



Kubernetesを使用したBlack Duck のインストール

Black Duck 2023.10.0

Copyright ©2023 by Synopsys.

All rights reserved.本ドキュメントの使用はすべて、Black Duck Software, Inc.とライセンス所有者の間の使用許諾契約に準拠します。本ドキュメントのいかなる部分も、Black Duck Software, Inc.の書面による許諾を受けることなく、どのような形態または手段によっても、複製または譲渡することが禁じられています。

Black Duck、Know Your Code、およびBlack Duckロゴは、米国およびその他の国におけるBlack Duck Software, Inc.の登録商標です。Black Duck Code Center、Black Duck Code Sight、Black Duck Hub、Black Duck Protex、およびBlack Duck Suiteは、Black Duck Software, Inc.の商標です。他の商標および登録商標はすべてそれぞれの所有者が保有しています。

12-03-2024

目次

まえがき.....	4
Black Duck ドキュメント.....	4
カスタマサポート.....	4
Synopsys Software Integrityコミュニティ.....	5
トレーニング.....	5
包括性と多様性に関するSynopsysの声明.....	6
Synopsysのセキュリティへの取り組み.....	6
 1. Kubernetesを使用したBlack Duckのインストール.....	7
 2. ハードウェア要件.....	8
 3. PostgreSQLのバージョン.....	10
 4. Helmを使用したBlack Duckのインストール.....	11
 5. Artifactory Integration.....	12
Artifactory Integrationの前提条件.....	12
インストールの順序.....	13
Artifactory Integrationのインストール.....	14
Artifactory プラグインのインストール(非推奨).....	16
Artifactory Integrationプラグインのインストール.....	16
Artifactoryプラグインの構成(非推奨).....	17
Artifactory Integrationプラグインの構成.....	19
接続のテスト.....	20
Artifactory Integrationタスク.....	20
Artifactory Integrationの設定.....	24
 6. 管理タスク.....	26
Kubernetesでのシークレットの暗号化の構成.....	26
Kubernetesでのシードの生成.....	27
バックアップシードの構成.....	27
Kubernetesでのシークレットローテーションの管理.....	28
Blackduck Storageのカスタムボリュームの構成.....	29
jobrunnerスレッドプールの設定.....	32

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。
Kubernetesを使用したBlack Duckのインストール	install_kubernetes.pdf	Kubernetes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。
OpenShiftを使用したBlack Duckのインストール	install_openshift.pdf	OpenShift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の使用方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には、Helmを使用します。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。Black DuckはHelm3をサポートしており、Kubernetesの最小バージョンは1.13です。

Black Duck 統合に関するドキュメントは、次のリンクから入手できます：

- ・ <https://sig-product-docs.synopsys.com/bundle/integrations-detect/page/integrations/integrations.html>
- ・ https://sig-product-docs.synopsys.com/category/cicd_integrations

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のチュートリ

アルについては、Black Duck UIの[ヘルプ]メニュー()から選択してください。

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。


Synopsysのセキュリティへの取り組み

Synopsys Software Integrityグループ(SIG)は、お客様のアプリケーションの保護とセキュリティの確保に専念する組織として、お客様のデータセキュリティとプライバシーにも同様に取り組んでいます。この声明は、SIGのお客様と将来のお客様に、当社のシステム、コンプライアンス認証、プロセス、その他のセキュリティ関連活動に関する最新情報をお届けすることを目的としています。

この声明は次の場所で入手できます。[セキュリティへの取り組み | Synopsys](#)

1. Kubernetesを使用したBlack Duckのインストール

Kubernetesは、コンテナを介してクラウドワークロードを管理するためのオーケストレーションツールです。

 **警告**： Black Duck 2023.7.0リリース時点で、Synopsysctlはサポートされなくなり、更新も行われなくなります。Synopsysctlのドキュメントは、<https://github.com/blackducksoftware/hub/tree/master/kubernetes/blackduck>にあります。

2. ハードウェア要件

Black Duckハードウェアのスケーリングガイドライン

スケーラビリティのサイジングに関するガイドラインについては、「[Black Duckハードウェアのスケーリングガイドライン](#)」をご参照ください。

Black Duckデータベース

⚠ 危険：Synopsysのテクニカルサポート担当者から指示がない限り、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。

ディスク容量の要件

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

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:

```
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。

3. PostgreSQLのバージョン


Black Duck 2023.10.0では、新しいPostgreSQLの機能がサポートされており、Black Duckサービスのパフォーマンスと信頼性が向上します。Black Duck 2023.10.0の時点では、内部PostgreSQLコンテナ用にサポートされているPostgreSQLのバージョンはPostgreSQL 14です。

Black Duck 2023.10.0以降、PostgreSQLコンテナを使用する導入では、PostgreSQLの設定は自動で設定されます。外部PostgreSQLを使用するお客様は、設定を引き続き手動で適用する必要があります。

PostgreSQLコンテナを使用してBlack Duckのバージョン2022.2.0～2023.7.x(表記を含む)からアップグレードするお客様は、PostgreSQL 14へ自動で移行されます。さらに古いバージョンのBlack Duckからアップグレードするお客様は、2023.10.0へアップグレードする前に、2023.7.xへアップグレード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最新バージョンのPostgreSQL 15を使用することをお勧めします。

 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。

4. Helmを使用したBlack Duckのインストール

Helmチャートは、HelmがBlack Duckを導入するのに必要なKubernetesのリソースセットを示しています。Black DuckはHelm3をサポートしており、Kubernetesの最小バージョンは1.13です。

Helmチャートは、<https://sig-repo.synopsys.com/artifactory/sig-cloudnative>から入手できます

Helmを使用してBlack Duckをインストールする手順については、[ここ](#)をクリックしてください。Helmチャートは、Helmパッケージマネージャを使用して、Kubernetesクラスタ上でBlack Duckの導入をbootstrapします。

Helmを使用したKubernetes上での移行

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、この移行ではCentOS PostgreSQLコンテナの使用がSynopsys提供のコンテナに置き換えられます。また、synopsys-initコンテナは、blackduck-postgres-waiterコンテナに置き換えられます。


プレーンなKubernetesでは、上書きされない限り、アップグレードジョブのコンテナはルートとして実行されます。ただし、唯一の要件は、ジョブがPostgreSQLデータボリュームの所有者と同じUID（デフォルトではUID=26）で実行されることです。

OpenShiftでは、アップグレードジョブは、PostgreSQLデータボリュームの所有者と同じUIDで実行されることを前提としています。

5. Artifactory Integration

概要


Artifactory Integrationは、ソフトウェアサプライチェーンを保護するためのBlack Duckメカニズムです。通常、Artifactoryはそのチェーンの最後のリンクの1つであるため、構成された一連のArtifactoryリポジトリ内で全アーティファクトをスキャンすることで、個々のサプライチェーンを制御できるようになります。デフォルトでは、このバージョンのArtifactory Integrationでは、スキャンしたArtifactoryリポジトリにBlack Duckポリシー違反があった場合、そのダウンロードが自動でブロックされます。RapidまたはBothとして定義されているBlack Duckポリシーは、Artifactory Integrationに適用されます。

 注：Artifactory Integrationを使用すると、Artifactory IntegrationプラグインのScanner機能とInspector機能が無効になります。

アーキテクチャ

Black Duck 2023.10.0では、完全ホスト型の導入を最優先でサポートするよう、Artifactory Integrationのアーキテクチャアプローチが改良されました。これらの変更については、以下のアーキテクチャ図をご参照ください。

Artifactory Integrationの前提条件

 注：この機能を活用するには、Artifactory Integrationをお使いの登録キーで有効にする必要があります。有効にしたら、以下をvalues.yamlファイルに追加します。

