



KubernetesとOpenShiftを使用したBlack Duckのインストール

Black Duck 2024.7.0

Copyright ©2024 by Black Duck.

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・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、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。

30-10-2024

目次

まえがき.....	5
Black Duck documentation.....	5
カスタマサポート.....	6
Black Duck Software Integrityコミュニティ.....	6
トレーニング.....	6
Black Duck 包括性と多様性に関する声明.....	7
Black Duck セキュリティへの取り組み.....	7
1. KubernetesとOpenShiftを使用したBlack Duckのインストール.....	8
2. ハードウェア要件.....	9
3. PostgreSQLのバージョン.....	10
一般的な移行プロセス.....	10
4. Helmを使用したBlack Duckのインストール.....	11
5. Artifactory Integration.....	12
6. 基本的なワークフロー.....	14
7. Artifactory Integrationの前提条件.....	15
8. インストールの順序.....	16
9. Artifactory Integrationプラグインのインストール.....	17
Artifactory Integrationプラグインの構成.....	18
接続のテスト.....	20
10. Artifactory Integrationタスク.....	21
11. 管理タスク.....	23
Kubernetesでのシークレットの暗号化の構成.....	23
Kubernetesでのシードの生成.....	24
バックアップシードの構成.....	24
Kubernetesでのシークレットローテーションの管理.....	25
Blackduck Storageのカスタムボリュームの構成.....	26

目次

jobrunnerスレッドプールの設定.....	29
Readiness Probeの設定.....	30
HUB_MAX_MEMORY設定の構成.....	30
Helmを使用したOpenShift上での移行.....	30

まえがき

Black Duck documentation

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の環境に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、Black Duckカスタマー サポートに問い合わせてください。

Black Duckサポートには、複数の方法でお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、Black Duck 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。

Black Duck Software Integrityコミュニティ

Black Duck Software Integrityコミュニティは、カスタマー サポート、ソリューション、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、Black Duckからの最新の製品ニュースやアップデートをいつでもご覧いただけます。これは、当社製品やサービスをより有効に活用し、オープンソースの価値を組織内で最大限に高めることができます。
- ・ 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

Black Duck Software Integrityグループ (SIG) は、Black Duckの教育ニーズをすべて満たすワンストップ 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。

Black Duck Software Integrityグループ (SIG) のCustomer Education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>でご確認ください。また、Black Duckに関するヘルプについては、ヘルプメニューの[チュートリアル] () (Black DuckのUIに表示)を選択してください。

Black Duck 包括性と多様性に関する声明

Black Duck は、すべての従業員、お客様、パートナー様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。


Black Duck セキュリティへの取り組み

Black Duck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、Black Duck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 | Black Duck](#)

1. KubernetesとOpenShiftを使用したBlack Duckのインストール

KubernetesとOpenShift™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、Black Duckctlはサポートされなくなり、更新も行われなくなります。Black Duck の展開はHelmチャート経由でサポートされています。次の場所にあるドキュメントとHelmチャートのサンプルを参照してください：[%install dir%/kubernetes/blackduck/](#)

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](#)」を参照してください。

Black Duck データベース

 **危険**：Black Duck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。Black Duck 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、Black Duck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。


 **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、2024.7.0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。

独自の外部PostgreSQLインスタンスを実行する場合、Black Duckは、新規インストールに最新バージョンのPostgreSQL 16を使用することをお勧めします。

 注：Black Duck 2024.4.0では、テスト目的でのみPostgreSQL 16を外部データベースとして使用するための事前サポートが追加されました。Black Duck 2024.7.0以降では、PostgreSQL 16は本番環境での使用に完全対応しています。

 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。

一般的な移行プロセス

このガイドンスは、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。

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Helm 3.5.4をサポートしており、Kubernetesの最小バージョンは1.17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、[こちら](#)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の導入をブートストラップします。

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Black Duck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。

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。Black Duck 高速または両方として定義されているポリシーが、Artifactory Integrationに適用されます。

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。

さらに、Black Duckポリシーに違反しているアーティファクトのダウンロードをブロックする機能は、リポジトリごとに設定できます。




注：ArtifactoryでArtifactory Integrationプラグインを使用するには、まずプラグインをインストールして設定し、Black Duckを使用するためのプラグインのAPIキーを取得する必要があります。

6. 基本的なワークフロー

ArtifactoryでBlack Duckプラグインの使用を開始するには、次のワークフローを使用します。

1. 要件を満たしていることを確認します。
2. Black Duckインスタンスの統合でArtifactoryサーバーを設定します。
3. Black DuckインスタンスでArtifactory統合用のAPIトークンを作成し、クリップボードにコピーします。
4. Black DuckArtifactory IntegrationプラグインをJFrog Artifactoryにインストールします。
5. Artifactory Integrationプラグインで以下の設定を行います。
 - a. Black Duck 使用するサーバー インスタンス。
 - b. そのBlack DuckインスタンスのAPIトークン。
 - c. Artifactory Server構成の名前。
6. HA構成のArtifactoryインスタンスまたは各ノードを再起動します。
7. スキャン後、結果として得られるアーティファクトのプロパティを調べるか、リンク(設定されている場合)に従ってBlack Duckインスタンスの結果を表示します。

