SYNOPSYS®

使用 Docker Swarm 安装 Black Duck

Black Duck 2023.4.0

内容

前言 6
Black Duck 文档 6
客户支持 6
Synopsys Software Integrity 社区 7
培训 7
Synopsys 关于包容性和多样性的声明 7

1. 概述 8
Black Duck 架构 8
在 Black Duck 服务器上托管的组件 8

2. 安装规划 9
入门 9
新安装 9
从早期版本的 升级 Black Duck 9
硬件要求 9
Docker 要求 12
Docker 版本 12
在 RHEL 上安装 Docker 引擎 12
操作系统 13
软件要求 13
网络要求 13
数据库要求 14
PostgreSQL 版本 14
常规迁移过程 15
Swarm 上的迁移概述 15
代理服务器要求 15
配置 Nginx 服务器以与 配合使用 Black Duck 16
Amazon 服务 17
其他端口信息 17
配置 keepalive 设置 17
知识库反馈服务 18
用户代理分析 18
禁用反馈服务 18

3. 安装 Black Duck 19
安装文件 19
从 GitHub 页面下载 19
使用 wget 命令下载 20
分发 20
安装 Black Duck 21
中的快速扫描 Black Duck 22

4. 管理任务 24

环境文件和变量 26
重复 BOM 检测 26
更改 bearer 令牌的过期时间 26
关于 KBMATCH_SENDPATH 参数 26
通过代理服务器访问 API 文档 27
提供从非 Black Duck 服务器访问 REST API 的权限 27
增加二进制扫描文件的大小 28
配置仪表板刷新率 29
管理证书 29
使用自定义证书 30
从知识库获取组件迁移数据 31
启用迁移记录 31
保留迁移数据 31
API 端点 31
启用分层 BOM 32
在报告中包括忽略的组件 32
配置安全 LDAP 32
获取 LDAP 信息 32
导入服务器证书 33
LDAP 信任存储密码 34
下载日志文件和热图数据 35
查看日志文件 35
更改默认内存限制 36
更改默认 webapp 容器内存限制 36
更改默认的 jobrunner 容器内存限制 37
更改默认的扫描容器内存限制 38
更改默认的 binaryscanner 容器内存限制 38
更改默认的 bomengine 容器内存限制 39
更改 logstash 的主机名 39
清除未映射的代码位置 40
使用 cron 字符串计划扫描清除作业 40
清除卡住的 BOM 事件 40
使用覆盖文件 41
在 中配置分析 Black Duck 41
配置外部 PostgreSQL 实例 41
修改现有外部数据库的 PostgreSQL 用户名 45
配置代理设置 45
代理密码 46
导入代理证书 46
配置报告数据库密码 47
扩展 jobrunner、扫描、bomengine 和 binaryscanner 容器 47
扩展 bomengine 容器 47
扩展 jobrunner 容器 47
扩展扫描容器 48
扩展 binaryscanner 容器 48
为单点登录配置 SAML 48
上传源文件 50
备份密封密钥和原始主密钥 52
更换密封密钥 52
其他配置选项 52
启动或停止 Black Duck 52
启动 Black Duck 52
使用覆盖文件时启动 Black Duck 53
关闭 Black Duck 53

配置用户会话超时....54
向客户支持提供您的 Black Duck 系统信息....55
了解默认 sysadmin 用户....56
配置 Black Duck 报告延迟....56
配置容器的时区....56
修改默认用法....56
bdio2 上传的 jsonld/bdio 文件的匹配类型....57
自定义 Black Duck 容器的用户 ID....58
配置 Web 服务器设置....59
配置主机名....59
配置主机端口....59
禁用 IPv6....60
使用 UTF8 字符编码创建 BOM 报告....60
扫描监控....60
配置 HTML 报告下载大小....60
配置 KB 许可证更新和安全更新作业....60
在 Black Duck 中配置密钥加密....61
生成种子....62
配置备份种子....62
管理密钥轮换....63
为 Blackduck 存储配置自定义卷....63
配置 jobrunner 线程池....66
为 BOM 支持配置快速扫描....67

5. 卸载 Black Duck....68

6. 升级 Black Duck....69
安装文件....69
从 GitHub 页面下载....69
使用 wget 命令下载....69
用于清除审核事件表中未使用的行的迁移脚本....69
从现有 Docker 架构升级....70
备份和还原数据....72

7. Docker 容器....74
身份验证容器....75
Binaryscanner 容器....76
Bomengine 容器....77
CA 容器....77
DB 容器....78
文档容器....79
Jobrunner 容器....79
Logstash 容器....80
Matchengine 容器....81
Rabbitmq 容器....81
Redis 容器....82
注册容器....83
扫描容器....84
存储容器....84
Uploadcache 容器....85

Webapp 容器.....86
Webserver 容器.....87
WebUI 容器.....87

前言

Black Duck 文档

Black Duck 的文档包括在线帮助和以下文档：

标题	文件	说明
发行说明	release_notes.pdf	包含与当前版本和先前版本中的新功能和改进功能、已解决问题和已知问题有关的信息。
使用 Docker Swarm 安装 Black Duck	install_swarm.pdf	包含有关使用 Docker Swarm 安装和升级 Black Duck 的信息。
入门	getting_started.pdf	为初次使用的用户提供了有关使用 Black Duck 的信息。
扫描最佳做法	scanning_best_practices.pdf	提供扫描的最佳做法。
SDK 入门	getting_started_sdk.pdf	包含概述信息和样本使用案例。
报告数据库	report_db.pdf	包含有关使用报告数据库的信息。
用户指南	user_guide.pdf	包含有关使用 Black Duck 的 UI 的信息。

在 Kubernetes 或 OpenShift 环境中安装 Black Duck 软件的安装方法是 Synopsysctl 和 Helm。单击以下链接查看文档。

- Helm 是 Kubernetes 的软件包管理器，可用于安装 Black Duck。Black Duck 支持 Helm3，Kubernetes 的最低版本为 1.13。
- Synopsysctl 是一款云原生管理命令行工具，用于在 Kubernetes 和 Red Hat OpenShift 中部署 Black Duck 软件。

Black Duck 集成文档可在 Confluence 上找到。

客户支持

如果您在软件或文档方面遇到任何问题，请联系 Synopsys 客户支持。

您可以通过以下几种方式联系 Synopsys 支持：

- 在线：https://www.synopsys.com/software-integrity/support.html
- 电话：请参阅我们的支持页面底部的“联系我们”部分以查找您当地的电话号码。

要打开支持案例，请登录 Synopsys Software Integrity 社区网站，网址为：https://community.synopsys.com/s/contactsupport。

另一个可随时使用的方便资源是在线客户门户。

Synopsys Software Integrity 社区

Synopsys Software Integrity 社区是我们提供客户支持、解决方案和信息的主要在线资源。该社区允许用户快速轻松地打开支持案例，监控进度，了解重要产品信息，搜索知识库，以及从其他 Software Integrity Group (SIG) 客户那里获得见解。社区中包含的许多功能侧重于以下协作操作：

- 连接 - 打开支持案例并监控其进度，以及监控需要工程或产品管理部门协助的问题
- 学习 - 其他 SIG 产品用户的见解和最佳做法，使您能够从各种行业领先的公司那里汲取宝贵的经验教训。此外，客户中心还允许您轻松访问 Synopsys 的所有最新产品新闻和动态，帮助您更好地利用我们的产品和服务，最大限度地提高开源组件在您的组织中的价值。
- 解决方案 - 通过访问 SIG 专家和我们的知识库提供的丰富内容和产品知识，快速轻松地获得您正在寻求的答案。
- 分享 - 与 Software Integrity Group 员工和其他客户协作并进行沟通，以众包解决方案，并分享您对产品方向的想法。

访问客户成功社区。如果您没有帐户或在访问系统时遇到问题，请单击此处开始，或发送电子邮件至 community.manager@synopsys.com。

培训

Synopsys Software Integrity 的客户教育 (SIG Edu) 板块是满足您的所有 Black Duck 教育需求的一站式资源。它使您可以全天候访问在线培训课程和操作方法视频。

每月都会添加新视频和课程。

在 Synopsys Software Integrity 的客户教育 (SIG Edu) 板块，您可以：

- 按照自己的节奏学习。
- 按照您希望的频率回顾课程。
- 进行评估以测试您的技能。
- 打印完成证书以展示您的成就。

要了解更多信息，请访问 https://community.synopsys.com/s/education，或者，要获取 Black Duck 的帮助信息，请选择 Black Duck 教程（位于 Black Duck UI 的“帮助”菜单（ ）中）。

Synopsys 关于包容性和多样性的声明

Synopsys 致力于打造一个包容性的环境，让每位员工、客户和合作伙伴都感到宾至如归。我们正在审查并移除产品中的排他性语言以及支持面向客户的宣传材料。我们的举措还包括通过内部计划从我们的工程和工作环境中移除偏见语言（包括嵌入我们软件和 IP 中的术语）。同时，我们正在努力确保我们的 Web 内容和软件应用程序可供不同能力的人使用。由于我们的 IP 实施了行业标准规范，目前正在审查这些规范以移除排他性语言，因此您可能仍在我们的软件或文档中找到非包容性语言的示例。

1. 概述

本文档提供在 Docker 环境中安装 Black Duck 的说明。

Black Duck 架构

Black Duck 部署为一组 Docker 容器。“Docker 化”Black Duck，以使不同的组件容器化，从而允许 Swarm 等第三方编排工具管理所有单独的容器。

Docker 架构为 Black Duck 带来了这些重大改进：

- 提高性能
- 更轻松的安装和更新
- 可扩展性
- 产品组件编排和稳定性

有关组成 Black Duck 应用程序的 Docker 容器的更多信息，请参阅 Docker 容器。

有关 Docker 的更多信息，请访问 Docker 网站：https://www.docker.com/。

要获取 Docker 安装信息，请访问 https://docs.docker.com/engine/installation/。

在 Black Duck 服务器上托管的组件

以下远程 Black Duck 服务供 Black Duck 使用：

- 注册服务器：用于验证 Black Duck 的许可证。
- Black Duck 知识库服务器：Black Duck 知识库 (KB) 是业界最全面的开源项目、许可证和安全信息数据库。利用云中的 Black Duck KB，可确保 Black Duck 无需定期更新 Black Duck 安装即可显示有关开源软件 (OSS) 的最新信息。

2. 安装规划

本章介绍在安装 Black Duck 之前必须执行的安装前规划和配置。

入门

安装 Black Duck 的过程取决于您是第一次安装 Black Duck 还是从以前版本的 Black Duck 升级（基于 AppMgr 架构或基于 Docker 架构）。

新安装

对于新安装的 Black Duck：

1. 请阅读本规划章节以查看所有要求。
2. 确保满足所有要求后，请转到 安装 Black Duck 以获取安装说明。
3. 审查 管理任务。

从早期版本的 升级 Black Duck

1. 请阅读本规划章节以查看所有要求，
2. 确保满足所有要求后，请转到 升级 Black Duck 以获取升级说明。
3. 有关任何管理任务，请查看 管理任务。

硬件要求

以下性能数据是使用 Black Duck 2022.10.0（缩减了特征扫描持久性）（默认值）和 Synopsys Detect 8.0.0 收集的。SPH 值是使用特征扫描、软件包管理器探测器扫描和快速扫描的组合计算的。平均扫描大小因客户而异，因此确切的 SPH 吞吐量极度取决于客户。这些指标是从 Google Cloud Platform 收集的，该平台为不同配置提供不同的数据库读/写 IOPS。

10sph	每小时扫描数：50 SPH 百分比增加：400% 每小时 API 数：2,500 项目版本：10,000	IOPS: • 读取：15,000 • 写入：9,000 Black Duck 服务： • CPU：12 核 • 内存：30 GB PostgreSQL： • CPU：2 核 • 内存：8 GB	总数： • CPU：14 核 • 内存：38 GB
120sph	每小时扫描数：120 SPH 百分比增加：0% 每小时 API 数：3,000 项目版本：13,000	IOPS: • 读取：15,000 • 写入：15,000 Black Duck 服务：	总数： • CPU：17 核 • 内存：62 GB

		• CPU：13 核 • 内存：46 GB PostgreSQL： • CPU：4 核 • 内存：16 GB	
250sph	每小时扫描数：300 SPH 百分比增加：20% 每小时 API 数：7,500 项目版本：15,000	IOPS: • 读取：15,000 • 写入：15,000 Black Duck 服务： • CPU：17 核 • 内存：118 GB PostgreSQL： • CPU：6 核 • 内存：24 GB	总数： • CPU：23 核 • 内存：142 GB
500sph	每小时扫描数：650 SPH 百分比增加：30% 每小时 API 数：18,000 项目版本：18,000	IOPS: • 读取：15,000 • 写入：15,000 Black Duck 服务： • CPU：28 核 • 内存：210 GB PostgreSQL： • CPU：10 核 • 内存：40 GB	总数： • CPU：38 核 • 内存：250 GB
1000sph	每小时扫描数：1,400 SPH 百分比增加：40% 每小时 API 数：26,000 项目版本：25,000	IOPS: • 读取：25,000 • 写入：25,000 Black Duck 服务： • CPU：47 核 • 内存：411 GB PostgreSQL： • CPU：18 核 • 内存：72 GB	总数： • CPU：65 核 • 内存：483 GB
1500sph	每小时扫描数：1,600 SPH 百分比增加：6% 每小时 API 数：41,000 项目版本：28,000	IOPS: • 读取：25,000 • 写入：25,000 Black Duck 服务： • CPU：60 核 • 内存：597 GB PostgreSQL：	总数： • CPU：92 核 • 内存：701 GB

		• CPU：26 核 • 内存：104 GB	
2000sph	每小时扫描数：2,300 SPH 百分比增加：15% 每小时 API 数：50,000 项目版本：35,000	IOPS: • 读取：60,000 • 写入：25,000 Black Duck 服务： • CPU：66 核 • 内存：597 GB PostgreSQL： • CPU：34 核 • 内存：136 GB	总数： • CPU：100 核 • 内存：733 GB

这个新指南基于当前的 Black Duck 2022.10.0 架构。此指南可能会针对后续版本进一步完善。如果您有任何疑问或疑虑，请联系产品管理部门。

注: 所需的磁盘空间量取决于要管理的项目的数量，因此各个要求可能有所不同。考虑每个项目大约需要 200 MB。

Black Duck Software 建议监视 Black Duck 服务器上的磁盘利用率，以防止磁盘达到可能导致 Black Duck 出现问题的容量。

调整 binaryscanner 副本的数量，以及根据每小时将执行的二进制扫描的预期数量增加 PostgreSQL 资源，从而完成 BDBA 扩展。对于每小时每 15 次二进制扫描，添加以下资源：