```
enableIntegration: true
```

カテゴリ	要件
HelmとKubernetes	<ul style="list-style-type: none"> Kubernetes 1.9以降 Helm3 SynopsysリポジトリをHelmリポジトリに追加します。 <pre>\$ helm repo add synopsys https://sig-repo.synopsys.com/artifactory/sig-cloudnative</pre>

クラスタのサイジング 以下のクラスタのサイジング情報は参考用であり、実際のホスト型構成とは異なる場合があります。データは、デフォルトの最大アーティファクトサイズである5 GBのサポートに必要なサイジングの説明を目的として提供されたものです。

表 1：コア

	必要なコア数	合計 コア数
1マネージャ	マネージャあたり2	2
1メッセージハンドラ	メッセージハンドラあたり1	1
5スキャナとBDBAワーカー	スキャナおよびBDBAワーカーあたり1	5
		8

カテゴリ	要件																		
	表 2: メモリ																		
	<table><tr><th></th><th>必要なメモリ</th><th>合計メモリ</th></tr><tr><td>1 マネージャ</td><td>マネージャあたり 4 GB</td><td>4 GB</td></tr><tr><td>1 メッセージハンドラ</td><td>メッセージハンドラあたり 4 GB</td><td>4 GB</td></tr><tr><td>5 スキャナ</td><td>スキャナあたり 5 GB</td><td>25 GB</td></tr><tr><td>5 BDBA ワーカー</td><td>BDBA ワーカーあたり 5 GB</td><td>25 GB</td></tr><tr><td></td><td></td><td>58 GB</td></tr></table>		必要なメモリ	合計メモリ	1 マネージャ	マネージャあたり 4 GB	4 GB	1 メッセージハンドラ	メッセージハンドラあたり 4 GB	4 GB	5 スキャナ	スキャナあたり 5 GB	25 GB	5 BDBA ワーカー	BDBA ワーカーあたり 5 GB	25 GB			58 GB
	必要なメモリ	合計メモリ																	
1 マネージャ	マネージャあたり 4 GB	4 GB																	
1 メッセージハンドラ	メッセージハンドラあたり 4 GB	4 GB																	
5 スキャナ	スキャナあたり 5 GB	25 GB																	
5 BDBA ワーカー	BDBA ワーカーあたり 5 GB	25 GB																	
		58 GB																	
	ストレージ: ホストされた環境には、100 GB 以上のディスク容量が含まれます (スキャナレプリカを 5 に設定)。																		
ストレージクラス	Artifactory Integration の導入には、永続ボリュームをサポートする完全にプロビジョニングされたストレージクラスが必要です。																		
永続ボリューム	スキャナがスキャン用のアーティファクトをダウンロードできるように、Artifactory Integration の導入では、十分な物理ディスク容量が必要になります。ホストされたシステムには、100 GB 以上のディスク容量が提供されます (スキャナレプリカを 5 に設定)。																		
その他の要件	Artifactory Integration プラグイン <ul style="list-style-type: none">ターゲット JFrog Artifactory Pro Server にインストールされた Black Duck Artifactory Integration プラグイン。プラグインのインストールは、これまでと同様に 現在の手順 に従ってください。ScanAsAService モジュールに固有のプラグインについては、設定項目に注意してください。																		

インストールの順序

ここでは、Artifactory Integration のインストールについて、順番に手順を概説します。

1. Black Duck インスタンスからアクセストークンを取得し、安全な場所に保存します。
2. Artifactory インスタンスからアクセストークンを取得し、安全な場所に保存します。
3. Kubernetes 環境を準備します。
 - a. sig-cloudnative から最新の Artifactory Integration 導入チャートを取得します。
 - b. Kubernetes で導入用の名前空間を作成します。
 - c. Black Duck および Artifactory のアクセストークンのために、新規作成した名前空間にシークレットを作成します。
4. Artifactory Integration パラメータを使用して values.yaml を編集します。

プロパティ	詳細
BLACKDUCK_SCA_ENGINE_SCHEME	<p>説明: Artifactory Integration インスタンスへの接続に使用するプロトコル。</p> <p>デフォルト値: 該当なし</p> <p>必須: はい</p> <p>使用者:</p>

	・ scaaas-scanner
BLACKDUCK_SCA_ENGINE_HOST	説明: Artifactory Integrationのマネージャインスタンスの名前。 デフォルト値: 該当なし 必須: はい 使用者:
	・ scaaas-scanner
BLACKDUCK_SCA_ENGINE_PORT	説明: Artifactory Integrationのマネージャインスタンスが実行されているポート。 デフォルト値: 該当なし 必須: はい 使用者:
	・ scaaas-scanner

- 名前空間にArtifactory Integrationをインストールします。
- Artifactory Integrationプラグインのインストールのために、以下を準備します。
 - GitHubからプラグインをダウンロードします。
 - ダウンロードしたファイルを解凍します。
 - プラグインファイルをArtifactoryインストールの適切なディレクトリに移動します。
- Artifactory IntegrationプラグインのblackDuckPlugin.propertiesファイルを、必要に応じて編集します。
- Artifactoryサーバーを再起動します。

Artifactory Integrationのインストール

Helmを介したArtifactory Integrationの導入

- Artifactory Integrationを以下の場所からダウンロードします。
 - Black Duck 2023.4.x以前のユーザーの方は、最新の導入チャートをsig-cloudnativeリポジトリからプルダウンしてください。この操作により、導入アーカイブ(tar.gz)がダウンロードされます。このアーカイブは、今後の導入手順のために展開する必要があります。

最新のチャートをダウンロードするには:

```
$ helm repo update
$ helm pull synopsys/sca-as-a-service
```

チャートを展開してアクセスするには:

```
$ tar xvf sca-as-a-service-x.x.x.tgz
$ cd sca-as-a-service
```

- Black Duck 2023.7.x以降のユーザーの方は、Artifactory Integrationを次のリンクからダウンロードしてください: <https://sig-repo.synopsys.com/artifactory/bds-integrations-release/com/synopsys/integration/artifactory-integration/>
- 名前空間がまだ作成されていない場合は、名前空間を作成します。

```
$ BD_NAME="bd"
```

```
$ kubectl create ns ${BD_NAME}
```

3. ArtifactoryインスタンスとBlack Duckインスタンスの両方のために、アクセストークンのシークレットを作成します。


```
$ BD_NAME="bd"
$ kubectl create secret generic ${BD_NAME}-scaaas-secret-store -n ${BD_NAME} --from-literal=scaaas-artifactory-token=<artifactory_token> --from-literal=scaaas-blackduck-token=<blackduck_token>
```

Artifactoryアクセストークンスコープ: 最小権限(以下参照)が満たされている限り、「Admin」ではなく「User」にできます。

許可	リポジトリ
<ul style="list-style-type: none"> 読み取り 注釈 	Artifactory Integrationでスキャンする必要のあるローカルまたはリモートのキャッシュリポジトリ。
<ul style="list-style-type: none"> 導入/キャッシュ 	スキャンレポートのアップロード先である「ローカル」リポジトリのみ。

4. Artifactory Integrationの導入を設定します。
 - a. Artifactory Integrationのhelmチャートをインストールする前にvalues.yamlファイルを更新します。
 - b. 適切な値を以下の環境変数に設定します(詳細については、後述の「[Artifactory Integrationアプリケーションの設定](#)」セクションを参照)。