7. 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
その他の要件	<p>Artifactory Integrationプラグイン</p> <ul style="list-style-type: none"> ・ JFrog Artifactory Proバージョン7.43.x以降 <ul style="list-style-type: none"> ・ Black Duck ターゲットのJFrog Artifactory Pro ServerにインストールされたArtifactory Integrationプラグイン。 ・ Artifactory 7.43.x対応のJavaバージョン11。 ・ Black Duck インスタンス。Black Duckのサポート対象バージョンについては、Black Duckリリース互換性を参照してください。 ・ Black Duck プラグインがBlack DuckインスタンスにアクセスするためのAPIトークン。 ・ このプラグインには、Black Duckのグローバルコードスキャナ、プロジェクト作成者、グローバルプロジェクトビューアのユーザーロールが必要です。 <ul style="list-style-type: none"> ・ スーパー ユーザーのロールを使用できますが、プラグインには必須ではありません。 ・ グローバル コード スキャナおよびプロジェクト作成者のロールでは、Black Duckユーザー アカウントを使用してスキャンしたプロジェクトのみを表示できます。 ・ すべてのプロジェクトを表示するには、グローバル プロジェクト ビューアのロールが必要です。

8. 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 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。
 - a. GitHubからプラグインをダウンロードします。
 - b. ダウンロードしたファイルを解凍します。
 - c. 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。
3. 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。
4. Artifactoryサーバーを再起動します。

9. 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。

Artifactoryプラグインのダウンロードと解凍

Black DuckArtifactory Integrationプラグイン アーカイブ(.zipまたは.tgz)ディストリビューションを、Black DuckArtifactory[Black Duck 外部SIGレポジトリ](#)からダウンロードします。

ディストリビューションをダウンロードした後、アーカイブ ファイルを解凍します。次のファイル構造に注意してください。

```
artifactory-integration-<version number>/
--blackDuckArtifactoryIntegration.groovy
- lib/
- -- artifactory-integration-common-<version number>.jar
- -- blackDuckArtifactoryIntegration.properties
- -- synopsysArtifactoryVersion.txt
```

ディストリビューションのコンポーネントは以下のとおりです。

- blackDuckArtifactoryIntegration.groovy : プラグイン。
- lib: プラグインの依存関係を含むライブラリ フォルダ。
- blackDuckArtifactoryIntegration.properties: プラグインの設定ファイル。

Black Duck資格情報の取得と構成

blackDuckArtifactoryIntegration.propertiesファイルで認証情報として使用する、[Black Duck APIトークン](#)を取得します。

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。

最後のステップ

artifactory-integration-<version>/blackDuckArtifactoryIntegration.groovyファイルとartifactory-integration-<version>/libフォルダを、\${ARTIFACTORY_HOME}/var/etc/plugins/にコピーします。

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
```

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/に移動します。

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 接続資格情報

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.api.token=
```

アクセストークンを使用していて、Black Duckにプロキシを使用していない場合、これはプロパティ ファイルのCredentialsセクションで必要とされる唯一の情報になります。

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=
```

Black Duckインスタンスに指定されたblackduck.artifactory.config.nameの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。

一般的なプロパティ

Artifactoryで使用される日付と時刻のパターンは、スキャン/検査のタイムスタンプを表示するように設定できます。Artifactoryプラグインは、有効なJava 8 ZoneIdを受け入れます。詳細については、<https://docs.oracle.com/javase/8/docs/api/java/time/ZoneId.html>を参照してください。

```
# 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=
```

以下は短縮IDのリストです。

ID	値
EST	・ 05:00
HST	・ 10:00
MST	・ 07:00
ACT	Australia/Darwin
AET	Australia/Sydney

AGT	America/Argentina/Buenos_Aires
ART	Africa/Cairo
AST	America/Anchorage
BET	America/Sao_Paulo
BST	Asia/Dhaka
CAT	Africa/Harare
CNT	America/St_Johns
CST	America/Chicago
CTT	Asia/Shanghai
EAT	Africa/Addis_Ababa
ECT	Europe/Paris
IET	America/Indiana/Indianapolis
IST	Asia/Kolkata
JST	Asia/Tokyo
MIT	Pacific/Apia
NET	Asia/Yerevan
NST	Pacific/Auckland
PLT	Asia/Karachi
PNT	America/Phoenix
PRT	America/Puerto_Rico
PST	America/Los_Angeles
SST	Pacific/Guadalcanal
VST	Asia/Ho_Chi_Minh

クライアント側スキャン

将来的な構成では、アーティファクトのスキャンをクライアント側のリソースで処理し、ファイアウォールを超えてスキャンをBlack Duckに転送する必要性を回避できるようになります。マッチングとポリシー評価は、引き続きお客様のBlack Duckインスタンスで実行されます。ファイアウォール内の顧客ハードウェアにクライアント側スキャンの設定とエンドポイントが必要です。