- 一个 binaryscanner 副本
- 一个用于 PostgreSQL 的 CPU
- 用于 PostgreSQL 的 4GB 内存

如果您的预期扫描速率不是 15 的倍数，则向上舍入。例如，每小时 24 次二进制扫描将需要以下资源：

- 两个 binaryscanner 副本、
- 两个用于 PostgreSQL 的额外 CPU，以及
- 用于 PostgreSQL 的 8GB 额外内存。

当二进制扫描为总扫描量（按扫描计数）的 20% 或更少时，此指南有效。

二进制扫描

如果您获得了二进制扫描的许可，则可能需要增加 uploadcache 容器/Pod 内存，因为这是二进制扫描程序提取和处理二进制文件的位置。默认情况下，内存设置为 512MB，这不足以进行大型扫描。扫描大二进制文件时，建议将 uploadcache 容器/Pod 的内存至少增加到 4 GB。为此，请查找覆盖 yaml 并将内存限制更新为 4096MB。

对于 Swarm 安装：

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

```
    replicas: 1
```

对于 Kubernetes 安装：

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

注: 安装 Black Duck Alert 需要 1 GB 额外内存。

Docker 要求

Docker Swarm 是首选的 Black Duck 安装方法，它是 Docker 容器的群集和计划工具。通过使用 Docker Swarm，您可以将 Docker 节点群集作为单个虚拟系统进行管理。

注: 为了实现可扩展性，Black Duck Software 建议在单节点 Swarm 部署上运行 Black Duck。有关完整可扩展性调整原则，请参见《Black Duck 发行说明》的“容器可扩展性”一节。

在 Docker Swarm 中使用 Black Duck 时有以下限制：

警告: 不要在 PostgreSQL 数据目录上运行防病毒扫描。防病毒软件会打开大量文件，锁定文件，等等。这些操作会干扰 PostgresSQL 的运行。具体错误因产品而异，但通常会导致 PostgresSQL 无法访问其数据文件。一个例子是，PostgresSQL 失败，并显示错误“系统中打开的文件太多”。

- PostgreSQL 数据库必须始终在群集中的同一节点上运行，以便数据不会丢失（blackduck-database 服务）。

  这不适用于使用外部 PostgreSQL 实例的安装。

- blackduck-webapp 服务和 blackduck-logstash 服务必须在同一主机上运行。

  这是必须的，以便 blackduck-webapp 服务可以访问需要下载的日志。

- blackduck-registration 服务必须始终在群集中的同一节点上运行，或者由 NFS 卷或类似系统提供支持，以便注册数据不会丢失。

  它不必与用于 blackduck-database 服务或 blackduck-webapp 服务的节点相同。

- blackduck-upload-cache 服务必须始终在群集中的同一节点上运行，或者由 NFS 卷或类似系统提供支持，以便数据不会丢失。

  它不需要与其他服务使用的节点相同。

Docker 版本

Black Duck 安装支持 Docker 版本 18.09.x、19.03.x 和 20.10.x（CE 或 EE）。

在 RHEL 上安装 Docker 引擎

请注意，Docker 目前仅在 s390x (IBM Z) 上提供适用于 RHEL 的软件包。RHEL 尚不支持其他架构，但您可以在 RHEL 上安装 CentOS 软件包。有关详细信息，请参阅在 CentOS 上安装 Docker 引擎页面。

操作系统

用于在 Docker 环境中安装 Black Duck 的首选操作系统是：

- CentOS 7.3
- Red Hat Enterprise Linux 服务器 7.3
- Ubuntu 18.04.x
- SUSE Linux Enterprise 服务器版本 12.x（64 位）
- Oracle Enterprise Linux 7.3

此外，Black Duck 还支持其他 Linux 操作系统（支持受支持的 Docker 版本）。

注：Docker CE 不支持 Red Hat Enterprise Linux、Oracle Linux 或 SUSE Linux Enterprise Server (SLES)。单击此处了解更多信息。

目前不支持 Windows 操作系统。

软件要求

Black Duck 是具有 HTML 界面的 Web 应用程序。您通过 Web 浏览器访问应用程序。以下 Web 浏览器版本已经过测试，可与 Black Duck 配合使用：

- Safari 版本 15.4（16613.1.17.1.13，16613）
  - 不再支持 Safari 13.0 和更低版本
- Chrome 版本 100.0.4896.75（正式版本）(x86_64)
  - 不再支持 Chrome 版本 71 和更低版本
- Firefox 版本 99.0（64 位）
  - 不再支持 Firefox 版本 71 和更低版本
- Microsoft Edge 版本 100.0.1185.36（正式版）（64 位）
  - 不再支持 Microsoft Edge 78 和更低版本

请注意，Black Duck 不支持兼容模式。

注：这些浏览器版本是 Black Duck Software 已在其上测试过 Black Duck 的当前发布版本。更新的浏览器版本可能在 Black Duck 发布后可用，并且可能会、也可能不会按预期运行。较旧的浏览器版本可能按预期运行，但尚未经过测试，可能不受支持。

网络要求

Black Duck 需要从外部访问以下端口：

- 端口 443 - Web 服务器 HTTPS 端口（用于 Black Duck，通过 NGiNX）
- 端口 55436 - 来自 PostgreSQL 的只读数据库端口，用于报告

如果您的公司安全策略要求注册特定的 URL，则从 Black Duck 安装到 Black Duck Software 托管服务器的连接仅限于通过端口 443 上的 HTTPS/TCP 与以下服务器进行通信：

- updates.suite.blackducksoftware.com（用于注册您的软件）
- kb.blackducksoftware.com（访问 Black Duck KB 数据）

- https://auth.docker.io/token?scope=repository/blackducksoftware/blackduckregistration/pull&service=registry.docker.io（访问 Docker 注册表）

注：如果使用的是网络代理，则必须在代理配置中将这些 URL 配置为目标。

确保以下地址在允许列表中：

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

要验证连接，请使用 cURL 命令，如以下示例所示。

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

提示：可以检查 Docker 主机上的连接，但最好从 Docker 网络中验证连接。

数据库要求

Black Duck 使用 PostgreSQL 对象关系数据库存储数据。

警告：请勿删除 Black Duck 数据库 (bds_hub) 中的数据，除非 Synopsys 技术支持代表指示这样做。确保遵循适当的备份程序。删除数据会导致 UI 问题、Black Duck 完全无法启动等问题。Synopsys 技术支持无法重新创建已删除的数据。如果没有可用的备份，Synopsys 将尽力提供支持。

在安装 Black Duck 之前，确定是要使用自动安装的数据库容器还是外部 PostgreSQL 实例。

重要：从 Black Duck 2022.7.0 开始，Synopsys 建议将 PosgtreSQL 14 用于使用外部 PostgreSQL 的新安装。对于外部 PostgreSQL 实例，不再支持 PostgreSQL 11.x。对于内部 PostgreSQL 容器的用户，PostgreSQL 13 仍然是支持的版本。

对于外部 PostgreSQL 实例，Black Duck 支持：

- PostgreSQL 13.x、14.x（通过 Amazon Relational Database Service (RDS)）
- PostgreSQL 13.x、14.x（通过 Google Cloud SQL）
- PostgreSQL 13.x、14.x (Community Edition)
- PostgreSQL 13.x、14.x（通过 Microsoft Azure）

注：不支持 PostgreSQL 10.x 或 PostgreSQL 12.x。

有关详细信息，请参阅配置外部 PostgreSQL 实例。

注：有关 PostgreSQL 调整指南，请参阅 Black Duck 硬件扩展指南。

PostgreSQL 版本

Black Duck 2022.10.0 支持新的 PostgreSQL 特性和功能，以提高 Black Duck 服务的性能和可靠性。从 Black Duck 2022.10.0 开始，PostgreSQL 容器 13 是内部 PostgreSQL 容器当前支持的 PostgreSQL 版本。

从较旧版本的 Black Duck（早于 2022.10.0）升级的客户将需要迁移到 PostgreSQL 13。Black Duck 2022.10.0 更新将内部 Black Duck PostgreSQL 数据库容器迁移到 PostgreSQL 版本 13。如果您使用数据库容器并在 OpenShift 上部署，则需要运行一次性迁移作业，如 Black Duck 发行说明和安装指南中所述。

注：有关 PostgreSQL 调整指南，请参阅 Black Duck 硬件扩展指南。

如果您选择运行自己的外部 PostgreSQL 实例，Synopsys 建议为新安装使用 PostgreSQL 14。由于 PostgreSQL 14.0 到 14.3 中的索引损坏错误，支持的最低 PostgreSQL 14 版本是 14.4。

注：Black Duck 2023.4.0 增加了对使用 PostgreSQL 15 作为外部数据库的初步支持。此支持仅用于测试；不支持生产用途。

警告：不要在 PostgreSQL 数据目录上运行防病毒扫描。防病毒软件会打开大量文件，锁定文件，等等。这些操作会干扰 PostgreSQL 的运行。具体错误因产品而异，但通常会导致 PostgreSQL 无法访问其数据文件。一个例子是，PostgreSQL 失败，并显示“系统中打开的文件太多”。

常规迁移过程

此处的指南适用于从任何基于 PG 9.6 的 Hub（早于 2022.2.0 的版本）升级到 2022.10.0 或更高版本。

1. 迁移由 blackduck-postgres-upgrader 容器执行。
2. 如果从基于 PostgreSQL 9.6 的 Black Duck 版本升级：
   - PostgreSQL 数据卷的文件夹布局经过重新排列，使未来的 PostgreSQL 版本升级更加简单。
   - 数据卷所有者的 UID 已更改。新的默认 UID 为 1001，但请参见特定于部署的说明。
3. 运行 pg_upgrade 脚本以将数据库迁移到 PostgreSQL 13。
4. 在 PostgreSQL 13 数据库上运行普通“分析”以初始化查询计划程序统计信息。
5. blackduck-postgres-upgrader 退出。

Swarm 上的迁移概述

- 迁移完全是自动进行的；除了标准的 Black Duck 升级之外，不需要额外的操作。
- blackduck-postgres-upgrader 容器必须以 root 身份运行才能更改上述布局和 UID。
- 随后 Black Duck 重新启动时，blackduck-postgres-upgrader 将确定不需要迁移，并立即退出。
- 可选：成功迁移后，blackduck-postgres-upgrader 容器不再需要以 root 身份运行。

如果从 Black Duck 4.2.0 版本之前的版本升级，请参阅第 6 章《升级 Black Duck》，了解数据库迁移说明。

代理服务器要求

Black Duck 支持：

- 无身份验证
- Digest
- 基本
- NTLM

如果要向 Black Duck 发出代理请求，请与代理服务器管理员合作，以获取以下必要信息：

- 代理服务器主机使用的协议（http 或 https）。

- 代理服务器主机的名称
- 代理服务器主机正在侦听的端口。

配置 Nginx 服务器以与 配合使用 Black Duck

如果 Black Duck 前面有一台 NGINX 服务器充当 HTTPS 服务器/代理，则必须修改 NGINX 配置文件，以便 NGINX 服务器将正确的标头传递给 Black Duck。 Black Duck 然后生成使用 HTTPS 的 URL。

注: NGINX 服务器上只有一个服务可以使用 https 端口 443。

要将正确的标头传递给 Black Duck，请将 nginx.config 配置文件中的 location 块编辑为：

```
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;
}
```

如果正在代理服务器/负载平衡器配置中指定 X-Forwarded-Prefix 标头，请在 nginx.conf 配置文件中编辑 location 块：

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

要成功扫描文件，您必须在使用命令行时使用 context 参数，或将其包含在 Black Duck 扫描程序的 Black Duck 服务器 URL 字段中。

注: 尽管这些说明适用于 NGINX 服务器，但需要对任何类型的代理服务器进行类似的配置更改。

如果代理服务器将请求重新写入 Black Duck，请让代理服务器管理员知道以下 HTTP 标头可用于保留原始请求主机详细信息。

HTTP 标头	说明
X-Forwarded-Host	跟踪为发出请求而重写或路由的主机列表。原始主机是逗号分隔列表中的第一个主机。 示例： `X-Forwarded-Host: "10.20.30.40,my.example, 10.1.20.20"`
X-Forwarded-Port	包含一个值，表示用于原始请求的端口。 示例： `X-Forwarded-Port: "9876"`
X-Forwarded-Proto	包含单个值，表示用于原始请求的协议方案。 示例： `X-Forwarded-Proto: "https"`
X-Forwarded-Prefix	包含用于原始请求的前缀路径。 示例： `X-Forwarded-Prefix: "prefixPath"` 要成功扫描文件，必须使用 context 参数

Amazon 服务

您可以：

- 在 Amazon Web Services (AWS) 上安装 Black Duck

  有关 AWS 的更多信息，请参阅 AWS 文档和 AMI 文档。
- 为 Black Duck 使用的 PostgreSQL 数据库使用 Amazon Relational Database Service (RDS)。

  有关 Amazon RDS 的更多信息，请参阅您的 Amazon Relational Database Service 文档。

  重要：Synopsys 建议您使用 PostgreSQL 版本 13（外部 PostgreSQL 数据库）。
  外部 PostgreSQL 实例现在支持仅由数字字符组成的用户名。

其他端口信息

防火墙规则或 Docker 配置不能阻止以下端口列表。说明了可能如何阻止这些端口的示例包括：

- 主机上的 iptable 配置。
- 主机上的 firewalld 配置。
- 网络中另一台路由器/服务器上的外部防火墙配置。
- 在 Docker 默认创建的规则和 Black Duck 默认创建的规则之上应用的特殊 Docker 网络规则。

必须保持“未阻止”状态的端口的完整列表为：

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

配置 keepalive 设置

net.ipv4.tcp_keepalive_time 参数控制应用程序允许打开的 TCP 连接保持空闲的时间长度。默认情况下，此值为 7200 秒（2 小时）。

为获得最佳 Black Duck 性能，此参数的值应介于 600 和 800 秒之间。

可以在 Black Duck 安装之前或之后配置此设置。

要编辑该值：

1. 编辑 /etc/sysctl.conf 文件。例如：
   ```
   vi /etc/sysctl.conf
   ```
   您也可以使用 sysctl 命令修改此文件。
2. 添加 net.ipv4.tcp_keepalive_time（如果参数不在文件中）或编辑现有值（如果参数在文件中）。
   ```
   net.ipv4.tcp_keepalive_time = <value>
   ```
3. 保存并退出文件。
4. 输入以下命令以加载新设置：
   ```
   sysctl -p
   ```
5. 如果 Black Duck 已安装，请重新启动。