```
environs:
  BLACKDUCK_SCAAAS_BLACKDUCK_HOST: ""
  BLACKDUCK_SCAAAS_BLACKDUCK_SCHEME: ""
  BLACKDUCK_SCAAAS_STRUCTURED_LOGGING: ""
  BLACKDUCK_SCA_ENGINE_SCHEME:
  BLACKDUCK_SCA_ENGINE_HOST:
  BLACKDUCK_SCA_ENGINE_PORT:
```

 注: 前述したリストの中で、未設定の環境変数があった場合は、ハッシュタグ(#)を先頭に追加してコメントアウトするか、単に削除することをお勧めします。


- c. 前に作成したシークレットにアクセスできるように、「secrets」セクションが更新されていることを確認します。

```
secrets:
  artifactory:
    token:
      name: bd-scaaas-secret-store
      key: scaaas-artifactory-token
  basicAuth:
    user: {}
    password: {}
  blackduck:
    token:
      name: bd-scaaas-secret-store
      key: scaaas-blackduck-token
```

5. helmチャートを使用してArtifactory Integrationを導入します。

```
$ BD_NAME="bd" && SCAAAS_NAME="scaaas"
$ helm install ${SCAAAS_NAME} sca-as-a-sevice/ --namespace ${BD_NAME}
```

Artifactory プラグインのインストール(非推奨)

 **警告:** このバージョンのArtifactoryプラグインは、Black Duck 2023.7.0リリースで非推奨になりました。現行バージョンのArtifactory Integrationプラグインについては、「Artifactory Integrationプラグインのインストール」をご参照ください。

次の手順では、Black Duck Artifactoryプラグインのインストールプロセスと設定プロセスを概説します。

1. Artifactoryプラグインを以下の場所からダウンロードします。

<https://github.com/blackducksoftware/blackduck-artifactory/releases>

2. artifactory-integration-<version>.[zip | tgz]ファイルをダウンロードして解凍すると、プラグインプロパティファイルを設定する準備が整います。
3. blackDuckArtifactoryIntegration.propertiesファイルで認証情報に使用するBlack Duck APIトークンを取得します。
4. /plugins/libフォルダにあるblackDuckPlugin.propertiesファイルを使用して、Black Duck認証情報を設定します。
Artifactory Integration の構成についての詳細は[こちら](#)。
5. pluginsおよびlibフォルダをコピーします。\${JFrog_ARTIFACTORY_HOME}/etc/plugins/
コピーすると次のようになります。
\${ARTIFACTORY_HOME}/etc/plugins/lib/blackDuckPlugin.properties
6. 次のフォルダのユーザーを変更します。

```
• chown -R 1030:1030 <path-to-blackDuckPlugin.groovy>
```

```
• chown -R 1030:1030 <path-to-plugin-lib-directory>
```

7. Artifactoryサーバーを再起動します。

接続のテスト

Black Duckプラグインをインストールして設定したら、接続をテストして、プラグインが正しく動作していることを確認することをお勧めします。次のcurlコマンドを使用して、接続をテストします。

```
curl -X GET -u USERNAME:PASSWORD "http://ARTIFACTORY_SERVER/artifactory/api/plugins/execute/blackDuckTestConfig"
```

DockerとともにインストールされたArtifactory

Docker cpコマンドを実行して、解凍した場所から、プラグインファイルとlibフォルダを移動します
\${ARTIFACTORY_HOME}/etc/plugins/.

Artifactory Integrationプラグインのインストール

以下の手順では、Artifactory Integrationプラグインのインストールと構成に関するプロセスを説明します。

1. Artifactoryプラグインを以下の場所からダウンロードします。

<https://sig-repo.synopsys.com/artifactory/bds-integrations-release/com/synopsys/integration/artifactory-integration/>

2. artifactory-integration-<version>.[zip | tgz]ファイルをダウンロードして解凍すると、プラグインプロパティファイルを設定する準備が整います。
3. blackDuckArtifactoryIntegration.propertiesファイルで認証情報に使用するBlack Duck APIトークンを取得します。
4. artifactory-integration-<version>/libフォルダにあるblackDuckArtifactoryIntegration.propertiesファイルを使用し、Black Duck認証情報を構成します。
プラグインの構成についての詳細は[こちら](#)。
5. artifactory-integration-<version>/blackDuckArtifactoryIntegration.groovyファイルとartifactory-integration-<version>/libフォルダを、次へコピーします: \${ARTIFACTORY_HOME}/var/etc/plugins/
6. 次のフォルダのユーザーを変更します。

```
• chown -R 1030:1030 ${ARTIFACTORY_HOME}/var/etc/plugins/
  blackDuckArtifactoryIntegration.groovy
```

```
• chown -R 1030:1030 ${ARTIFACTORY_HOME}/var/etc/plugins/lib
```

7. Artifactoryサーバーを再起動します。


DockerとともにインストールされたArtifactory

Docker cpコマンドを実行し、プラグインgroovyファイルとlibフォルダを解凍した場所から\${ARTIFACTORY_HOME}/var/etc/plugins/へ移動します。

Artifactoryプラグインの構成(非推奨)

Black Duck Artifactory Integrationプラグインバージョン6.0.0以降では、すべてのプラグインとその構成プロパティがblackDuckPlugin.propertiesファイルに組み込まれています。

プラグインを機能させるには、blackDuckPlugin.propertiesファイルを変更する必要があります。このファイルは、任意のテキストエディタを使用して、プロパティファイルを手動で編集して設定します。デフォルト値を含めて、任意のプロパティを編集できます。したがって、6.0.0より前のプラグインバージョンを使用している場合は、完全な再インストールを強くお勧めします。

 注: Black Duck Artifactory Integrationプラグインバージョン6.0.0以降では、Hubを参照するすべてのプロパティが削除され、サポートされなくなりました。

ここでは、blackDuckPlugin.propertiesファイルの重要設定について概要を示します。

Black Duck接続認証情報

プロパティファイルで設定したBlack Duckへの接続が必要です。

少なくとも、プロパティファイルのBlackDuck認証情報の下に、blackduck.api.token=<BD API token>トークンとBlack Duck URLを追加する必要があります。


```
# BlackDuck credentials
blackduck.url=
blackduck.username=
blackduck.password=
blackduck.api.token=
```

アクセストークンを使用していて、Black Duckのプロキシを使用していない場合は、以上がプロパティファイルの[認証情報]セクションで必要となる情報のすべてになります。

Scan-as-a-Serviceの設定

最も重要なスキャナ設定は、リポジトリリストです。Scan-as-a-Serviceモジュールが確認応答するリポジトリの名前を定義します。これらの設定を調整しないと、定義したポリシーに違反するアイテムをブロックできなくなります。

```
blackduck.artifactory.scaaas.blocking.repos=ext-release-local,libs-release
blackduck.artifactory.scaaas.blocking.docker.repos=ext-docker-repo
```

 注：仮想リポジトリにはコンポーネントが含まれていないため、仮想リポジトリはサポートされていません。仮想リポジトリが参照するローカルリポジトリは、スキャン対象として設定する必要があります。

もう1つの重要な設定は、カットオフ日です。スキャンする必要のない古いアプリケーションがある場合は、この日付によりカットオフを定義します。この場合、この日付よりも古いアプリケーションは無視されます。lastUpdatedの時刻がこの値よりも前になっているアーティファクトは、ブロック戦略の値に関係なく、ブロック戦略の対象とはなりません。

```
blackduck.artifactory.scan.cutoff.date=2019-05-04T00:00:00.000
```

プロパティファイル

プラグインを機能させるには、次のプロパティファイルを変更する必要があります。

```
# suppress inspection "UnusedProperty" for whole file

# BlackDuck credentials
blackduck.url=
blackduck.username=
blackduck.password=
blackduck.api.token=
blackduck.timeout=120
blackduck.proxy.host=
blackduck.proxy.port=
blackduck.proxy.username=
blackduck.proxy.password=
blackduck.trust.cert=false

# General
# The date time pattern used by the artifactory to display the scan/inspection timestamp.
# blackduck.artifactory.scan.cutoff.date must comply to this pattern
blackduck.date.time.pattern=yyyy-MM-dd'T'HH:mm:ss.SSS
blackduck.date.time.zone=

# Scanner
blackduck.artifactory.scan.enabled=false
blackduck.artifactory.scan.repos=ext-release-local,libs-release
blackduck.artifactory.scan.repos.csv.path=
blackduck.artifactory.scan.name.patterns=*.jar,*.war,*.zip,*.tar.gz,*.hpi
blackduck.artifactory.scan.binaries.directory.path=
blackduck.artifactory.scan.memory=4096
blackduck.artifactory.scan.dry.run=false
blackduck.artifactory.scan.repo.path.codelocation=true
blackduck.artifactory.scan.repo.path.codelocation.include.hostname=true
blackduck.artifactory.scan.cutoff.date=2020-05-03T00:00:00.000
blackduck.artifactory.scan.cron=0 0/1 * 1/1 * ?