```
blackduck.client.scan.url=
blackduck.client.concurrent.scans=
```

blackduck.client.scan.urlにURLが指定されていない場合、Artifactory Integrationはスキャン、マッピング、ポリシー評価のためにアーティファクトをBlack Duckに転送します。

接続のテスト

Black Duckでは、Black Duckプラグインをインストールして設定する際には、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推奨します。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
```

10. 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
```

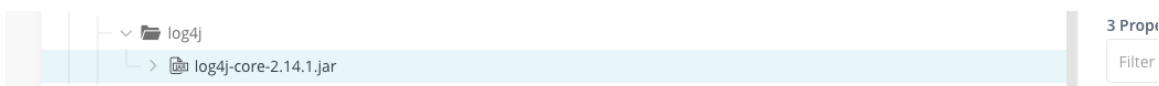
2. Artifactory Integrationをアップグレードします

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

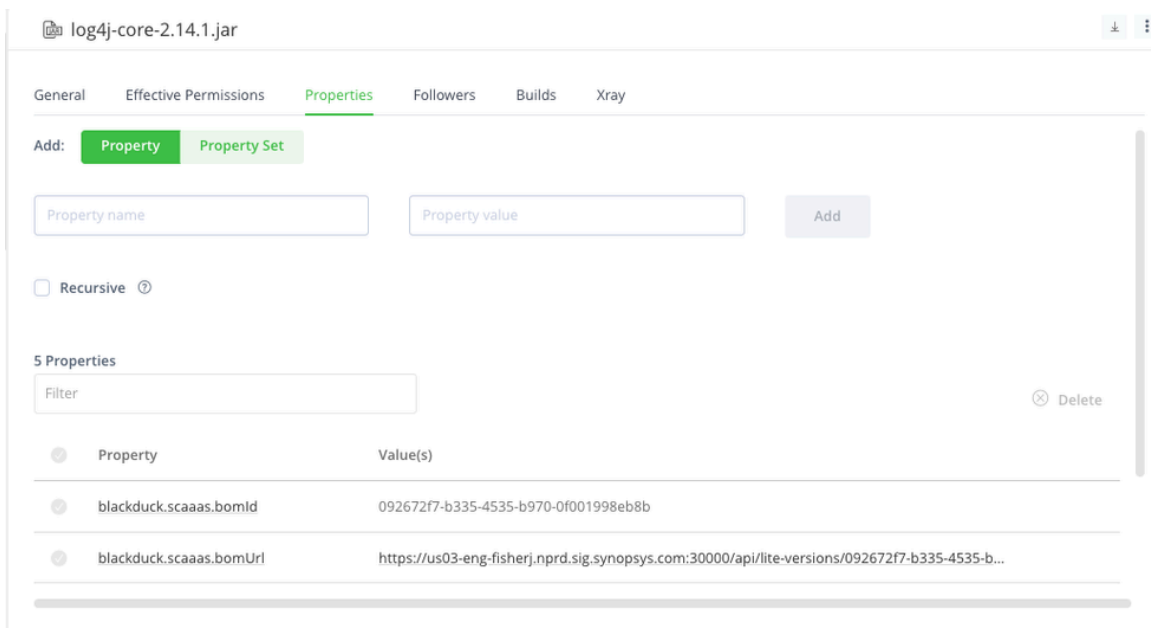
ブロックされたダウンロードの手動上書き

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. [プロパティ]を選択します。



3. [プロパティ名]テキスト フィールドに「blackduck.allowDownload」と入力します。

4. [プロパティ値]テキスト フィールドに「true」と入力します。

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：このプロパティの設定は、このアーティファクトにのみ影響し、このアーティファクトを含む他のアーティファクトには影響しません。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）、再スキャンされ、blackduck.allowDownloadプロパティが削除される場合があります。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BDBA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
```

11. 管理タスク

Kubernetesでのシークレットの暗号化の構成

Black Duck は、システム内の重要なデータの保存時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、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・削除できます。

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

Kubernetesで前のキーシークレットを削除するには

```
kubectrl 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必要があるので、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
      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(優先度の数値が低い)、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。構成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できません。
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(優先度の数値が最低)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。
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。

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

設定	詳細
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、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、利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、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、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、次のマウント 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。 注記: デフォルトの移行モード。
drain	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。

移行モード	詳細
	<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
```

Readiness Probeの設定

values.yamlの以下のブール値フラグを編集することにより、Readiness Probeを有効化あるいは無効化できます：

```
enableLivenessProbe: true
enableReadinessProbe: true
enableStartupProbe: true
```

HUB_MAX_MEMORY設定の構成

Kubernetesベースの展開では、関連するコンテナに対して構成パラメータHUB_MAX_MEMORYが自動で設定されます。この値は、メモリ制限のパーセンテージとして計算され、90%がデフォルト値です。

gen04展開のサイズ設定では、maxRamPercentageが使用される割合を制御します。この設定の値は、HUB_MAX_MEMORYが以前と同じ値になるように選択されています。

Helmを使用したOpenShift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Black Duck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。