知识库反馈服务

知识库反馈用于增强 Black Duck 知识库 (KB) 功能。

- 当您对 KB 生成的匹配的组件、版本、来源、来源 ID 或许可证进行 BOM 调整时，将发送反馈。
- 如果您识别出与组件不匹配的文件，也会发送反馈；对于没有关联文件的手动添加的组件，不会发送反馈。
- 反馈用于提高未来匹配的准确性。此信息还有助于 Synopsys 确定资源的优先级，以便更详细地检查对客户至关重要的组件。

重要：不会向 KB 传输客户身份信息。

用户代理分析

知识库使用原始用户代理分析来提高知识库服务的可扩展性并提高用户的服务质量。

附加标头信息增加了发送到知识库的外发 HTTP 请求的标头大小。某些中间出口代理（由客户管理）可能需要重新配置以支持额外的标头大小，但这种情况不太可能出现。

禁用反馈服务

默认情况下，知识库反馈服务处于启用状态：您对 BOM 所做的调整将发送到知识库。

- 您可以使用 BLACKDUCK_KBFEEDBACK_ENABLED 环境变量覆盖反馈服务。值 false 会覆盖反馈服务，BOM 调整不会发送到知识库。
- 要禁用反馈服务，请将 BLACKDUCK_KBFEEDBACK_ENABLED=false 添加到 blackduck-config.env file 中。

注：将值设置为 true 可重新启用反馈服务。

3. 安装 Black Duck

安装 Black Duck 之前，请确保满足以下要求：

Black Duck 安装要求	
硬件要求	
√	您已确保您的硬件满足最低硬件要求。
Docker 要求	
√	您已确保您的系统满足 Docker 要求。
软件要求	
√	您已确保您的系统和潜在客户端满足软件要求。
网络要求	
√	您已确保您的网络满足网络要求。具体而言： • 端口 443 和端口 55436 可从外部访问。 • 服务器可以访问用于验证 Black Duck 许可证的 updates.suite.blackducksoftware.com。
数据库要求	
√	您已选择数据库配置。 具体来说，如果使用外部 PostgreSQL 实例，则您已配置数据库设置。
代理要求	
√	您已确保您的网络满足代理要求。 在安装 Black Duck 之前或之后配置代理设置。
Web 服务器要求	
√	在安装 Black Duck 之前或之后配置 Web 服务器设置。

安装文件

GitHub 上提供了安装文件。

下载编排文件。作为安装/升级过程的一部分，这些编排文件将提取必要的 Docker 映像。

请注意，尽管 tar.gz 文件的文件名因访问文件的方式而异，但内容是相同的。

从 GitHub 页面下载

1. 选择从 GitHub 页面下载 .tar.gz 文件的链接：https://github.com/blackducksoftware/hub。
2. 解压缩 Black Duck .gz 文件：
   ```
   gunzip hub-2023.4.0.tar.gz
   ```
3. 解压缩 Black Duck.tar 文件：
   ```
   tar xvf hub-2023.4.0.tar
   ```

使用 wget 命令下载

1. 运行以下命令：

   ```
   wget https://github.com/blackducksoftware/hub/archive/v2023.4.0.tar.gz
   ```

2. 解压缩 Black Duck .gz 文件：

   ```
   gunzip v2023.4.0.tar.gz
   ```

3. 解压缩 Black Duck.tar 文件：

   ```
   tar xvf v2023.4.0.tar
   ```

分发

docker-swarm 目录包含您安装或升级 Black Duck 所需的以下文件。

- blackduck-config.env：用于配置 Black Duck 设置的环境文件。
- docker-compose.bdba.yml：安装带有 Black Duck - Binary Analysis 的 Black Duck 和使用 Black Duck 提供的数据库容器时使用的 Docker Compose 文件。
- docker-compose.dbmigrate.yml：用于在使用 Black Duck 提供的数据库容器时迁移 PostgreSQL 数据库的 Docker Compose 文件。
- docker-compose.externaldb.yml：用于外部 PostgreSQL 数据库的 Docker Compose 文件。
- docker-compose.local-overrides.yml：用于覆盖 .yml 文件中的任何默认设置的 Docker Compose 文件。
- docker-compose.readonly.yml：将文件系统声明为 Swarm 服务的只读系统的 Docker Compose 文件。
- docker-compose.yml：使用 Black Duck 提供的数据库容器时的 Docker Compose 文件。
- external-postgres-init.pgsql：用于配置外部 PostgreSQL 数据库的 PostgresSQL.sql 文件。
- hub-bdba.env：包含 Black Duck - 二进制分析 的附加设置的环境文件。此文件不需要任何修改。
- hub-postgres.env：用于配置外部 PostgreSQL 数据库的环境文件。
- hub-webserver.env：用于配置 Web 服务器设置的环境文件。

在 bin 目录中：

- bd_get_source_upload_master_key.sh：用于在上传源文件时备份主密钥和密封密钥的脚本。
- hub_create_data_dump.sh：使用 Black Duck 提供的数据库容器时用于备份 PostgreSQL 数据库的脚本。
- hub_db_migrate.sh：用于在使用 Black Duck 提供的数据库容器时迁移 PostgreSQL 数据库的脚本。
- hub_reportdb_changepassword.sh：用于设置和更改报告数据库密码的脚本。
- recover_master_key.sh：创建用于上传源文件的新密封密钥的脚本。
- system_check.sh：用于收集 Black Duck 系统信息以发送给客户支持的脚本。

在 sizes-gen02 目录中：

- resources.yaml：这些是用于增强扫描的资源定义文件。

在 sizes-gen03 目录中：

- 10sph.yaml、120sph.yaml、250sph.yaml、500sph.yaml、1000sph.yaml、1500sph.yaml、2000sph.yaml: 这些是各种每小时扫描大小的资源定义文件。有关详细信息，请参阅硬件要求一节。

安装 Black Duck

在安装 Black Duck 之前，确定是否有任何需要配置的设置。

注: 这些说明适用于 Black Duck 的新安装。有关升级 Black Duck 的详细信息，请参阅第 6 章。

在以下 Black Duck 安装说明中，您可能需要是 Docker 组中的用户、root 用户或具有 sudo 访问权限。请参阅下一节，以通过非 root 用户的身份安装 Black Duck。

使用 PostgreSQL 数据库容器安装 Black Duck

1.
```
docker swarm init
```

   docker swarm init 命令创建单节点 swarm。

2.
```
docker stack deploy -c docker-compose.yml -c sizes-gen03/120sph.yaml -c docker-compose.local-overrides.yml hub
```

   在 Black Duck 2022.4.0 及更高版本中，您将不再直接在 docker-compose.yml 中分配容器资源。相反，在单独的覆盖文件中指定资源。在 sizes-gen02/resources.yaml 中找到了 Black Duck 2022.4 0 之前使用的资源分配。对于 Black Duck 2022.4.0 及更高版本，将在 sizes-gen03 文件夹中提供多个可能的分配。将根据测得的负载即每小时平均扫描数进行 7 次分配；如果您的预期负载与其中一个预定义的分配不匹配，请对其进行取整。

   在上例中：

   - docker-compose.yml：库存部署，不编辑此文件。
   - sizes-gen03/120.yaml：资源定义文件。在此示例中，sizes-gen03/120.yaml 将用作所需的资源定义，但可以更改此定义，以更好地适应您的扫描模式和用法。有关所有可用选项的列表，请参阅 分发 一节。请注意每个选项的系统要求，因为它们随着资源定义的增加而增加。
   - docker-compose.local-overrides.yml：如果您的安装具有自定义配置（比如密钥、内存限制等），则此文件为可选文件。

使用 PostgreSQL 数据库容器安装带有 Black Duck - 二进制分析 的 Black Duck

1.
```
docker swarm init
```

   docker swarm init 命令创建单节点 swarm。

2.
```
docker stack deploy -c docker-compose.yml -c sizes-gen03/120sph.yaml -c docker-compose.bdba.yml hub
```

   可以更改上面使用的资源定义文件，以更好地适应您的扫描模式和使用情况。有关所有可用选项的列表，请参阅 分发 一节。请注意每个选项的系统要求，因为它们随着资源定义的增加而增加。

安装带有外部 PostgreSQL 实例的 Black Duck

1.
```
docker swarm init
```

   docker swarm init 命令创建单节点 swarm。

2.
```
docker stack deploy -c sizes-gen03/120sph.yaml -c docker-compose.externaldb.yml hub
```

   可以更改上面使用的资源定义文件，以更好地适应您的扫描模式和使用情况。有关所有可用选项的列表，请参阅 分发 一节。请注意每个选项的系统要求，因为它们随着资源定义的增加而增加。

使用外部 PostgreSQL 实例安装带有 Black Duck - 二进制分析 的 Black Duck

1. ```
   docker swarm init
   ```

   docker swarm init 命令创建单节点 swarm。

2. ```
   docker stack deploy -c sizes-gen03/120sph.yaml -c docker-compose.externaldb.yml -c docker-compose.bdba.yml hub
   ```

   可以更改上面使用的资源定义文件，以更好地适应您的扫描模式和使用情况。有关所有可用选项的列表，请参阅 分发 一节。请注意每个选项的系统要求，因为它们随着资源定义的增加而增加。

   注: 在结合使用外部数据库和新的指导文件（用于部署调整）时，请在部署之前从要使用的 tshirt 大小文件中移除 postgres 和 postgres-upgrader 服务。

以只读方式安装带有文件系统的 Black Duck

1. ```
   docker swarm init
   ```

   docker swarm init 命令创建单节点 swarm。

2. ```
   docker stack deploy -c docker-compose.yml -c sizes-gen03/120sph.yaml -c docker-compose.readonly.yml hub
   ```

   可以更改上面使用的资源定义文件，以更好地适应您的扫描模式和使用情况。有关所有可用选项的列表，请参阅 分发 一节。请注意每个选项的系统要求，因为它们随着资源定义的增加而增加。

   要安装具有 Swarm 服务的只读文件系统的 Black Duck，请将 docker-compose.readonly.yml 文件添加到前面的说明中。

注: 有些 Docker 版本中，如果映像位于专用存储库中，Docker 堆栈将不会提取映像，除非将以下标志添加到上述命令中：--with-registry-auth。

验证安装

您可以通过运行 docker ps 命令查看每个容器的状态，从而确认安装是否成功。“正常”状态表示安装成功。请注意，安装后，容器可能会处于“正在启动”状态几分钟。

当 Black Duck 的所有容器都已启动后，Black Duck 的 Web 应用程序将在端口 443 上公开给 Docker 主机。请确保您已配置主机名，然后输入以下内容即可访问 Black Duck：

https://hub.example.com

首次访问 Black Duck 时，将出现《注册和最终用户许可协议》。您必须接受条款和条件才能使用 Black Duck。

输入提供给您的注册密钥以访问 Black Duck。

注: 如果您需要重新注册，则必须再次接受《最终用户许可协议》的条款和条件。

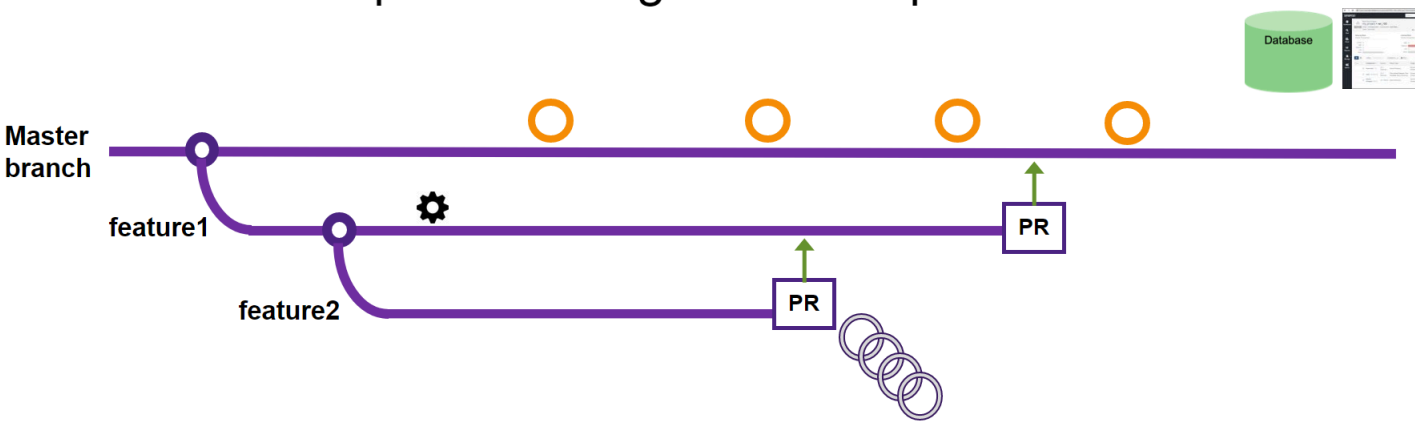
中的快速扫描 Black Duck

Black Duck Black Duck 2021.6.0 中默认提供快速扫描。

快速扫描（依赖关系扫描）为开发人员提供了一种高效的方法，可以在合并代码之前检查代码中是否存在漏洞或策略违反，而无需在 Black Duck 中创建 BOM。

您必须使用 Synopsys Detect 7.0.0 或更高版本才能使用快速扫描。

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 comm
- ✓ Pass/fail indicator based on policy violations
- ✓ Visibility into existing vulnerabilities
- ✓ Quick (<1 min) and scalable (>30k scans/da

4. 管理任务

本章介绍以下管理任务：

A

- 关于 KBMATCH_SENDPATH 参数。
- API 文档，提供通过代理服务器访问 API 文档的权限。
- API，提供从非 Black Duck 服务器访问 REST API 的权限。

B

- 二进制扫描文件，增加大小

C

- 证书，替换现有的自签名证书
- 更改 logstash 的主机名
- 清除未映射的代码位置
- 更改 bearer 令牌的过期时间
- 清除卡住的 BOM 事件
- 配置 HTML 报告下载大小
- 为 Blackduck 存储配置自定义卷