# If metadata.block.repos/metadata.block.repos.csv.path is left blank, all scanned repositories
are used

blackduck.artifactory.scan.metadata.block=false
blackduck.artifactory.scan.metadata.block.repos=
blackduck.artifactory.scan.metadata.block.repos.csv.path=

# If policy.repos/policy.repos.csv.path is left blank, all scanned repositories are used

blackduck.artifactory.scan.policy.block=true
blackduck.artifactory.scan.policy.repos=
blackduck.artifactory.scan.policy.repos.csv.path=
blackduck.artifactory.scan.policy.severity.types=BLOCKER,CRITICAL,MAJOR,MINOR,TRIVIAL,UNSPECIFIED

# Inspector
```

```

blackduck.artifactory.inspect.enabled=false
blackduck.artifactory.inspect.repos=jcenter-cache
blackduck.artifactory.inspect.repos.csv.path=
blackduck.artifactory.inspect.patterns.bower=*.tar.gz,*.tgz
blackduck.artifactory.inspect.patterns.cocoapods=*.tar.gz
blackduck.artifactory.inspect.patterns.composer=*.zip
blackduck.artifactory.inspect.patterns.conda=*.tar.bz2
blackduck.artifactory.inspect.patterns.cran=*.tar.gz,*.tgz,*.zip
blackduck.artifactory.inspect.patterns.rubygems=*.gem,*.gem.rz,*.gemspec.rz
blackduck.artifactory.inspect.patterns.maven=*.jar
blackduck.artifactory.inspect.patterns.go=*.mod,*.zip
blackduck.artifactory.inspect.patterns.gradle=*.jar
blackduck.artifactory.inspect.patterns.pypi=*.whl,*.tar.gz,*.zip,*.egg
blackduck.artifactory.inspect.patterns.nuget=*.nupkg
blackduck.artifactory.inspect.patterns.npm=*.tgz
blackduck.artifactory.inspect.cron=0 0/1 * 1/1 * ?
blackduck.artifactory.inspect.reinspect.cron=0 0 0 1/1 * ? *
blackduck.artifactory.inspect.retry.count=5
blackduck.artifactory.inspect.metadata.block=false

# If metadata.block.repos/metadata.block.repos.csv.path is left blank, all inspected repositories
# are used
blackduck.artifactory.inspect.metadata.block=false
blackduck.artifactory.inspect.metadata.block.policy.repos=
blackduck.artifactory.inspect.metadata.block.repos.csv.path=

# If policy.repos/policy.repos.csv.path is left blank, all inspected repositories are used
blackduck.artifactory.inspect.policy.block=true
blackduck.artifactory.inspect.policy.repos=
blackduck.artifactory.inspect.policy.repos.csv.path=
blackduck.artifactory.inspect.policy.severity.types=BLOCKER,CRITICAL,MAJOR,MINOR,TRIVIAL,UNSPECIFIED

# Scan-as-a-Service (scaaas)
# If Scan-as-a-Service is enabled, Scanner and Inspector *will* be disabled
blackduck.artifactory.scaaas.enabled=true
blackduck.artifactory.scaaas.blocking.strategy=BLOCK_NONE
blackduck.artifactory.scaaas.blocking.repos=ext-release-local,libs-release
blackduck.artifactory.scaaas.blocking.repos.csv.path=
blackduck.artifactory.scaaas.allowed.patterns=
blackduck.artifactory.scaaas.excluded.patterns=
# Download of items prior to this date will be ALLOWED regardless of the
# value of blackduck.artifactory.scaaas.blocking.strategy
# This date MUST comply with the format in blackduck.date.time.pattern
blackduck.artifactory.scaaas.cutoff.date=
# blocking.docker.repos and blocking.docker.repos.csv.path contain the list of repositories
# that are defined in Artifactory as Docker repositories. These MUST be specified explicitly
# and DO NOT have to be specified as part of blocking.repos or blocking.repos.csv.path.
# If empty, assumes NO repositories are of type Docker.
blackduck.artifactory.scaaas.blocking.docker.repos=ext-docker-repo
blackduck.artifactory.scaaas.blocking.docker.repos.csv.path=

# Analytics
blackduck.artifactory.analytics.enabled=true

```

Artifactory Integrationプラグインの構成

プラグインを機能させるには、blackDuckArtifactoryIntegration.propertiesファイルを変更する必要があります。このファイルは、任意のテキストエディタを使用して、プロパティファイルを手動で編集して設定します。

ここでは、blackDuckArtifactoryIntegration.propertiesファイルの重要設定について概要を示します。

Black Duck接続認証情報

プロパティファイルで設定したBlack Duckへの接続が必要です。

プロパティファイルのBlack Duck認証情報の下に、Black Duckトークンblackduck.api.token=<BD API token>とBlack Duck URLを追加する必要があります。

```

# BlackDuck credentials
blackduck.url=
blackduck.username=

```

5. Artifactory Integration · Artifactory Integrationタスク

```
blackduck.password=  
blackduck.api.token=
```

アクセストークンを使用していて、Black Duckのプロキシを使用していない場合は、以上がプロパティファイルの[認証情報]セクションで必要となる情報のすべてになります。

Artifactory構成名

Black DuckインスタンスのArtifactory Integration構成に指定された名前にマッチする構成名を設定する必要があります。Artifactory Integrationが初期化されると、Black Duckインスタンス(上記で構成)へ接続され、統合設定が以下の構成に基づいて取得されます。

```
blackduck.artifactory.config.name=
```

指定されたblackduck.artifactory.config.nameのBlack Duckインスタンスに構成が存在しない場合は、エラーのログが記録され、Artifactory IntegrationはArtifactoryインスタンスに読み込まれません。Artifactoryインスタンスの名前を変更し、再起動する必要があります。

接続のテスト

Black Duckプラグインをインストールして設定したら、接続をテストして、プラグインが正しく動作していることを確認することをお勧めします。以下のcurlコマンドで、接続をテストします。

```
curl -X GET -u USERNAME:PASSWORD http://ARTIFACTORY_SERVER/artifactory/api/plugins/execute/  
blackDuckTestConfig
```

Artifactory Integrationタスク

Artifactory Integrationのアップグレード

1. 新しいバージョンにアップグレードする前に、以下のコマンドを実行して、チャートミュージアムから最新バージョンのチャートを取得します。

```
$ helm repo update  
$ helm pull synopsys/sca-as-a-service
```

2. Artifactory Integrationをアップグレードします

```
$ helm upgrade ${SCAAAS_NAME} sca-as-a-service/ --namespace ${BD_NAME}
```

Artifactory Integrationの更新

更新は、特定のタイプのアップグレードであり、ENV変数の追加など、変更を同一バージョンに適用できます。更新時には、`--reuse-values`フラグを利用できます。

