーケストレーションで実行する場合、Docker Swarmがデフォルトで使用するのではないホスト名をこれらのコンテナに設定すると便利な場合があります。これらの環境変数は、デフォルトのホスト名を上書きするように設定できます。</p> <ul style="list-style-type: none"> ・ cfssl: \$HUB_CFSSL_HOST ・ logstash: \$HUB_LOGSTASH_HOST ・ rabbitmq: \$RABBIT_MQ_HOST ・ webserver: \$HUB_WEBSERVER_HOST
リソース/制約	<ul style="list-style-type: none"> ・ デフォルトの最大Javaヒープサイズ: 該当なし ・ コンテナメモリ: 2 GB ・ コンテナCPU: 1 CPU
ユーザー/グループ	このコンテナはUID 0として実行されます。
環境ファイル	hub-bdba.env