D

- 检测重复 BOM

H

- 分层 BOM，启用

I

- 忽略的组件，在报告中包括忽略的组件

L

- LDAP，配置安全 LDAP。
- 日志文件，访问

L

- LDAP，配置安全 LDAP。
- 日志文件，访问

M

- 内存限制，更改默认内存限制
- 从知识库检索迁移数据

O

- 覆盖文件，使用

P

- PostgreSQL，配置外部 PostgreSQL 实例。
- 代理设置，配置

R

- 报告数据库，配置报告数据库密码。
- 快速扫描，为完整 BOM 支持而配置

S

- 扩展 jobrunner、扫描和 binaryscanner 容器。
- 扫描监控配置
- 单点登录 (SSO)，配置
- 源文件，上传
- 启动或停止 Black Duck
- 支持，向支持部门提供您的 Black Duck 系统信息。
- Sysadmin 用户，了解默认的 sysadmin 用户。

T

- 时区，配置容器的时区

T

- 时区，配置容器的时区

U

- 用法，修改默认用法
- 用户 ID，自定义 Black Duck 容器的用户 ID

W

- Web 服务器设置，配置 Web 服务器设置，比如配置主机名、主机端口或禁用 IPv6。

环境文件和变量

使用环境文件

请注意，某些配置使用环境文件；例如，配置 Web 服务器、代理或外部 PostgreSQL 设置。用于配置这些设置的环境文件位于 docker-swarm 目录中。

要配置使用环境文件的设置：

- 要在安装 Black Duck 之前设置配置设置，请按如下所述编辑文件并保存更改。
- 要在安装 Black Duck 之后修改现有设置，请修改设置，然后在堆栈中重新部署服务。

环境变量和扫描二进制文件

当您使用 Black Duck - 二进制分析 (BDBA) 扫描二进制文件时，必须确保将 HUB_SCAN_ALLOW_PARTIAL='true' 参数添加到 jobrunner 容器环境变量中，以显示 BOM 中没有版本的组件。BDBA 扫描程序与 Black Duck 扫描不同，在二进制文件中无法识别版本字符串信息时，它可显示没有版本的组件。在 BOM 上，组件名称旁边会有一个问号 ( )，以向用户表示，在将安全漏洞分配给组件之前需要检查此组件，因为 Black Duck 需要一个版本以将安全漏洞映射到组件。

重复 BOM 检测

为提高扫描性能，默认情况下启用重复 BOM 检测功能。

如果该功能确定扫描将生成与现有 BOM 相同的 BOM，它将跳过 BOM 计算。您可以使用以下设置禁用它：

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

您可以在 blackduck-config.env 文件中更改此设置。

注：在 Black Duck 2021.4.0 中，只有当 Detect 发现的依赖关系组合与上一个扫描发现的组合相同时，此功能才会影响软件包管理器（依赖关系）扫描。此功能将在未来的版本中扩展。

更改 bearer 令牌的过期时间

要延长 REST API 中使用的 bearer 令牌的过期时间，请使用 docker-compose.local-overrides.yml 文件，通过使用新的过期值（以秒为单位）配置 HUB_AUTHENTICATION_ACCESS_TOKEN_EXPIRE 环境变量来覆盖默认设置。

HUB_AUTHENTICATION_ACCESS_TOKEN_EXPIRE 属性是访问令牌过期所需耗费的秒数。

注：过期配置更改仅适用于在更改 docker-compose.local-overrides.yml 文件中的设置后创建的 API 令牌。更改设置并重新启动服务时，不会为现有数据库记录/API 令牌更新您配置的过期时间。

关于 KBMATCH_SENDPATH 参数

KBMATCH_SENDPATH：此参数将排除文件路径和文件名，使其不会被我们的知识库用于匹配目的和准确性。Synopsys 不建议更改此项，因为它可能会对匹配结果产生一些影响。

要在 Black Duck 中关闭路径匹配，请在 blackduck-config.env 中将系统属性 KBMATCH_SENDPATH 设置为 false（默认设置为 true）。

注: 此参数自 2023.4.0 起已弃用，并将在将来的更新中移除。启用了 “匹配即服务” (Maas) 的 Black Duck 用户将无法使用此参数。想要继续使用此选项的客户将需要联系 Black Duck 支持部门，以便为其 Black Duck 注册密钥禁用 Maas。

通过代理服务器访问 API 文档

如果您使用的是反向代理并且该反向代理在子路径下具有 Black Duck，请配置 BLACKDUCK_SWAGGER_PROXY_PREFIX 属性，以便您可以访问 API 文档。BLACKDUCK_SWAGGER_PROXY_PREFIX 的值是 Black Duck 路径。例如，如果通过 “https://customer.companyname.com/hub” 访问 Black Duck，则 BLACKDUCK_SWAGGER_PROXY_PREFIX 的值将为 “hub” 。

要配置此属性，请编辑 docker-swarm 目录中的 blackduck-config.env 文件。

提供从非 Black Duck 服务器访问 REST API 的权限

您可能希望从非 Black Duck 服务器提供的网页访问 Black Duck REST API。要允许从非 Black Duck 服务器访问 REST API，必须启用跨源站资源共享 (CORS)。

用于为 Black Duck 安装启用和配置 CORS 的属性包括：

属性	说明
BLACKDUCK_HUB_CORS_ENABLED	必填。定义是否启用 CORS；“true” 表示启用了 CORS。
BLACKDUCK_CORS_ALLOWED_ORIGINS_PROP_NAME	必填。CORS 允许的源站。浏览器在发出跨源站请求时发送源站服务器标头。这是必须在 blackduck.hub.cors.allowedOrig BLACKDUCK_CORS_ALLOWED_ ORIGINS_PROP_NAME 属性中列出的源站。例如，如果运行的服务器提供的页面来自 http://123.34.5.67:8080，则浏览器应将此设置为源站服务器，并将此值添加到属性中。请注意，协议、主机和端口必须匹配。使用逗号分隔的列表指定多个基本源站 URL。

属性	说明
BLACKDUCK_CORS_ALLOWED_HEADERS_PROP_NAME	可选。可用于发出请求的标头。
BLACKDUCK_CORS_EXPOSED_HEADERS_PROP_NAME	可选。请求 CORS 的浏览器可以访问的标头。
BLACKDUCK_CORS_ALLOWED_ORIGIN_PATTERNS_PROP_NAME	BLACKDUCK_CORS_ALLOWED_ PROP_NAME*,* 的替代方案，支持通过通配符模式声明的源站。BLACKDUCK_CORS_ALLOW PROP_NAME 将覆盖 BLACKDUCK_CORS_ALLOWED_
BLACKDUCK_CORS_ALLOW_CREDENTIALS_PROP_NAME	指定浏览器是否应将身份凭据（比如 Cookie）与跨域请求一起发送到注释的端点。在预检请求的 Access-Control-Allow-Credentials 响应标头上设置已配置的值。配置 ALLOW_CREDENTIALS=true 和 ALLOWED_ORIGIN=* 是无效的。如果允许的源站需要“ALL”配置值，则应使用 ALLOWED_ORIGIN_PATTERNS 配置属性进行配置，而不是继续。

要配置这些属性，请编辑 docker-swarm 目录中的 blackduck-config.env 文件。

增加二进制扫描文件的大小

使用 Black Duck - 二进制分析 时，可扫描的二进制文件的最大大小为 6 GB。您可以通过将环境变量 BINARY_UPLOAD_MAX_SIZE 添加到 docker-swarm 目录中的 hub_webserver.env 文件并指定一个以兆字节为单位的值来增加此限制。

例如，要将最大二进制扫描增加到 7 GB，请添加以下内容：

BINARY_UPLOAD_MAX_SIZE=7168m

配置仪表板刷新率

通过配置 blackduck-config.env 文件中的 SEARCH_DASHBOARD_REFRESH_JOB_DUTY_CYCLE 环境变量来计划仪表板刷新率。

允许的 SEARCH_DASHBOARD_REFRESH_JOB_DUTY_CYCLE 值如下所示：

- 默认值为 20
- 最小值为 10。
- 最大值为 50。

与允许值不匹配的值将重置为最接近的允许值。

示例：

SEARCH_DASHBOARD_REFRESH_JOB_DUTY_CYCLE=100 配置的值将重置为最接近的允许值（10、20 或 50），即 50。

SEARCH_DASHBOARD_REFRESH_JOB_DUTY_CYCLE=10 配置的值将保持为此允许值，即 10。

SEARCH_DASHBOARD_REFRESH_JOB_DUTY_CYCLE=5 配置的值将重置为最接近的允许值（10、20 或 50），即 10。

管理证书

默认情况下，Black Duck 使用 HTTPS 连接。用于运行 HTTPS 的默认证书是自签名证书，这意味着它是在本地创建的，未由认可的证书颁发机构 (CA) 签署。

如果您使用此默认证书，则需要设置安全例外才能登录 Black Duck 的 UI，因为您的浏览器无法识别证书的颁发机构，因此默认情况下不会接受它。

扫描时连接到 Black Duck 服务器时，您还会收到一条有关证书的消息，因为扫描程序无法验证该证书，因为它是自签名的，不是由 CA 颁发的。

您可以从您选择的证书颁发机构获取签署的 SSL 证书。要获取已签署的 SSL 证书，请创建证书签名请求 (CSR)，然后 CA 使用该请求创建一个证书，该证书将把运行您的 Black Duck 实例的服务器标识为“安全”。从 CA 收到签署的 SSL 证书后，可以替换自签名证书。

要创建 SSL 证书密钥库：

1. 在命令行中，要生成 SSL 密钥和 CSR，请键入：

   openssl genrsa -out <keyfile> <keystrength>

   openssl req -new -key <keyfile> -out <CSRfile>

   其中：

   - <keyfile> 是 <your company’ s server name>.key
   - <keystrength> 是您的站点的公共加密密钥的大小
   - <CSRfile> 是 <your company’ s server name>.csr

   注: 为公司服务器输入的名称必须是 SSL 服务器将驻留的完整主机名，并且组织名称必须与域的 “whois” 记录中的名称相同。

   例如：

   openssl genrsa -out server.company.com.key 1024

   openssl req -new -key server.company.com.key -out server.company.com.csr

   此示例为 server.company.com 创建 CSR 以从 CA 获取证书。
2. 通过首选方法（通常通过 Web 门户）将 CSR 发送给 CA。
3. 指示您需要 Apache Web 服务器的证书。
4. 向 CA 提供请求的有关您的公司的任何信息。此信息必须与您的域注册表信息匹配。
5. 从 CA 收到证书后，请按照下一节中的说明将证书上传到 Black Duck 实例中。

使用自定义证书

webserver 容器具有从 Docker 获得的自签名证书。您可能需要用自定义证书密钥对替换此证书。

1. 使用 docker secret 命令，通过 WEBSERVER_CUSTOM_CERT_FILE 和 WEBSERVER_CUSTOM_KEY_FILE 向 Docker Swarm 告知证书和密钥。密钥的名称必须包括堆栈名称。在以下示例中，堆栈名称是 “hub” ：

   ```
   docker secret create hub_WEBSERVER_CUSTOM_CERT_FILE <certificate file>
   docker secret create hub_WEBSERVER_CUSTOM_KEY_FILE <key file>
   ```
2. 将密钥添加到 docker-compose.local-overrides.yml 文件中的 webserver 服务：

   ```
   webserver:
   secrets:
    - WEBSERVER_CUSTOM_CERT_FILE
    - WEBSERVER_CUSTOM_KEY_FILE
   ```
3. 从位于 docker-swarm 目录中的 docker-compose.local-overrides.yml 文件末尾的 secrets 部分移除注释字符 (#)：

   ```
   secrets:
     WEBSERVER_CUSTOM_CERT_FILE:
       external: true
       name: "hub_WEBSERVER_CUSTOM_CERT_FILE"
     WEBSERVER_CUSTOM_KEY_FILE:
       external: true
       name: "hub_WEBSERVER_CUSTOM_KEY_FILE"
   ```
4. docker-compose.local-overrides.yml 文件中的 webserver 服务内的 healthcheck 属性必须指向来自密钥的新证书：

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

5. 通过运行以下命令重新部署堆栈：

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

使用自定义证书颁发机构进行证书身份验证

您可以使用自己的证书颁发机构进行证书身份验证。

要使用自定义证书颁发机构：

1. 将名为 AUTH_CUSTOM_CA 的 Docker 密钥（自定义证书颁发机构证书文件）添加到位于 docker-swarm 目录中的 docker-compose.local-overrides.yml 文件内的 webserver 和身份验证服务：

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

2. 将以下文本添加到 docker-swarm 目录中的 docker-compose.local-overrides.yml 文件的末尾：

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

3. 启动 webserver 容器和身份验证服务。
4. 启动 Black Duck 服务后，发出 API 请求，该请求将返回 JSON Web 令牌 (JWT) 以及与受信任的证书颁发机构签署的证书密钥对。例如：

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

注：用于身份验证的证书的用户名必须作为其常用名 (CN) 存在于 Black Duck 系统中。

从知识库获取组件迁移数据

使用迁移 API 从 Black Duck 知识库获取新的或更改的组件 ID。这些 API 返回知识库中已迁移组件的原始数据和新 ID。

您可以使用 API 提交旧组件 ID，例如组件 ID：bf368a1d-ef4f-422c-baca-a138737595e7，以从知识库获取新组件 ID。

迁移跟踪 API 可以获取特定组件或组件版本的迁移信息，或获取特定日期发生的迁移的详细信息。

启用迁移记录

要启用迁移记录，请在 中设置以下属性 blackduck-config.env file.