```
$ helm upgrade ${SCAAAS_NAME} sca-as-a-service/ --reuse-values --namespace ${BD_NAME}
```

Artifactory Integrationの再起動

Artifactory Integrationサービスの再起動には、次のオプションを使用できます。

1. ポッドを「0」に縮小してから、目的のレプリカにスケールバックします。

停止するには:

```
$ kubectl scale deployment <deployment-name> --replicas=0
```

開始するには:

```
$ kubectl scale deployment <deployment-name> --replicas=1
```

2. values.yamlでステータスを編集します。

ステータスを[実行中]から[停止]に変更し、「helm upgrade」を実行します。

```
$ helm upgrade ${SCAAAS_NAME} sca-as-a-service/ --reuse-values --set status="Stopped" --namespace ${BD_NAME}
```

ステータスを[停止]から[実行中]に変更し、「helm upgrade」を実行します。

```
helm upgrade ${SCAAAS_NAME} sca-as-a-service/ --reuse-values --set status="Stopped" --namespace ${BD_NAME}
```

Artifactory Integrationの削除

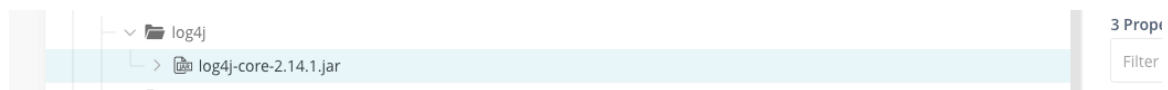
導入ファイルをアンインストール/削除するには:

```
$ helm delete ${SCAAAS_NAME} --namespace ${BD_NAME}
```


ブロックされたダウンロードの手動上書き

Artifactoryのアイテムが、BlackDuckの定義ポリシーに違反しているため、アイテムを上書きして、アイテムのダウンロードを可能にする場合は、次の手順に従ってください。

1. Artifactory UIにログインし、上書きする違反アイテムを見つけます。



2. 右側のペインから[プロパティ]を選択します。

 log4j-core-2.14.1.jar

General Effective Permissions **Properties** Followers Builds Xray

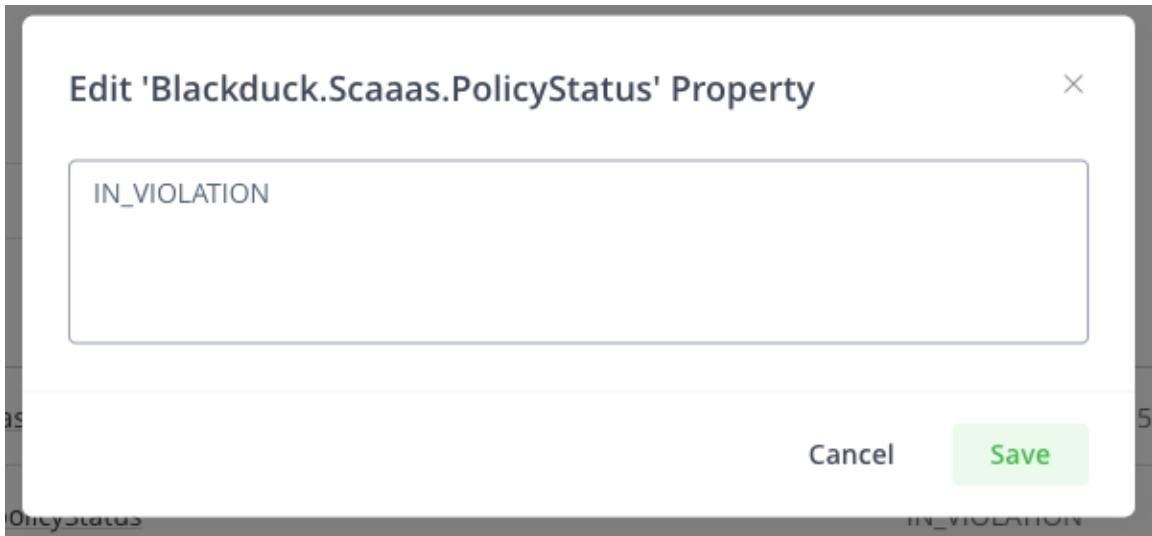
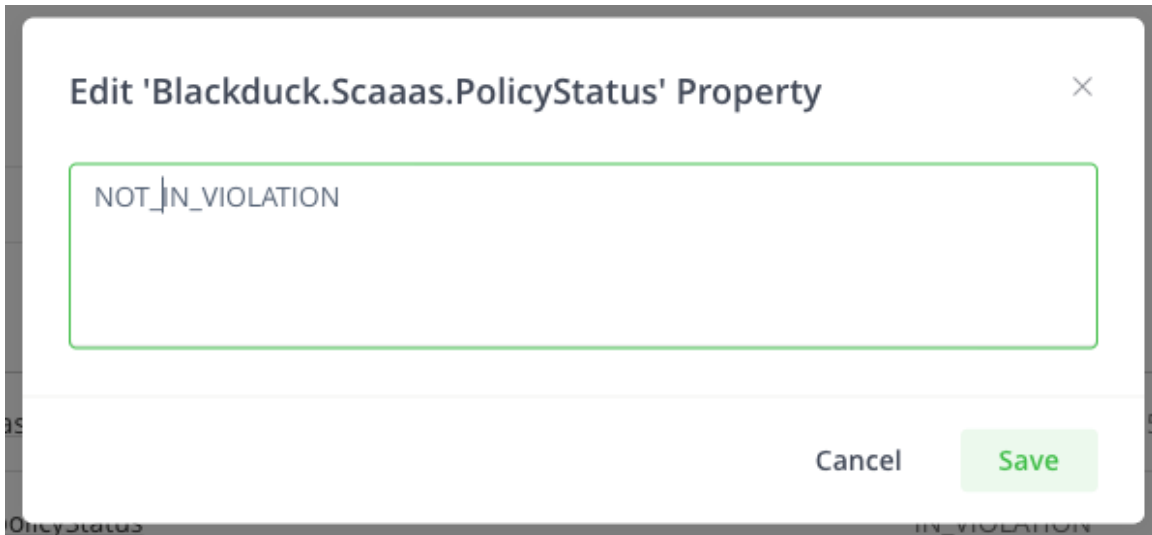
Add: **Property** Property Set

☐ Recursive [?](#)


3 Properties

<input type="radio"/> Property	Value(s)
<input type="radio"/> blackduck.scaaas.lastUpdate	2023-03-28T15:19:53.755Z
<input type="radio"/> blackduck.scaaas.policyStatus	IN_VIOLATION
<input type="radio"/> blackduck.scaaas.scanStatus	SUCCESS

3. `blackduck.scaaas.policyStatus`プロパティを選択し、その値を`NOT_IN_VIOLATION`に変更して[保存]をクリックします。

これで、Artifactory Integrationプラグインに設定されているブロック戦略に関係なく、アイテムをダウンロードできるようになります。

 **注：** アイテムが更新された場合（新しいバージョンがアップロードされた場合など）は、再スキャンされ、`policyStatus`が`IN_VIOLATION`に再度設定される可能性があります。ファイルを上書きして、ダウンロードできるようにするには、これらの手順を再度実行する必要があります。

バイナリスキャンとコンテナスキャンの無効化

ライセンスでバイナリおよびコンテナのスキャンが許可されていない場合は、`values.yaml`ファイルでBDDBAを無効にしてください。この操作を実行した場合に、`bdbaworker`コンテナがすでにロードされていた場合、コンテナのロードまたはアンロードは実行されません。署名スキャンのみがサポートされます。

バイナリスキャンとコンテナスキャンを無効にするには、`values.yaml`ファイルの「`bdbaworker`」セクションを編集し、次のように設定します。

```
enabled: false
```

ファイルを保存し、前述の「[Artifactory Integrationの更新](#)」セクションの手順に従い、導入環境に変更を適用します。

Artifactory Integrationの設定

次の表に、Artifactory Integrationチャートの設定可能パラメータとそのデフォルト値を示します。

Artifactory Integrationアプリケーションの設定

次の表に、Artifactory Integrationの設定アイテムを示します。

プロパティ名	詳細
BLACKDUCK_RABBIT_SSL	<p>説明: 高速通信にSSLを使用します。</p> <p>デフォルト値: false</p> <p>必須: いいえ</p> <p>使用者:</p> <ul style="list-style-type: none"> scaaas-manager scaaas-scanner rabbitmq
BLACKDUCK_SCAAAS_BLACKDUCK_HOST	<p>説明: Blackduckインスタスの名前。</p> <p>デフォルト値: 該当なし</p> <p>必須: はい</p> <p>使用者:</p> <ul style="list-style-type: none"> scaaas-scanner
BLACKDUCK_SCAAAS_BLACKDUCK_PORT	<p>説明: Black Duckインスタスが実行されているポート。</p> <p>デフォルト値: 該当なし</p> <p>必須: はい</p> <p>使用者:</p> <ul style="list-style-type: none"> scaaas-scanner
BLACKDUCK_SCAAAS_BLACKDUCK_SCHEME	<p>説明: Blackduckインスタスへの接続に使用するプロトコル。</p> <p>デフォルト値: 該当なし</p> <p>必須: はい</p> <p>使用者:</p> <ul style="list-style-type: none"> scaaas-scanner
BLACKDUCK_SCAAAS_BLACKDUCK_TOKEN	<p>説明: スキャン認証用のBlackduck APIトークン(開発導入専用)。</p> <p>デフォルト値: 該当なし</p> <p>必須: いいえ</p> <p>使用者:</p> <ul style="list-style-type: none"> scaaas-scanner
BLACKDUCK_SCAAAS_COMPRESS_ALL_MQ_MESSAGES	<p>説明: 有効にすると、すべてのrabbitMQメッセージが圧縮されます。</p> <p>デフォルト値: false</p> <p>必須: いいえ</p> <p>使用者:</p> <ul style="list-style-type: none"> scaaas-manager

プロパティ名	詳細
	<ul style="list-style-type: none"> scaaas-scanner
BLACKDUCK_SCAAAS_DETECT_BDBA_TIMEOUT	<p>説明: Detect BDBAスキャンタイムアウト(秒単位)。 デフォルト値: 3600 必須: いいえ 使用者:</p> <ul style="list-style-type: none"> scaaas-scanner
BLACKDUCK_SCAAAS_DETECT_TIMEOUT	<p>説明: Detectスキャンタイムアウト(秒単位)。 デフォルト値: 600 必須: いいえ 使用者:</p> <ul style="list-style-type: none"> scaaas-scanner
BLACKDUCK_SCAAAS_DOWNLOAD_TIMEOUT_MINUTES	<p>説明: スキャナのダウンロードが例外で終了し、このタイムアウト値よりも時間がかかった場合、そのアイテムはFAILEDとしてマークされます。 デフォルト値: 30分 必須: いいえ 使用者:</p> <ul style="list-style-type: none"> scaaas-scanner
BLACKDUCK_SCAAAS_STALL_ON_FAILURE	<p>説明: Javaをバックグラウンドで実行し、それを監視することで、Javaの起動に問題がある場合、クラッシュが発生した場合に、オプションで停止できるようになっています。このオプションを有効にしても、デバッグと分析のためにコンテナに接続することは可能です。 デフォルト値: false 必須: いいえ 使用者:</p> <ul style="list-style-type: none"> scaaas-manager scaaas-scanner
DETECT_LATEST_RELEASE_VERSION	<p>説明: スキャンに使用するDetectのバージョン。Artifactory IntegrationではDetect 8.2.0以上が必要です。 デフォルト値: <value as configured from synopsys-detect dependency> 必須: いいえ 使用者:</p> <ul style="list-style-type: none"> scaaas-scanner
RABBIT_MQ_PORT	<p>説明: rabbitMQ通信に使用するポート。 デフォルト値: 5672 必須: はい 使用者:</p> <ul style="list-style-type: none"> scaaas-manager scaaas-scanner rabbitmq


6. 管理タスク

Kubernetesでのシークレットの暗号化の構成

Black Duckは、システム内で重要なデータの、保存データの暗号化をサポートします。この暗号化は、オーケストレーション環境(Docker SwarmまたはKubernetes)によってBlack Duckインストールにプロビジョニングされたシークレットに基づいています。次に、組織のセキュリティポリシーに基づいてこのシークレットを作成および管理し、バックアップシークレットを作成し、シークレットをローテーションするプロセスの概要を示します。

暗号化される重要なデータは、次のとおりです。

- ・ SCM統合OAuthトークン
- ・ SCM統合プロバイダOAuthアプリケーションクライアントシークレット
- ・ LDAP認証情報
- ・ SAMLプライベート署名証明書

 注：シークレットの暗号化は、いったん有効にすると、無効にすることはできません。

暗号化シークレットの概要

暗号化シークレットは、システム内のリソースをアンロックする目的で内部暗号化キーを生成するために使用されるランダムなシーケンスです。Black Duckのシークレットの暗号化は、3つの対称キー、つまりルートキー、バックアップキーおよび以前のキーによって制御されます。これらの3つのキーは、KubernetesおよびDocker SwarmシークレットとしてBlack Duckに渡されたシードによって生成されます。3つのシークレットは、次のように名前が付けられます。

- ・ crypto-root-seed
- ・ crypto-backup-seed
- ・ crypto-prev-seed

通常の状態では、3つのシードはすべて、アクティブな使用中にはなりません。ローテーションアクションが進行中ではない限り、アクティブな唯一のシードはルートシードになります。

ルートシードのセキュリティ保護

ルートシードを保護することは重要です。システムデータのコピーとともにルートシードを所有するユーザーは、システムの保護されたコンテンツをアンロックし、読み取る可能性があります。一部のDocker SwarmシステムまたはKubernetesシステムは、デフォルトでは、保存時のシークレットを暗号化しません。これらのオーケストレーションシステムを内部で暗号化するように構成して、後でシステムに作成されるシークレットが安全に保たれるようにすることを強くお勧めします。

ルートシードは、災害復旧計画の一部としてバックアップからシステム状態を再作成するのに必要です。ルートシードファイルのコピーは、オーケストレーションシステムとは別の秘密の場所に保存して、シードとバックアップの組み合わせでシステムを再作成できるようにする必要があります。ルートシードをバックアップファイルと同じ場所に保存することはお勧めしません。一方のファイルセットが漏洩したり盗まれたりした場合、両方が漏洩したり盗まれたりしたことになります。したがって、バックアップデータ用とシードバックアップ用で別々の場所を用意することをお勧めします。

Kubernetesでのシークレットの暗号化の有効化

Kubernetesでシークレットの暗号化を有効にするには、values.yamlオーケストレーションファイルのenableApplicationLevelEncryptionの値をtrueに変更する必要があります。