RECORD_MIGRATIONS = true（默认值为 false）

这使得可以在检测到迁移时写入记录。

保留迁移数据

要设置返回记录的天数，请在 blackduck-config.env 文件中配置以下属性：

MIGRATED_OBJECT_RETENTION_DAYS = <number_of_days>（默认值为 30 天）

API 端点

转至 API 文档以开始使用以下 API。

对于在特定日期之后发生的迁移：

```
GET /api/component-migrations
```

对于特定组件或组件版本的迁移：

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

有关详细信息，请参阅 https://<blackduck_server>/api-doc/public.html#_component_component_version_migration_endpoints 上的 Black Duck API 文档。

启用分层 BOM

默认情况下，分层 BOM 被禁用。要启用此功能，请将 HUB_HIERARCHICAL_BOM 环境变量添加到 .env 文件中。将值设置为 “true” ，例如 HUB_HIERARCHICAL_BOM=true。将该值重置为 “false” 将禁用该功能。

您也可以在位于 docker-swarm 目录中的 docker-compose.local-overrides.yml 文件内编辑 webapp 服务：

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

请注意，如果有多个环境变量，请用逗号 (,) 分隔它们。例如：

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

在报告中包括忽略的组件

默认情况下，忽略的组件以及与这些已忽略组件关联的漏洞将从漏洞状态报告、漏洞更新报告、漏洞修复报告和项目版本报告中排除。要包括忽略的组件，请将 docker-swarm 目录中的 blackduck-config.env 文件内的 BLACKDUCK_REPORT_IGNORED_COMPONENTS 环境变量的值设置为 “true” 。

将 BLACKDUCK_REPORT_IGNORED_COMPONENTS 的值重置为 “false” 将排除忽略的组件。

配置安全 LDAP

如果在将安全 LDAP 服务器连接到 Black Duck 时看到证书问题，最可能的原因是 Black Duck 服务器尚未设置与安全 LDAP 服务器的信任连接。如果使用自签名证书，通常会发生这种情况。

要设置与安全 LDAP 服务器的信任连接，请通过以下方式将服务器证书导入本地 Black Duck LDAP 信任存储区：

1. 获取 LDAP 信息。
2. 使用 Black Duck UI 导入服务器证书。

注：所有托管客户都应该通过 SAML 或 LDAP，利用我们为单点登录 (SSO) 提供的现成支持来保护对 Black Duck 应用程序的访问。有关如何启用和配置这些安全功能的信息，请参见安装指南。此外，我们鼓励使用 SAML SSO 提供程序（提供双因素授权功能）的客户也启用和利用该技术来进一步保护对 Black Duck 应用程序的访问。

获取 LDAP 信息

请与 LDAP 管理员联系并收集以下信息：

LDAP 服务器详细信息

这是 Black Duck 用于连接目录服务器的信息。

- （必需）目录服务器的主机名或 IP 地址，包括实例侦听的协议方案和端口。
  示例：ldaps://<server_name>.<domain_name>.com:339
- （可选）如果您的组织不使用匿名身份验证，并且需要 LDAP 访问凭据，则需提供有权读取目录服务器的用户的密码和 LDAP 名称或绝对 LDAP 判别名 (DN)。
  绝对 LDAP DN 示例：uid=ldapmanager,ou=employees,dc=company,dc=com

  LDAP 名称示例：jdoe
- （可选）如果 LDAP 访问需要凭据，则使用的身份验证类型为：简单或 digest-MD5。

LDAP 用户属性

这是 Black Duck 用于在目录服务器中查找用户的信息：

- （必需）用户可位于其下的绝对基本 DN。
  示例：dc=example,dc=com
- （必需）用于匹配特定唯一用户的属性。此属性的值使用用户名对用户档案图标进行个性化设置。
  示例：uid={0}

测试用户名和密码

- （必需）用于测试目录服务器连接的用户凭据。

导入服务器证书

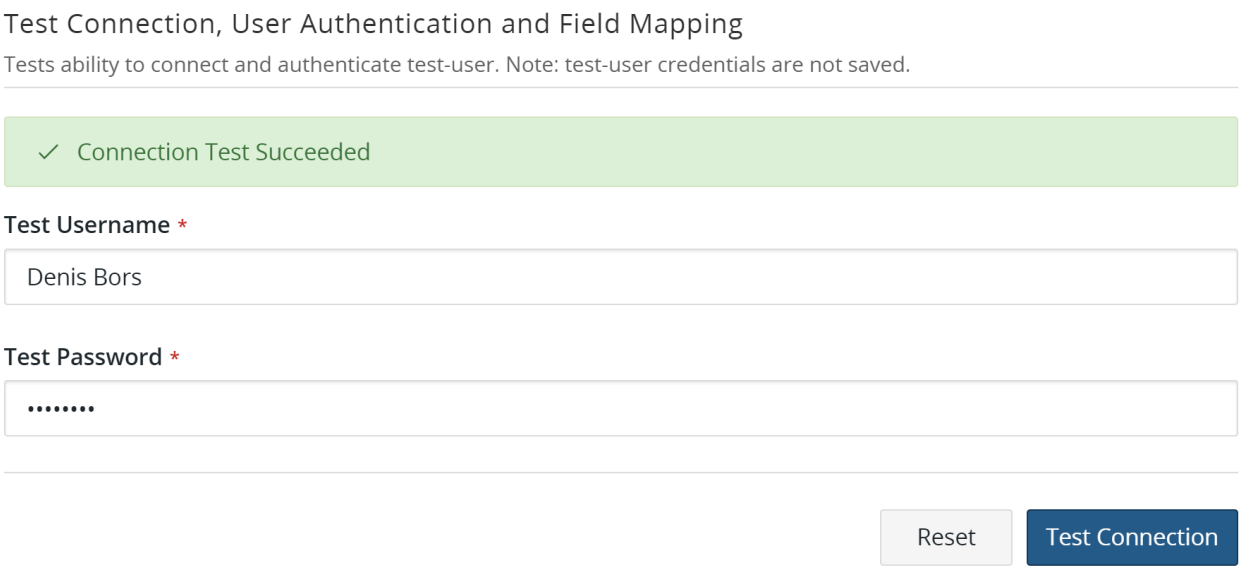
要导入服务器证书：

1. 以系统管理员身份登录 Black Duck。
2. 单击

。
3. 选择系统设置。
4. 单击用户身份验证。
5. 选中启用 LDAP 配置复选框并在 LDAP 服务器详细信息部分中填写信息，如上所述。在服务器 URL 字段中，确保已配置安全 LDAP 服务器：协议方案为 ldaps://。
6. 如上所述，在 LDAP 用户属性部分中填写信息。

   （可选）清除在 Black Duck 中自动创建用户帐户复选框以关闭使用 LDAP 进行身份验证时自动创建用户的功能。默认情况下，此复选框处于选中状态，因此，在使用 LDAP 登录到 Black Duck 时，将自动创建不存在于 Black Duck 中的用户。这适用于新安装和升级。
7. 在测试连接、用户身份验证和字段映射部分输入用户凭据，然后单击测试连接。

8. 如果证书没有问题，则会自动导入证书，并显示“连接测试成功”消息：



9. 如果证书存在问题，则会出现一个对话框，列出有关证书的详细信息。执行以下操作之一：
   - 单击取消以修复证书问题。

     修复后，重新测试连接以验证证书问题是否已修复且证书是否已导入。如果成功，将显示“连接测试成功”消息。
   - 单击保存以导入此证书。

     通过单击测试连接验证证书是否已导入。如果成功，将显示“连接测试成功”消息。

LDAP 信任存储密码

如果添加自定义 Black Duck Web 应用程序信任存储，请使用以下方法指定 LDAP 信任存储密码。

使用 Docker Swarm 时，请使用以下方法指定 LDAP 信任存储密码。

- 使用 docker secret 命令，通过 LDAP_TRUST_STORE_PASSWORD_FILE 向 Docker Swarm 告知密码。密钥的名称必须包括堆栈名称。在此示例中，“HUB”是堆栈名称：

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

  将密码密钥添加到 docker-swarm 目录中的 docker-compose.local-overrides.yml 文件内的 webapp 服务：

  ```
    secrets:
      - LDAP_TRUST_STORE_PASSWORD_FILE
  ```

  将文本（如以下内容）添加到位于 docker-compose.local-overrides.yml 文件末尾的 secrets 部分：

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

- 通过在位于 docker-swarm 目录中的 docker-compose.local-overrides.yml 文件中添加 webapp 服务的卷部分，将包含名为 LDAP_TRUST_STORE_PASSWORD_FILE 的文件的目录挂载到 /run/secrets。

  ```
  webapp:
  ```

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

注: 如果信任存储被完全替换，并且受其他密码保护，则只需挂载包含 LDAP_trust_store_password_file 的目录。

下载日志文件和热图数据

您可能需要对问题进行故障排除或向客户支持人员提供日志文件。具有“系统管理员”角色的用户可以下载包含当前日志文件或您的系统的热图数据的压缩文件。

请注意，准备日志文件或热图文件可能需要几分钟时间。有关获取日志和配置热图数据的详细信息，请参阅安装指南。

访问日志文件

要从 Black Duck UI 下载日志文件

1. 以“系统管理员”角色登录 Black Duck。
2. 单击

。
3. 选择作业。
4. 选择系统信息选项卡。
5. 单击下载日志 (.zip)。

访问热图数据

要审查和分析终端扫描趋势，您可以下载压缩 CSV 格式的热图，并在电子表格程序中将热图创建为透视图。

要为您的系统下载热图数据：

1. 以“系统管理员”角色登录 Black Duck。
2. 单击

。
3. 在“管理”菜单的诊断部分选择作业。
4. 选择系统信息选项卡。
5. 单击下载热图 (.zip)。

查看日志文件

获取日志

要从容器中获取日志：

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

其中“logs/”是日志将复制到的本地目录。

查看容器的日志文件

使用 docker-compose logs 命令查看所有日志：

```
docker-compose logs
```

有关 Docker 命令的更多信息，请访问 Docker 文档网站：https://docs.docker.com/

清除日志

默认情况下，日志文件在 14 天后自动清除。要修改此值：

1. 停止容器。
2. 编辑位于 docker-swarm 目录中的 docker-compose.local-overrides.yml 文件：
   a. 添加 logstash 服务。
   b. 使用新值添加 DAYS_TO_KEEP_LOGS 环境变量。此示例在 10 天后清除日志文件：

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

3. 重新启动容器。

更改默认内存限制

某些容器可能需要高于默认内存限制的内存，具体取决于 Black Duck 上的负载。

注: 切勿降低默认内存限制，因为这会导致 Black Duck 无法正常工作。

您可以更改以下容器的默认内存限制：

- webapp
- jobrunner
- 扫描
- binaryscanner
- bomengine

更改默认 webapp 容器内存限制

webapp 容器有三种内存设置：

- HUB_MAX_MEMORY 环境变量控制最大 Java 堆大小。
- limits memory 和 reservations memory 设置控制 Docker 用于计划和限制 webapp 容器总内存的限制。
  - limits memory 设置是容器可以使用的内存量。
  - Docker 使用 reservations memory 设置确定是否可以将容器部署（计划）到机器中。通过使用此值，Docker 可确保部署到计算机的所有容器都有足够的内存，而不是所有容器都要争用相同的内存。

  请注意，每个设置的值必须大于最大 Java 堆大小。如果更新 Java 堆大小，Black Duck Software 建议将 limits memory 和 reservations memory 值各自设置为比最大 Java 堆大小高出至少 1GB。

使用 docker-compose.local-overrides.yml 文件中的 webapp 部分，如有必要，移除注释字符 (#) 并输入新值。

以下示例将 Web 应用程序容器的最大 Java 堆大小更改为 8GB ，并将 limit memory 和 reservations memory 设置的值各自更改为 9GB。

原始值：

```
#webapp:
#environment: {HUB_MAX_MEMORY: REPLACE_WITH_NEW_MAX_MEMORYm}
#deploy:
```

```
  #limits: {MEMORY: REPLACE_WITH_NEW_MEM_LIMITm}
  #reservations: {MEMORY: REPLACE_WITH_NEW_VALUEm}
```

更新后的值：

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

更改默认的 jobrunner 容器内存限制

Jobrunner 容器有三种内存设置：

- HUB_MAX_MEMORY 环境变量控制最大 Java 堆大小。
- limits memory 和 reservations memory 设置控制 Docker 用于计划和限制 jobrunner 容器总内存的限制。
  - limits memory 设置是容器可以使用的内存量。
  - Docker 使用 reservations memory 设置确定是否可以将容器部署（计划）到机器中。通过使用此值，Docker 可确保部署到计算机的所有容器都有足够的内存，而不是所有容器都要争用相同的内存。

  请注意，每个设置的值必须大于最大 Java 堆大小。如果更新 Java 堆大小，Black Duck Software 建议将 limits memory 和 reservations memory 值各自设置为比最大 Java 堆大小高出至少 1GB。

注: 这些设置适用于所有 jobrunner 容器，包括扩展的 jobrunner 容器。

使用 docker-compose.local-overrides.yml 文件中的 jobrunner 部分，如有必要，移除注释字符 (#) 并输入新值。

以下示例将 jobrunner 容器的最大 Java 堆大小更改为 8GB ，并将 limit memory 和 reservations memory 设置的值各自更改为 9GB。

原始值：

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

更新后的值：

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

更改默认内存限制

如果在修改 HUB_MAX_MEMORY 值时为容器提供了更多内存，则可以更改以下值。

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

指导示例

```
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
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: “大”和“超大”仅在 jobrunner 实例的数量上有所不同。内存和线程数相同。

注: 两个池的最大线程数不应超过 32

更改默认的扫描容器内存限制

扫描容器有三种内存设置：

- HUB_MAX_MEMORY 环境变量控制最大 Java 堆大小。
- limits memory 和 reservations memory 设置控制 Docker 用于计划和限制扫描容器总内存的限制。
  - limits memory 设置是容器可以使用的内存量。
  - Docker 使用 reservations memory 设置确定是否可以将容器部署（计划）到机器中。通过使用此值，Docker 可确保部署到计算机的所有容器都有足够的内存，而不是所有容器都要争用相同的内存。

  请注意，每个设置的值必须大于最大 Java 堆大小。如果更新 Java 堆大小，Black Duck Software 建议将 limits memory 和 reservations memory 值各自设置为比最大 Java 堆大小高出至少 1GB。

注: 这些设置适用于所有扫描容器，包括扩展的扫描容器。

使用 docker-compose.local-overrides.yml 文件中的 scan 部分，如有必要，移除注释字符 (#) 并输入新值。

以下示例将扫描容器的最大 Java 堆大小增加为 4GB ，并将 limits memory 和 reservations memory 设置的值各自更改为 5GB。

原始值：

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

更新后的值：

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

更改默认的 binaryscanner 容器内存限制

binaryscanner 容器的唯一默认内存大小是容器的实际内存限制。

注: 这些设置适用于所有 binaryscanner 容器，包括扩展的 binaryscanner 容器。

将 binaryscanner 部分添加到 docker-compose.local-overrides.yml 文件中。

以下示例将容器内存限制更改为 4GB。

更新后的值：

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

更改默认的 bomengine 容器内存限制

bomengine 容器有三种内存设置：

- HUB_MAX_MEMORY 环境变量控制最大 Java 堆大小。
- limits memory 和 reservations memory 设置控制 Docker 用于计划和限制 bomengine 容器总内存的限制。
  - limits memory 设置是容器可以使用的内存量。
  - Docker 使用 reservations memory 设置确定是否可以将容器部署（计划）到机器中。通过使用此值，Docker 可确保部署到计算机的所有容器都有足够的内存，而不是所有容器都要争用相同的内存。

注: 这些设置适用于所有 bomengine 容器，包括扩展的 bomengine 容器。

使用 docker-compose.local-overrides.yml 文件中的 bomengine 部分，如有必要，移除注释字符 (#)，然后输入新值。

以下示例将 bomengine 容器的最大 Java 堆大小更改为 8GB ，并将 limits memory 和 reservations memory 设置的值各自更改为 9GB。

原始值：

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

更新后的值：

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

重要: Synopsys 建议使用与 jobrunner 容器相同的值分配 HUB_MAX_MEMORY 并限制 bomengine 容器的内存。

更改 logstash 的主机名

当您更改 logstash 的主机名和服务名称时，它们不会被推送到内部 PostgreSQL 容器。例如，您可能希望将 logstash 主机名更改为 bdlogstash，因为您正在将 logstash 用于其他用途，现在您希望将 PostgreSQL 日志写入 bdlogstash；请进行以下更改：

1. 在 blackduck-config.env 中，将 logstash 更改为 bdlogstash HUB_LOGSTASH_HOST=bdlogstash。

2. 编辑 docker-compose.yml 并将 logstash 更改为 bdlogstash。

   bdlogstash：

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

3. 将 env_file: [blackduck-config.env] 添加到 docker-compose.yml 中的 postgres 容器定义，以便它读取主机名更改。

   env_file: [blackduck-config.env]

清除未映射的代码位置

未映射的代码位置是未映射到项目版本的代码位置。在 Black Duck 内部，代码位置由一个或多个扫描表示。

- 有时，代码扫描会创建代码位置，而不会将它们映射到项目版本，这会导致代码位置未映射。
- 当用户删除项目版本时，代码位置/扫描数据可能会被孤立。

不希望清除未映射的代码位置数据的用户可能希望在 blackduck-config.env 文件中禁用此功能，默认情况下，该文件设置为 true，每 30 天清除一次。

定期扫描并希望经常丢弃数据的用户可能希望将保留期设置为较短的天数，例如 14 天。

要计划清理未映射的代码位置，请在 blackduck-config.env 文件中配置以下属性：

- BLACKDUCK_HUB_UNMAPPED_CODE_LOCATION_CLEANUP=true

  默认设置为 true。

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

  默认设置为 30 天。

注：当您配置清理未映射的代码位置并重新启动系统时，扫描清除将开始移除符合保留条件（早于配置的保留天数）的未映射代码位置。默认情况下，扫描清除作业每 15 分钟运行一次。

使用 cron 字符串计划扫描清除作业

您可以通过在 blackduck-config.env 文件中为 Docker swarm 实施设置变量来配置扫描清除。

可以使用以下任一方法在 blackduck-config.env 中设置变量以计划扫描清除作业：

- 使用 cron 作业： blackduck.scan.processor.scanpurge.cronstring
  使用以下 cron 格式： second minute hour dayOfMonth month daysOfWeek
- 使用固定延迟：blackduck.scan.processor.scanpurge.fixeddelay（配置为以毫秒为单位运行 scanpurgejob 的频率，默认为 15 分钟）

注：如果提供了 blackduck.scan.processor.scanpurge.cronstring，则会忽略 blackduck.scan.processor.scanpurge.fixeddelay 设置，因为改为使用 cron 字符串。

清除卡住的 BOM 事件

BOM 事件清理作业可清除可能因处理错误而卡住的 BOM 事件。默认情况下，作业每天在午夜运行，并移除过去 24 小时之前的卡住事件。如果需要，用户可以根据其系统的静默时间更改 cron 计划，也可以在 1 - 48 之间更改过去保留的小时数。

- 为保留 BOM 事件的过去小时数设置保留期。这对于清除由于处理错误或拓扑更改而卡住的 BOM 事件非常有用。默认值为 24 小时。有效范围为 1 - 48 小时。例如，如果要移除过去 12 小时之前的所有卡住事件。

  ```
  BLACKDUCK_BOM_EVENT_CLEANUP_BEFORE_HOURS=12
  ```

- 通过 Cron 表达式 (when should job run) 设置计划，以清除 BOM 事件。建议在系统的静默时间运行它。默认情况下，它在午夜运行，值为 0 0 * * * ? 例如，如果 BOM 事件清理作业应在每天凌晨 2:00 运行

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

默认情况下启用 VersionBomEventClearnupJob，您无法禁用此作业。

使用覆盖文件

您可能需要覆盖 Black Duck 使用的某些默认设置。使用 docker-swarm 目录中的 docker-compose.local-overrides.yml，而不是直接编辑 .yml 文件。

通过使用此文件修改默认设置，您的更改将在升级时保留：每次 Black Duck 升级后，您不再需要修改 .yml 文件。T

请注意，在 docker-compose 命令中，docker-compose.local-overrides.yml 文件必须是最后使用的 .yml 文件。例如，以下命令使用外部数据库启动 Black Duck：

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

在 中配置分析 Black Duck

在 Black Duck 中，您可以通过关闭 blackduck-config.env 文件中的分析来全局禁用 Synopsys Detect 的 Phone Home 功能。

1. 在 blackduck-config.env 文件中，配置 ANALYTICS=false。
2. 重新启动 Black Duck。

配置外部 PostgreSQL 实例

Black Duck 支持使用外部 PostgreSQL 实例。对于 Black Duck 2022.10.x，支持 PostgreSQL 版本 13.x 和 14.x。

注: 将 azure.extensions 值设置为 PGCRYPTO 可启用 init 脚本以访问 Azure 数据库。

Synopsys 建议对使用外部 PostgreSQL 的新安装使用 PostgreSQL 14.5（或 14.x 的更高版本）。也支持 PostgreSQL 13.x。

重要: Black Duck 2022.10.0 不再支持 PostgreSQL 11.x。Black Duck 2023.4.0 增加了对使用 PostgreSQL 15 作为外部数据库的初步支持。此支持仅用于测试；不支持生产用途。

要配置外部 PostgreSQL 数据库：

1. 使用 UTF8 编码初始化外部 PostgreSQL 群集。完成此操作的方法可能取决于您的外部 PostgreSQL 提供商。例如，使用 PostgreSQL initdb 工具时，请运行以下命令：

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

2. 创建和配置数据库用户名和密码。PostgreSQL 数据库有三个用户：一个管理员（默认情况下，blackduck 是用户名）、一个用户（默认情况下，blackduck_user 是用户名）和一个 Black Duck 报告数据库用户（默认情况下，blackduck_reporter 是用户名）。您可以：
   - 使用默认用户名创建帐户。
   - 使用自定义用户名创建帐户。
3. 配置 PostgreSQL 实例。

使用默认用户名创建和配置帐户：

按照以下说明使用默认的 blackduck、blackduck_user 和 blackduck_reporter 用户名。

完成这些步骤后，转至配置 PostgreSQL 实例。

1. 创建一个名为 blackduck、具有管理员权限的数据库用户。

   对于 Amazon RDS，在创建数据库实例时将 “主用户” 设置为 blackduck。

   不需要其他特定值。
2. 在 external-postgres-init.pgsql 文件内（在 docker-swarm

   目录中）替换以下内容。

   将 POSTGRESQL_USER 替换为 blackduck

   将 HUB_POSTGRES_USER 替换为 blackduck_user

   将 BLACKDUCK_USER_PASSWORD 替换为您用于 的密码 blackduck_user

   重要：此步骤是必需的。
3. 运行位于 docker-swarm 目录中的 external-postgres-init.pgsql 脚本，以创建用户、数据库和其他必要项目。例如：

   ```
   psql -U blackduck -h <hostname> -p <port> -f external-postgres-init.pgsql postgres
   ```
4. 使用首选的 PostgreSQL 管理工具，为 blackduck、blackduck_user 和 blackduck_reporter 数据库用户配置密码。

   这些用户是由 external-postgres-init.pgsql 脚本在上一步中创建的。
5. 转至配置 PostgreSQL 实例。

注：Black Duck 使用 pgcrypto 扩展并希望它可用。默认情况下，CloudSQL、RDS 和 Azure 提供了 pgcrypto 扩展，因此无需进一步配置。社区 PostgreSQL 的用户需要安装 postgresql-contrib 软件包（如果尚未安装）。请注意，软件包名称因发行版而异。

使用自定义用户名和密码创建和配置帐户：

使用以下说明创建自定义数据库用户名。

在这些说明中：

- DBAdminName 是新的自定义管理员的用户名。
- DBUserName 是新的自定义数据库用户的用户名。
- DBReporterName 是新的自定义数据库报告者的用户名。

仅由数字组成的用户名可用于外部 PostgreSQL 实例中的 PostgreSQL 用户名。

更新后的 external-postgres-init.pgsql 脚本对用户名使用双引号括起来，以便数字 PostgreSQL 用户名可以在脚本中成功运行。在 external-postgres-init.pgsql 脚本中，您将搜索 HUB_POSTGRES_USER 和 POSTGRESQL_USER 的默认名称值，并将其替换为数字用户名。

完成这些步骤后，转到下一节配置 PostgreSQL 实例。

1. 创建一个名为 DBAdminName、具有管理员权限的数据库用户。

   对于 Amazon RDS，在创建数据库实例时将 “主用户” 设置为 DBAdminName。

   不需要其他特定值。

2. 编辑位于 docker-swarm 目录中的 external-postgres-init.pgsql 脚本，使用的帐户名称是您要用于 DBAdminName、DBUserName 和 DBReporterName 的帐户名称。

3. 运行位于 docker-swarm 目录中已编辑的 external-postgres-init.pgsql 脚本，以创建用户、数据库和其他必要项目。例如：

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

4. 使用首选的 PostgreSQL 管理工具，为 DBAdminName、DBUserName 和 DBReporterName 数据库用户配置密码。

   这些用户是由 external-postgres-init.pgsql 脚本在上一步中创建的。

5. 编辑 hub-postgres.env 环境文件。该文件列出 HUB_POSTGRES_USER 和 HUB_POSTGRES_ADMIN 的默认用户名。将这些默认值替换为数据库用户和管理员的自定义用户名。

6. 转到下一节配置 PostgreSQL 实例。

配置 PostgreSQL 实例：

创建用户并配置密码后，请完成以下步骤：

1. 编辑位于 docker-swarm 目录中的 hub-postgres.env 环境文件，以指定数据库连接参数。您可以选择：

   - 在数据库连接中启用 SSL。

     对于身份验证，您可以选择使用证书或用户名和密码，或同时使用两者。

   - 在数据库连接中禁用 SSL。

     如果禁用 SSL，则必须使用用户名和密码身份验证。

参数	说明
HUB_POSTGRES_ENABLE_SSL	定义在数据库连接中是否使用 SSL。 • 将该值设置为 “false” 可禁止在数据库连接中使用 SSL。这 • 将该值设置为 “true” 可允许在数据库连接中使用 SSL。
HUB_POSTGRES_ENABLE_SSL_CERT	定义身份验证是否需要证书。 • 将该值设置为 “false” 可禁用客户端证书身份验证。这是默 • 将该值设置为 “true” ，可在数据库连接中使用 SSL 时要求
HUB_POSTGRES_HOST	带有 PostgreSQL 实例的服务器的主机名。
HUB_POSTGRES_PORT	要为 PostgreSQL 实例连接的数据库端口。

2. 如果您使用的是用户名和密码身份验证，请将 PostgreSQL 管理员和用户密码提供给 Black Duck：

   a. 使用数据库用户的密码创建名为 HUB_POSTGRES_USER_PASSWORD_FILE 的文件。如果使用的是默认用户名或上一示例中的 DBUserName，则这是 blackduck_user 用户名。

   b. 使用数据库管理员用户的密码创建名为 HUB_POSTGRES_ADMIN_PASSWORD_FILE 的文件。如果使用的是默认用户名或上一示例中的 DBAdminName，则这是 blackduck 用户名。

   c. 将包含两个文件的目录挂载到 /run/secrets。使用位于 docker-swarm 目录中的 docker-compose.local-overrides.yml 文件。对于每个服务（webapp、jobrunner、身份验证、bomengine 和扫描），执行以下操作：

      1. 如有必要，请移除服务名称前的注释字符 (#)。
      2. 将卷挂载添加到服务。

      此示例将卷添加到 webapp 服务：

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

      您还需要将此文本添加到身份验证、jobrunner、bomengine 和扫描服务中。

   您可以使用 docker secret 命令创建一个名为 HUB_POSTGRES_USER_PASSWORD_FILE 的密钥和一个名为 HUB_POSTGRES_ADMIN_PASSWORD_FILE 的密钥，而不是执行第 2a-c 步。

   a. 使用 docker secret 命令，向 Docker Swarm 告知密钥。密钥的名称必须包括堆栈名称。在以下示例中，堆栈名称是“hub”：

      ```
      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. 将密码密钥添加到 docker-compose.local-overrides.yml 文件中的 webapp、jobrunner、身份验证、bomengine 和扫描服务。此示例适用于 webapp 服务：

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

      如有必要，请移除注释字符 (#)。

      移除注释字符，如有必要，将堆栈名称更改为 docker-compose.local-overrides.yml 文件末尾的文本：

      ```
      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. 如果使用证书身份验证，请通过编辑位于 docker-swarm 目录中的 docker-compose.local-overrides.yml 文件，将包含所有证书文件（HUB_POSTGRES_CA [服务器 CA 文件]、HUB_POSTGRES_CRT [客户端证书文件]、HUB_POSTGRES_KEY [客户端密钥文件]）的目录挂载到 webapp、jobrunner、身份验证和扫描服务中的 /run/secrets。请参见第 2c 步中的示例。

4. 为了能结合使用 SSL 与证书和/或用户名/密码身份验证，请将 HUB_POSTGRES_ENABLE_SSL_CERT 设置为“true”并完成第 2 步和第 3 步。

5. 安装或升级 Black Duck。

修改现有外部数据库的 PostgreSQL 用户名

默认情况下，PostgreSQL 数据库用户的用户名为 blackduck_user，PostgreSQL 管理员的用户名为 blackduck。

如果您使用的是外部 PostgreSQL 数据库，则可以更改这些用户名。

这些说明适用于现有 Black Duck 实例（其中，外部数据库当前使用 blackduck 和 blackduck_user 用户名）。要更改外部数据库的新配置的用户名，请按照上一节中的说明进行操作。

重要：对于不具有管理员权限的 Black Duck 数据库用户（这在 GCP 和 RDS 等托管提供商中是常见的），请连接到 bds_hub 数据库并运行 GRANT blackduck_user TO blackduck;

要修改现有 PostgreSQL 帐户名称：

1. 停止 Black Duck。
2. 重命名用户并重置 bds_hub 数据库中的密码。