```
# if true, enables application level encryption
enableApplicationLevelEncryption: true
```

キーシード管理スクリプト

サンプル管理スクリプトは、Black Duck GitHubパブリックリポジトリで確認できます。

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

このスクリプトは、Black Duckシークレットの暗号化を管理するためではなく、ここに文書化されている、低レベルのDockerおよびKubernetesコマンドの使用を示すためのものです。2つのスクリプトセットがあり、それぞれが専用のサブディレクトリにあります（Kubernetesプラットフォームでの使用、Docker Swarmプラットフォームでの使用に対応しています）。KubernetesおよびDocker Swarm用の個々のスクリプト間に1対1の対応があります（該当する場合）。たとえば、両方のスクリプトセットに次のスクリプトが含まれています。

createInitialSeeds.sh

Kubernetesでのシードの生成

OpenSSLでのシードの生成

シードの内容は、少なくとも1024バイト長の、セキュリティで保護されたランダムな内容を生成する任意のメカニズムを使用して生成できます。シードは、作成され、シークレットに保存されたら、すぐにファイルシステムから削除し、プライベートな、セキュリティで保護された場所に保存する必要があります。

OpenSSLコマンドは、次のとおりです。

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

Kubernetesでのシードの生成

シークレットを作成するKubernetesコマンドラインは多数あります。以下にリストされているコマンドにより、シークレットとその変更の有無をより適切に追跡でき、オンラインシステムでシークレットを操作できることとの互換性が保証されます。シークレットは、Black Duckがアクティブに実行されているKubernetesで作成および削除できます。

```
kubect1 create secret generic crypto-root-seed -n $NAMESPACE --save-config --dry-run=client --
from-file=crypto-root-seed=./root_seed -o yaml | kubect1 apply -f -
```

Kubernetesで前のキーシークレットを削除するには

```
kubect1 delete secret crypto-prev-seed -n $NAMESPACE
```

バックアップシードの構成

災害復旧シナリオでシステムを確実に復元できるように、バックアップルートシードを用意することをお勧めします。バックアップルートシードは、システムを復元するために配置できる代替ルートシードです。したがって、ルートシードと同じ方法で安全に保管する必要があります。

バックアップルートシードは、いったんシステムに関連付けられると、ルートシードのローテーションにわたって利用できるという特別な機能がいくつかあります。バックアップシードがシステムによって処理されたら、攻撃および漏洩へ

の暴露を限定するために、シークレットから削除する必要があります。バックアップルートシードは、シークレットがシステム内でどの時点でも「アクティブ」にならないようにするため、異なる(あまり頻繁ではない)ローテーションスケジュールを持つことができます。

ルートシードをローテーションする必要があるかローテーションしたい場合は、まず、現在のルートシードを前のルートシードとして定義する必要があります。その後、新しいルートシードを生成し、所定の場所に配置できます。

システムがこれらのシードを処理するとき、リソースをローテーションして新しいルートシードを使用するために、前のルートキーが使用されます。この処理の後、前のルートシードをシークレットから削除して、ローテーションを完了し、古いリソースをクリーンアップする必要があります。

バックアップルートシードの作成

バックアップシード/キーは、最初に作成されると、TDEK(テナントの復号化、暗号化キー)低レベルキーをラップします。サンプルスクリプト`createInitialSeeds.sh`は、ルートシードとバックアップシードの両方を作成します。Black Duckは、実行されると、両方のキーを使用してTDEKをラップします。

この操作が完了し、ルートシードとバックアップシードの両方が別の場所に安全に保存されたら、バックアップシードシークレットを削除する必要があります。[サンプルスクリプト](#) `cleanupBackupSeed.sh`を参照してください。

ルートキーが紛失または漏洩した場合、バックアップキーを使用してルートキーを置き換えることができます。[サンプルスクリプト](#) `useRootSeed.sh`を参照してください。

バックアップシードのローテーション

ルートキーと同様に、バックアップシードは定期的にローテーションする必要があります。ルートシード(古いルートシードが以前のシードシークレットとして保存され、新しいルートシードシークレットがシステムに提示されます)とは異なり、バックアップシードは新しいバックアップシードを作成するだけでローテーションされます。[サンプルスクリプト](#) `rotateBackupSeed.sh`を参照してください。

ローテーションが完了したら、新しいバックアップシードを安全に保存し、Black Duckホストファイルシステムから削除する必要があります。

Kubernetesでのシークレットローテーションの管理

組織のセキュリティポリシーに従って、使用中のルートシードを定期的にローテーションすることをお勧めします。これを行うには、ローテーションを実行するために追加のシークレットが必要です。ルートシードをローテーションするために、現在のルートシードが「前のルートシード」として構成され、新しく生成されるルートシードがルートシードとして生成および構成されます。システムがこの構成を処理すると(詳細は以下)、シークレットがローテーションされます。

その時点では、古いシードと新しいシードの両方が、システムの内容をアンロックできます。デフォルトでは、新しいルートシードが使用され、システムが意図したとおりに動作していることをテストおよび確認できます。すべてが確認されたら、「前のルートシード」を削除することで、ローテーションを完了します。

前のルートシードは、システムから削除されると、システムの内容のアンロックに使用できなくなるため、破棄してかまいません。これで、新しいルートシードが適切なルートシードになりました。このルートシードは、適切にバックアップおよびセキュリティ保護する必要があります。

ルートキーは、Black Duckのシークレットを実際に暗号化および復号化する、低レベルのTDEK(テナントの復号化、暗号化キー)をラップするために使用されます。定期的に、Black Duck管理者にとって都合が良く、ユーザー組織のルールに準拠しているタイミングで、ルートキーをローテーションする必要があります。

ルートキーをローテーションする手順としては、現在のルートシードの内容で以前のシードシークレットを作成します。その後、新しいルートシードが作成され、ルートシードシークレットに保存される必要があります。

Kubernetesでのシークレットローテーション

Kubernetesでは、Black Duckを実行しながら、3つの操作を行うことができます。KubernetesサンプルスクリプトrotateRootSeed.shは、ルートシードをprev_rootに抽出し、新しいルートシードを作成してから、前のシードとルートシードを再作成します。

ローテーションが完了したら、前のシードシークレットを削除する必要があります。[サンプルスクリプトcleanupPreviousSeed.sh](#)を参照してください。繰り返しになりますが、このクリーンアップは、実行中のKubernetes Black Duckインスタンスに対して実行できます。

ローテーションの状態は、ユーザーインターフェイスで、[管理者] > [システム情報] > [暗号]の順に移動し、暗号診断タブを表示することで追跡できます。

Blackduck Storageのカスタムボリュームの構成

ストレージコンテナは、ファイルベースのオブジェクトを保存するために、最大3個のボリュームを使用するように構成できます。さらにこの構成は、あるボリュームから別のボリュームにオブジェクトを移行するように設定できます。

複数のボリュームを使用する理由

デフォルトでは、ストレージコンテナは、単一のボリュームを使用してすべてのオブジェクトを格納します。このボリュームのサイズは、一般的なユーザーが保存オブジェクトに使用する容量に基づいています。使用状況はユーザーごとに異なるため、ボリュームで提供可能な容量よりも多くの空き容量が必要になる場合もあります。すべてのボリュームが拡張可能とは限りません。したがって、別の大きなボリュームを追加して、その新しいボリュームにデータを移行しなければならない場合もあります。

複数のボリュームが必要になるもう1つの理由は、ボリュームがリモートシステム(NASまたはSAN)でホストされており、そのリモートシステムが廃止される予定になった場合です。ホストする2番目のボリュームを新しいシステムで作成し、コンテンツをそこに移動する必要があります。

複数ボリュームの設定

Kubernetesでカスタムストレージプロバイダを設定するには、以下を含む上書きファイルを作成します。

```
storage:
  providers:
    - name: "file-1"
      enabled: true
      index: 1
      type: "file"
      preference: 20
      readonly: false
      migrationMode: "none"
      existingPersistentVolumeClaimName: ""
      pvc:
        size: "100Gi"
        storageClass: ""
        existingPersistentVolumeName: ""
      mountPath: "/opt/blackduck/hub/uploads"
    - name: "file-2"
      enabled: true
      index: 2
      type: "file"
      preference: 10
      readonly: false
      migrationMode: "none"
      existingPersistentVolumeClaimName: ""
      pvc:
        size: "200Gi"
        storageClass: ""
        existingPersistentVolumeName: ""
      mountPath: "/opt/blackduck/hub/uploads2"
    - name: "file-3"
      enabled: false
```

6. 管理タスク・Blackduck Storageのカスタムボリュームの構成

```
index: 3
type: "file"
preference: 30
readonly: false
migrationMode: "none"
existingPersistentVolumeClaimName: ""
pvc:
  size: "100Gi"
  storageClass: ""
  existingPersistentVolumeName: ""
mountPath: "/opt/blackduck/hub/uploads3"
```

上記の上書きファイルでは、プロバイダ1と2の両方が有効になっていますが、プロバイダ2の優先度が高くなっているため(優先度の数値は小さい)、すべての新しい内容はプロバイダ2に転送されます。

各プロバイダの使用可能な設定は次のとおりです。

設定	詳細
name	デフォルト: なし。 有効な値: すべて。 注記: これは、これらのプロバイダの管理に役立つ表示用ラベルです。
enabled	デフォルト: true (プロバイダ1の場合)、false (その他の場合)。 有効な値: true または false。 注記: プロバイダが有効かどうかを示します。
index	デフォルト: なし。 有効な値: 1、2、3。 注記: プロバイダ番号を示します。設定ファイル内の順序は重要ではありません。
type	デフォルト: file。 有効な値: file。 注記: "file" はサポートされる唯一のプロバイダタイプです。
preference	デフォルト: index の10倍。 有効な値: 0-999。 注記: プロバイダの優先度を設定します。優先度が最大 (優先度の数値は最小) のプロバイダには、新しいコンテンツが追加されます。 注: すべてのプロバイダには固有な優先度を指定する必要があります。2つのプロバイダが同じ値になることは許されません。
readonly	デフォルト: false。 有効な値: true または false。 注記: プロバイダが読み取り専用であることを示します。優先度が最大 (優先度の数値は最小) のプロバイダは読み取り専用にできません。読み取り専用にすると、システムが機能しなくなります。 読み取り専用プロバイダでは、データの追加やデータの削除によってストレージボリュームが変更されることはありません。ただし、データベース内のメタデータは、オブジェクトの削除やその他の変更を記録するように制御されます。

設定	詳細
migrationMode	デフォルト: none。 有効な値: none、drain、delete、duplicate。 注記: プロバイダの移行モードを設定します。このモードの詳細と使い方については、このドキュメントの移行セクションを参照してください。
existingPersistentVolumeClaimName	デフォルト: ""。 有効な値: 任意の有効なk8s識別子。 注記: このボリュームに対して、特定の永続ボリューム要求名を指定できます。
pvc.size	デフォルト: none。 有効な値: 任意の有効なサイズ。 注記: ボリュームに対して使用可能な容量を指定できます。
pvc.storageClass	デフォルト: ""。 有効な値: 任意の有効なk8s識別子。 注記: このボリュームに対して、特定のストレージクラスを指定できます。
pvc.existingPersistentVolumeName	デフォルト: ""。 有効な値: 任意の有効なk8s識別子。 注記: このボリュームに対して、特定の永続ボリューム名を指定できます。
mountPath	デフォルト: インデックス固有。注記を参照。 有効な値: /opt/blackduck/hub/uploads /opt/blackduck/hub/uploads2 /opt/blackduck/hub/uploads3 注記: 特定のプロバイダのためにマウントポイントを設定します。インデックス1のプロバイダには、次のマウントポイントを指定する必要があります /opt/blackduck/hub/uploads。インデックス2のプロバイダには、次のマウントポイントを指定する必要があります /opt/blackduck/hub/uploads2。インデックス3のプロバイダには、次にマウントポイントを指定する必要があります /opt/blackduck/hub/uploads3

ボリューム間の移行

複数のボリュームを設定した場合、1個以上のプロバイダボリュームから新しいプロバイダボリュームにコンテンツを移行できます。これは、優先度が最高(優先度の数値が最小)ではないプロバイダに対してのみ実行できます。この操作を実行するには、次のいずれかの移行モードでボリュームを設定します。設定後、移行を開始するために、Black Duckを再起動する必要があります。移行は、完了するまで、ジョブによりバックグラウンドで実行されます。

移行モード	詳細
none	目的: 移行が進行中でないことを示します。

移行モード	詳細
	注記: デフォルトの移行モード。
drain	<p>目的: このモードでは、設定されているプロバイダから、優先度が最大(優先度の数値は最小)のプロバイダにコンテンツが移動されます。コンテンツが移動されると、コンテンツはすぐにソースプロバイダから削除されます。</p> <p>注記: これは、ターゲットプロバイダに追加し、ソースから削除するという直接移動の操作です。</p>
delete	<p>目的: このモードでは、設定されているプロバイダから、優先度が最大(優先度の数値は最小)のプロバイダにコンテンツがコピーされます。コンテンツがコピーされると、ソースプロバイダでは削除対象のマークがコンテンツに付けられます。標準的な削除保留期間が適用されます。この期間が経過すると、コンテンツは削除されます。</p> <p>注記: この移動の場合、削除の保留期間中、ソースプロバイダのコンテンツは存続可能な状態で保持されており、バックアップからシステムをリカバリできるようになっています。デフォルトの削除保留期間は6時間です。</p>
duplicate	<p>目的: このモードでは、設定されているプロバイダから、優先度が最大(優先度の数値は最小)のプロバイダにコンテンツがコピーされます。コンテンツがコピーされても、メタデータを含めて、ソースのコンテンツは変更されません。</p> <p>注記: 複製移行の後、データベース内の全コンテンツとメタデータが保存された2つのボリュームが存在することになります。「複製とダンプ」プロセスで次の手順に進み、元のボリュームの構成を解除した場合、コンテンツファイルは削除されますが、メタデータはデータベース内に残されたままになります。この場合、不明なボリュームを参照すると、ブルーニングジョブで警告が生成されます(ジョブエラー)。このエラーを解決するには、次のプロパティを使用して、孤立したメタデータレコードのブルーニングを有効にします。</p> <p><code>storage.pruner.orphaned.data.pruning.enable=true</code></p>

jobrunnerスレッドプールの設定

Black Duckには、2つのジョブプールがあります。1つはスケジュールされたジョブを実行するプール(別名: 定期プール)、もう1つはAPIやユーザーによる操作など、特定のイベントで開始されるジョブを実行するプール(別名: オンデマンドプール)です。

各プールには、最大スレッドとプリフェッチの2つの設定があります。

[最大スレッド]は、jobrunnerコンテナが同時に実行できるジョブの最大数です。ほとんどのジョブはデータベースを使用しますが、最大32の接続を確立できるため、定期的な最大スレッドとオンデマンドの最大スレッドの合計数が32を超えないように注意してください。jobrunnerメモリは使用率がすぐに最大になるため、デフォルトのスレッド数は非常に小さい値に設定されています。

[プリフェッチ]は、データベースへの1回のアクセスで取得するjobrunnerコンテナあたりのジョブ数です。大きな値を設定すると効率的ですが、小さい値を設定すると、複数のjobrunner間で負荷がより均等に分散されます(一般的には、負荷分散もjobrunnerの設計目標ではありません)。

Kubernetesでは、次の上書きファイルを使用して、スレッドカウント設定を上書きできます。

```
jobrunner:
  maxPeriodicThreads: 2
  maxPeriodicPrefetch: 1
  maxOndemandThreads: 4
  maxOndemandPrefetch: 2
```