```
alter user blackduck_user rename to NewName1 ;
alter user blackduck rename to NewName2 ;
alter user NewName1 password 'NewName1Password' ;
alter user NewName2 password 'NewName2Password' ;
```

3. 在位于 docker-swarm 目录中的 hub-postgres.env 文件内，编辑 HUB_POSTGRES_USER 和 HUB_POSTGRES_ADMIN 的值。HUB_POSTGRES_USER 的值是 blackduck_user 的新用户名。HUB_POSTGRES_ADMIN 的值是 blackduck 的新用户名。例如：

```
HUB_POSTGRES_USER=NewName1
HUB_POSTGRES_ADMIN=NewName2
```

4. 重新启动 Black Duck。

   注：在 2020.4 0 版本中，BDIO 数据库已被移除。

配置代理设置

编辑 blackduck-config.env 文件以配置代理设置。如果外部互联网访问需要代理，则需要配置这些设置。

以下是需要访问 Black Duck Software 托管的服务的容器：

- 身份验证
- 注册
- Jobrunner
- Web 应用程序
- 扫描
- Bomengine

代理环境变量包括：

- HUB_PROXY_HOST。代理服务器主机的名称。
- HUB_PROXY_PORT。代理服务器主机正在侦听的端口。
- HUB_PROXY_SCHEME。用于连接到代理服务器的协议。
- HUB_PROXY_USER。用于访问代理服务器的用户名。

NTLM 代理的环境变量包括：

- HUB_PROXY_WORKSTATION。发出身份验证请求的工作站。本质上，是此计算机的计算机名称。

- HUB_PROXY_DOMAIN。要在其中进行身份验证的域。

代理密码

如果通过代理利用身份验证，以下服务需要代理密码：

- 身份验证
- Bomengine
- Web 应用程序
- 注册
- Jobrunner
- 扫描

有三种方法可指定代理密码：

- 将包含名为 HUB_PROXY_PASSWORD_FILE 的文本文件的目录挂载到 /run/secrets。这是最安全的选项。
- 指定一个名为 HUB_PROXY_PASSWORD 的环境变量，其中包含代理密码。
- 使用 docker secret 命令创建名为 HUB_PROXY_PASSWORD_FILE 的密钥，如下所述：
  1. 使用 docker secret 命令，向 Docker Swarm 告知密钥。密钥的名称必须包括堆栈名称。在以下示例中，堆栈名称是“hub”：

     docker secret create hub_HUB_PROXY_PASSWORD_FILE <file containing password>
  2. 在位于 docker-swarm 目录中的 docker-compose.local-overrides.yml 文件内，对于每个服务（身份验证、webapp、注册、jobrunner、匹配引擎、BOM 引擎和扫描），提供密钥的访问权限。此示例适用于扫描服务：

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

     如有必要，请移除注释字符 (#)。
  3. 在 docker-compose.local-overrides.yml 文件末尾的 secrets 部分中，添加以下内容：

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

     如有必要，请移除注释字符 (#)。

如果未在单独挂载的文件或密钥中指定环境变量，则可以使用 blackduck-config.env 文件来指定该环境变量：

1. 移除 HUB_PROXY_PASSWORD 前面的井号 (#)，使其不再被注释排除。
2. 输入代理密码。
3. 保存文件。

注：如果在 Docker Swarm 部署中使用 docker secret HUB_PROXY_PASSWORD_FILE 提供代理密码时 KB 调用失败，请在 blackduck-config.env 文件中提供密码以解决问题。

导入代理证书

您可以导入要与代理配合使用的代理证书。

1. 使用代理证书文件创建一个名为 <stack name>_HUB_PROXY_CERT_FILE 的 Docker 密钥。例如

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

2. 在位于 docker-swarm 目录中的 docker-compose.local-overrides.yml 文件内，向以下服务提供密钥的访问权限：身份验证、webapp、注册、jobrunner 和扫描。此示例适用于扫描服务：

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

配置报告数据库密码

本节提供有关配置报告数据库密码的说明。

使用位于 docker-swarm/bin 目录中的 hub_reportdb_changepassword.sh 脚本，以设置或更改报告数据库密码。

注：在使用 Black Duck 自动安装的数据库容器时，此脚本可设置或更改报告数据库密码。如果使用外部 PostgreSQL 数据库，请使用首选的 PostgreSQL 管理工具来配置密码。

请注意，要运行脚本以设置或更改密码：

- 您可能需要是 Docker 组中的用户、root 用户或具有 sudo 访问权限。
- 您必须位于运行 PostgreSQL 数据库容器的 Docker 主机上。

在以下示例中，报告数据库密码设置为 “blackduck” ：

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

扩展 jobrunner、扫描、bomengine 和 binaryscanner 容器

可以扩展 jobrunner、扫描、bomengine 和 binaryscanner 容器。

您可能需要是 Docker 组中的用户、root 用户或具有运行以下命令的 sudo 访问权限。

扩展 bomengine 容器

此示例添加了第二个 bomengine 容器：

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

您可以通过指定比当前 bomengine 容器数更小的数字来移除 bomengine 容器。以下示例将 bomengine 容器恢复为单个容器：

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

注：Synopsys 建议您将 bomengine 容器扩展到与 jobrunner 容器相同的级别。

扩展 jobrunner 容器

此示例添加了第二个 jobrunner 容器：

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

您可以通过指定比当前 jobrunner 容器数更小的数字来移除 jobrunner 容器。以下示例将 jobrunner 容器恢复为单个容器：

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

扩展扫描容器

此示例添加了第二个扫描容器：

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

您可以通过指定比当前扫描容器数更小的数字来移除扫描容器。以下示例将扫描容器恢复为单个容器：

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

扩展 binaryscanner 容器

binaryscanner 容器与 Black Duck - 二进制分析 一起使用。

此示例添加了第二个 binaryscanner 容器：

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

注: 每个附加的 binaryscanner 容器都需要一个额外的 CPU、2 GB RAM 和 100 GB 可用磁盘空间。

您可以通过指定比当前 binaryscanner 容器数更小的数字来移除 binaryscanner 容器。以下示例将 binaryscanner 容器恢复为单个容器：

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

为单点登录配置 SAML

安全断言标记语言 (SAML) 是一种基于 XML 的开放标准数据格式，用于在各方之间交换身份验证和授权数据。例如，在身份提供程序和服务提供程序之间。Black Duck的 SAML 实施提供了单点登录 (SSO) 功能，使 Black Duck 用户能够在启用 SAML 时自动登录到 Black Duck。启用 SAML 适用于所有 Black Duck 用户，不能选择性地应用于单个用户。

所有托管客户都应该通过 SAML 或 LDAP，利用我们为单点登录 (SSO) 提供的现成支持来保护对 Black Duck 应用程序的访问。有关如何启用和配置这些安全功能的信息，请参见安装指南。此外，我们鼓励使用 SAML SSO 提供程序（提供双因素授权功能）的客户也启用和利用该技术来进一步保护对 Black Duck 应用程序的访问。

请注意以下事项：

- 不能同时配置 SAML 和 LDAP。
- 要启用或禁用 SAML 功能，您必须是具有系统管理员角色的用户。
- Black Duck 如果属性声明中提供了外部用户的信息，则能够同步和获取外部用户的信息（名称、名字、姓氏和电子邮件）。请注意，名字和姓氏值区分大小写。

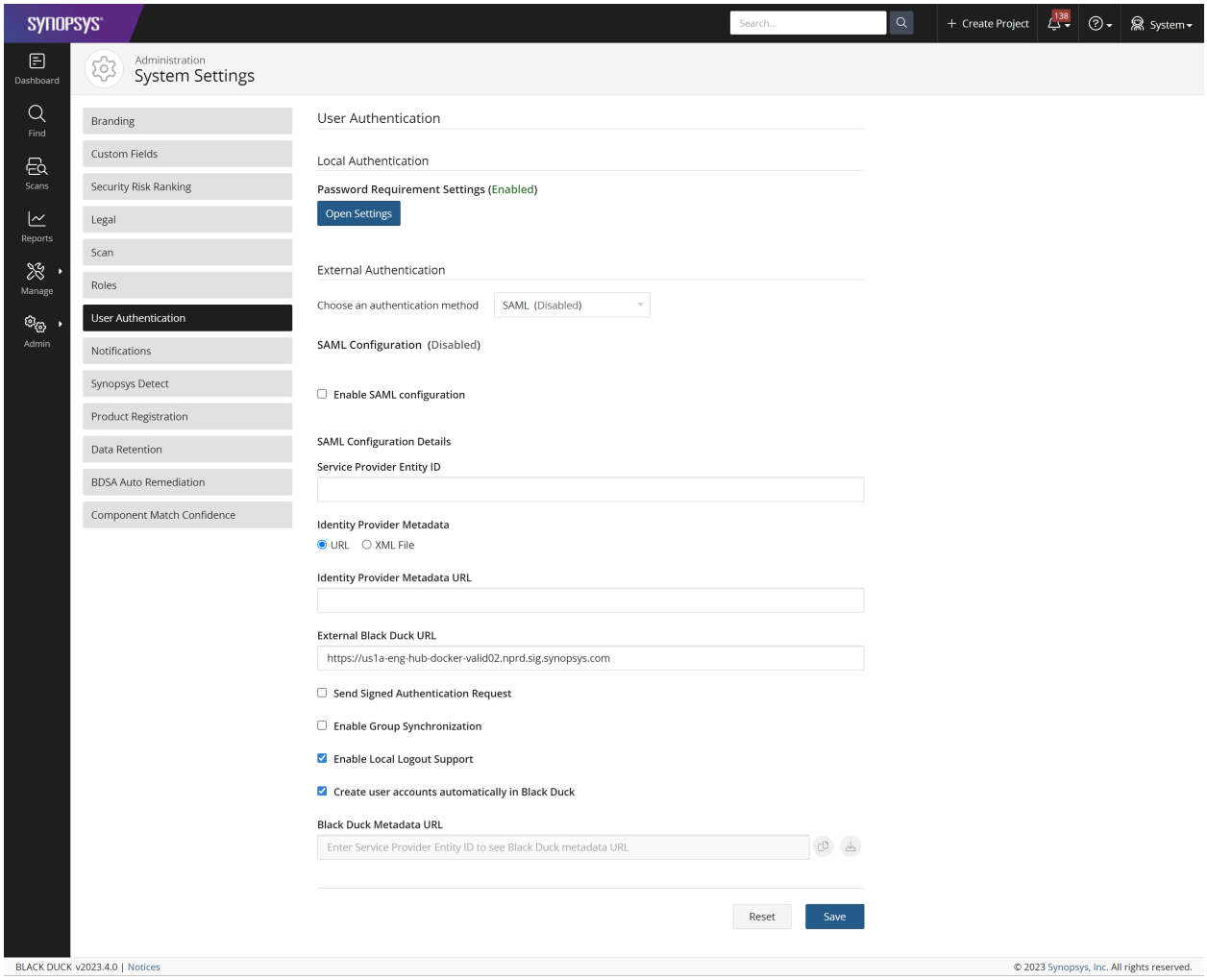
  Black Duck 如果在 Black Duck 中启用了组同步，也可以同步外部用户的组信息。
- 在启用 SAML 的情况下登录时，系统会将您重定向到身份提供程序的登录页面，而不是 Black Duck 的登录页面。
- 当 SSO 用户注销 Black Duck 时，将显示一个注销页面，通知他们已成功注销 Black Duck。此注销页面包括一个重新登录 Black Duck 的链接；用户可能不需要提供凭据即可成功重新登录 Black Duck。
- 如果 SSO 系统出现问题，并且需要禁用 SSO 配置，则可以输入 URLBlack Duck servername/sso/login 以登录 Black Duck。

使用 SAML 启用或禁用单点登录

1. 单击要编辑的注释旁边的

。

2. 选择用户身份验证以显示“用户身份验证”页面。



3. 在外部身份验证设置中，完成以下操作：
   a. 从选择身份验证方法下拉菜单中选择 SAML。
   b. 选中启用 SAML 配置复选框。
   c. 服务提供商实体 ID 字段：以 https://host 格式输入环境中的 Black Duck 服务器的信息，其中 host 是您的 Black Duck 服务器。
   d. 选择以下身份提供商元数据之一：
      - URL 并输入您的身份提供商的 URL。
      - XML 文件并拖放文件或单击显示的区域以打开一个对话框，您可以从中选择 XML 文件。
   e. 服务提供商实体 ID 字段。以 https://host 格式输入环境中的 Black Duck 服务器的信息，其中 host 是您的 Black Duck 服务器。
   f. 外部 Black Duck URL 字段。Black Duck 服务器的公共 URL 的 URL。

      例如：https://blackduck-docker01.dc1.lan
   g. （可选）选择以下任意项：
      - 发送签名的身份验证请求：如果启用此选项，则指示了声明方的首选项：依赖方应在发送前签署身份验证请求。
      - 启用组同步：如果启用此选项，则在登录时，将在 Black Duck 中创建来自身份提供程序 (IDP) 的组，并将用户分配给这些组。请注意，您必须配置 IDP，以便发送带有属性名称“Groups”的属性陈述中的组。
      - 启用本地注销支持：如果启用了此选项，则在注销 Black Duck 后，将显示 IDP 的登录页面。启用本地注销支持后，将发送 SAML 请求且 ForceAuthn="true"。与 IDP 核实以确认支持此功能。
      - 在 Black Duck 中自动创建用户帐户：如果用户使用 IdP 登录，而该用户在 Black Duck 中不存在，则我们会在 Black Duck 数据库中创建一个本地用户（如果选中了该选项）。
4. 单击保存。

单击保存后，将显示 Black Duck 元数据 URL 字段。您可以复制链接或直接下载 SAML XML 配置信息。

要使用 SAML 禁用单点登录

1. 单击

。
2. 选择用户身份验证以显示“用户身份验证”页面。
3. 在外部身份验证设置中，清除启用 SAML 配置复选框。
4. 单击保存。

其他信息

- 声明消费者服务 (ACS)：https://<host>/saml/SSO
- 建议的服务提供程序实体 ID：https://<host>，其中 host 是您的 Black Duck 服务器位置。

上传源文件

BOM 审核者需要能够通过确认匹配和调查漏报来轻松确认扫描结果。审查代码段匹配时， 查看源文件与匹配项的并排比较，这有助于评估和审查匹配。

Black Duck 提供上传源文件的功能，以便 BOM 审查者可以从 Black Duck UI 中查看文件内容。

当您在扫描过程中启用深度许可证数据检测或版权文本搜索时，上传源文件使 BOM 审核者能够从 Black Duck UI 中查看发现的许可证或版权文本。上传文件时，Black Duck 会提供嵌入式许可证和版权文本列表，并在文件中显示突出显示的文本。

要让 BOM 审核者在 Black Duck UI 中查看文件内容：

1. 管理员必须启用源文件的上传。
   a. 管理员使用环境变量启用该功能，并提供由称为密封密钥的“用户密钥”组成的、与 Docker 相关的密钥。
   b. Black Duck 检索密封密钥并生成用于加密和解密源文件的“主密钥”。

      “主密钥”通过“用户密钥”（AES-GCM-256，密钥长度为 32 字节）加密存储，并存储在装载的卷上。
2. 具有“全局代码扫描者”或“项目代码扫描者”角色的用户必须使用特征扫描程序并启用 --upload-source 参数。有关详细信息，请参阅在线帮助或用户指南。

   扫描客户端通过 SSL/TLS 安全端点并使用正确的授权令牌将源文件内容发送到 Black Duck 实例。

   “主密钥”对文件进行加密。上传的文件使用其关联的扫描标识符和文件签名存储，而不是按其文件名进行存储。

   在 Black Duck UI 中，源文件在网络上通过 HTTPS 传输。

请注意以下事项：

- 确保您有足够的磁盘空间用于文件上传。
- 您一次可以上传的最大总源大小为 4 GB (4000 MB)。此值可配置。
- 上传的文件将在 180 天后删除。此值可配置。
- 当上传服务达到最大磁盘设置的 95% 时，文件将被删除。

  该服务将删除最旧的文件，直到磁盘空间等于最大磁盘设置的 90%。

要启用文件上传：

1. 通过在主机系统上创建名为 SEAL_KEY 的文件来创建“用户密钥”。此文件应存储在安全位置。长度必须为 32 个字节或 32 个字符。例如：

   ```
   vi opt/secrets/SEAL_KEY
   ```
2. 要启用此功能，请将 docker-swarm 目录中的 blackduck-config.env 文件内的 ENABLE_SOURCE_UPLOADS 环境变量设置为 true。

   ```
   ENABLE_SOURCE_UPLOADS=true
   ```
3. 使用 Docker secret 命令，通过 SEAL_KEY 向 Docker Swarm 告知密钥。密钥的名称必须包括堆栈名称。在以下示例中，堆栈名称是“hub”：

   ```
   docker secret create hub_SEAL_KEY <location of key file>
   ```
4. 通过将注释字符 (#) 从 docker-swarm 目录中的 docker-compose.local-overrides.yml 文件内的 uploadcache 部分移除，向 upload-cache 服务提供密钥的访问权限：

   ```
   uploadcache:

   secrets:
    - SEAL_KEY
   ```
5. 从位于 docker-swarm 目录中的 docker-compose.local-overrides.yml 文件末尾的以下文本移除注释字符 (#)：

   ```
   secrets:
   ```

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

6. 重新启动容器。

您现在可以使用特征扫描程序上传源文件。

备份密封密钥和原始主密钥

备份密封密钥和原始主密钥。如果未备份这些密钥，则在密封密钥丢失时可能会丢失数据。

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

更换密封密钥

如果您丢失了密封密钥，则可以在备份现有主密钥的情况下将其更换，如上一节所述。

要更换密封密钥，请使用以下脚本：

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

其他配置选项

Black Duck 提供以下附加配置：

- 数据保留。默认情况下，文件数据保留 180 天。

  使用 DATA_RETENTION_IN_DAYS 环境变量更改此设置。

- 总源大小。默认情况下，最大总源大小为 4 GB (4000 MB)。

  使用 MAX_TOTAL_SOURCE_SIZE_MB 环境变量更改此设置。

要修改这些设置，请将环境变量添加到位于 docker-swarm 目录中的 docker-compose.local-overrides.yml 文件内的上传缓存服务，并定义新设置。例如：

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

您也可以限制上传文件的大小。默认情况下，最大大小为 5 MB。要修改此设置，请在命令行中输入：

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

其中 # 是以 MB 为单位的新值。例如，要将最大大小增加到 6 MB，请输入：

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

启动或停止 Black Duck

使用这些命令启动或关闭 Black Duck。

启动 Black Duck

如果尚未使用覆盖文件修改默认配置设置，请使用这些命令。

- 运行以下命令以启动带有 PostgreSQL 数据库容器的 Black Duck：

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

- 运行以下命令以启动带有 Black Duck - 二进制分析 的 Black Duck 并使用 PostgreSQL 数据库容器：

  ```
  docker swarm init
  ```

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

- 运行以下命令以启动带有外部数据库的 Black Duck：

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

- 运行以下命令以使用外部数据库启动带有 Black Duck - 二进制分析 的 Black Duck：

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

- 如果您运行的是具有 Swarm 服务的只读文件系统的 Black Duck，请将 docker-compose.readonly.yml 文件添加到前面的说明中。

  例如，要安装带有 PostgreSQL 数据库容器的 Black Duck，请输入以下命令：

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

使用覆盖文件时启动 Black Duck

如果已使用覆盖文件修改默认配置设置，请使用这些命令。

注: docker-compose.local-overrides.yml 文件必须是 docker-compose 命令中使用的最后一个 .yml 文件。

- 运行以下命令以启动带有 PostgreSQL 数据库容器的 Black Duck：

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

- 运行以下命令以启动带有 Black Duck - 二进制分析 的 Black Duck 并使用 PostgreSQL 数据库容器：

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

- 运行以下命令以启动带有外部数据库的 Black Duck：

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

- 运行以下命令以使用外部数据库启动带有 Black Duck - 二进制分析 的 Black Duck：

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

- 如果您运行的是具有 Swarm 服务的只读文件系统的 Black Duck，请将 docker-compose.readonly.yml 文件添加到前面的说明中。

  例如，要安装带有 PostgreSQL 数据库容器的 Black Duck，请输入以下命令：

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

关闭 Black Duck

- 运行以下命令以关闭 Black Duck：

```
docker stack rm hub
```

配置用户会话超时

配置用户会话超时值以自动从 Black Duck 服务器注销用户，并与您的公司安全策略保持一致。

1. 要查看当前超时值，请发出以下 GET 请求：

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

   注：用户必须具有 OAuth 客户端的读取权限才能使用 GET 方法。

2. 要更改当前超时值，请使用 PUT 请求正文发出以下 PUT 请求。
   PUT https://<Black-Duck-server>/api/system-oauth-client

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

   注：用户必须具有更新 OAuth 客户端的权限，才能对此任务使用 PUT 方法。系统管理员角色包括所需的权限。

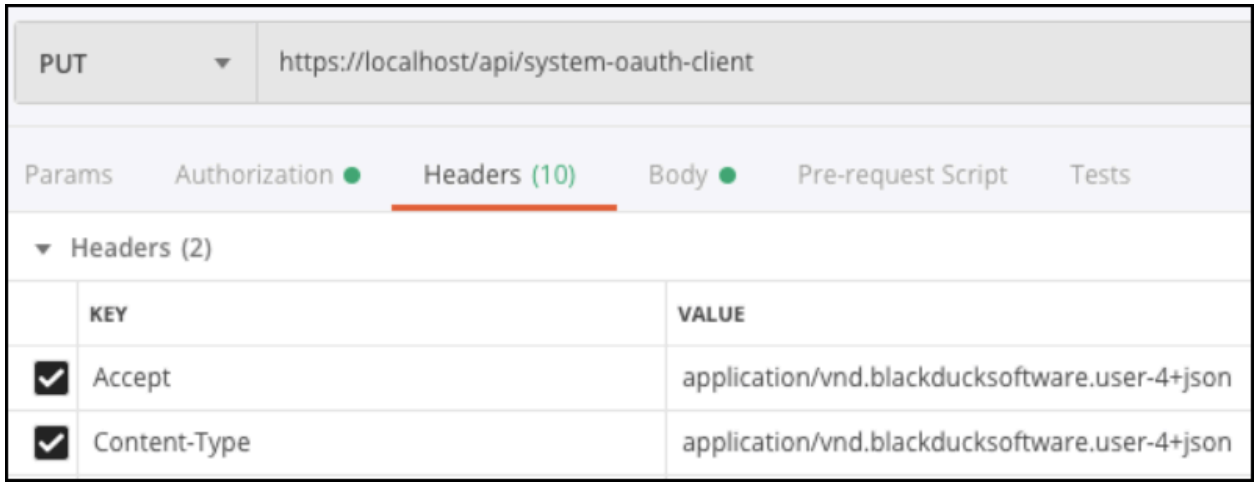
   您在 PUT 请求正文中键入的值是新的超时值。

   接受 30 分钟（1800 秒）和 24 小时 (86400) 之间的超时值。

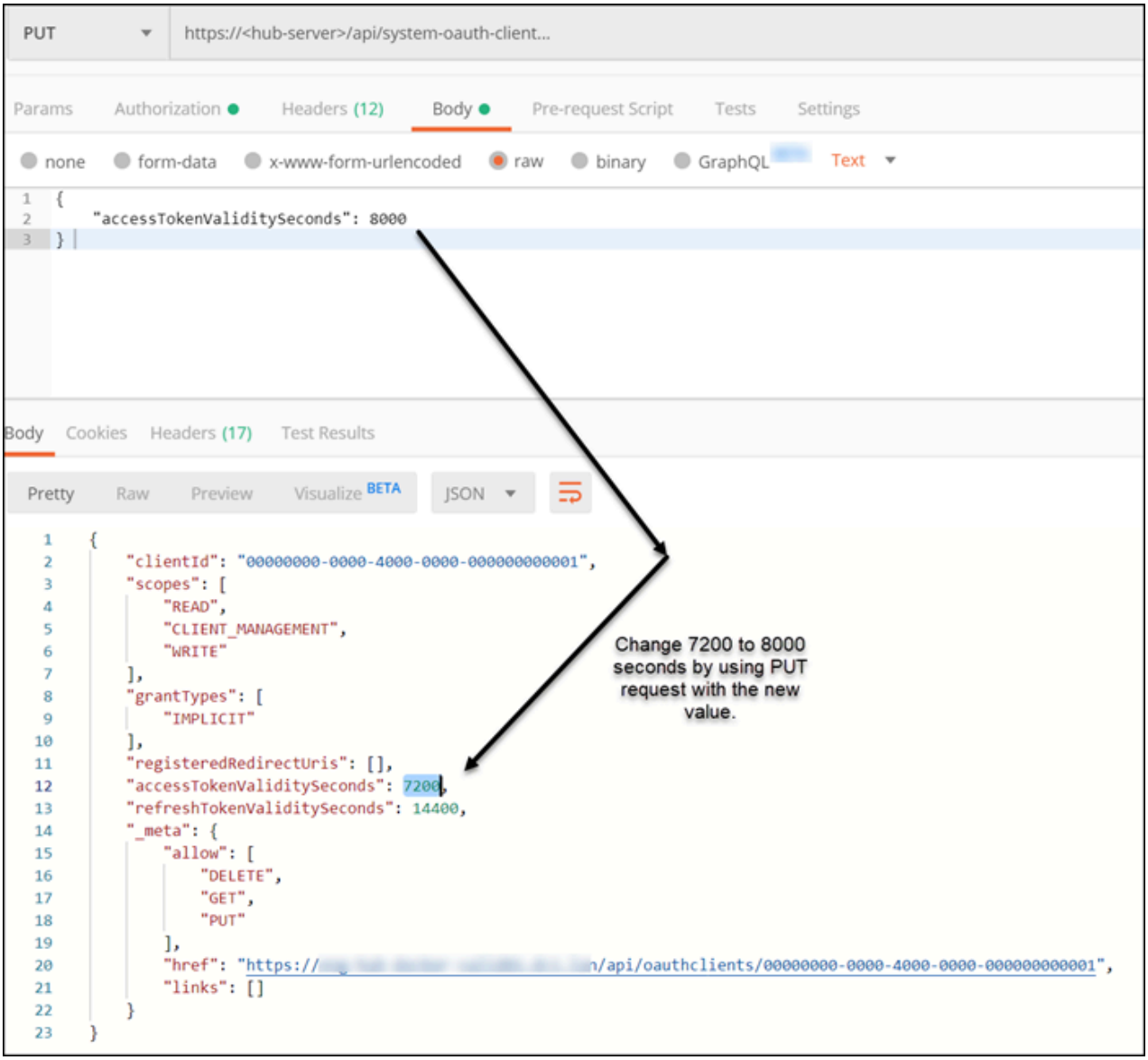
接受以下介质类型：

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

以下是 Postman 中的一个示例：



在以下示例中，您将超时值从 7200 更改为 8000 秒。



向客户支持提供您的 Black Duck 系统信息

客户支持可能会要求您向他们提供有关 Black Duck 安装的信息，例如系统统计信息以及环境或网络信息。为了便于您快速获取此信息，Black Duck 提供了一个脚本 system_check.sh， 您可以使用它来收集此信息。脚本将此信息输出到位于工作目录中的 system_check.txt 文件中，然后您可以将其发送给客户支持。

system_check.sh 脚本位于 docker-swarm/bin 目录中：

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

请注意，要运行此脚本，您可能需要是 Docker 组中的用户、root 用户或具有 sudo 访问权限。

了解默认 sysadmin 用户

安装 Black Duck 时，已配置了默认的系统管理员 (sysadmin) 帐户。默认 sysadmin 用户具有与其关联的所有角色和权限。

提示: 作为最佳做法，您应该在初始登录时使用默认 sysadmin 帐户，然后立即更改默认密码 blackduck，以确保服务器的安全。要更改密码，请从 Black Duck UI 右上角的用户名/用户配置文件图标中选择我的配置文件。

要编辑与 sysadmin 用户关联的默认电子邮件地址，请转到 Black Duck UI 上的用户管理页面，选择 sysadmin 用户名，更改电子邮件地址并保存。要查看更改，您必须注销并重新登录。

要访问用户管理页面，您使用的用户帐户必须具有超级用户角色，默认情况下，该角色分配给 sysadmin 帐户。电子邮件地址的主要用途是作为用户帐户的参考联系方式。

配置 Black Duck 报告延迟

在 Black Duck 2023.4.0 中，报告数据库作业流程每 480 分钟运行一次，这是可配置的。

要配置不同的报告延迟：

1. 编辑 docker-swarm 目录中的 blackduck-config.env 文件并配置 BLACKDUCK_REPORTING_DELAY_MINUTES=<value in minutes>

   例如， BLACKDUCK_REPORTING_DELAY_MINUTES=360

2. 重新启动容器。

配置容器的时区

默认情况下，Black Duck 容器的时区为 UTC。出于监控目的，您可能希望更改此值，以便日志中显示的时间戳反映本地时区。

要配置不同的时区：

1. 将 docker-swarm 目录中的 blackduck-config.env 文件内的 TZ 环境变量的值设置为新时区。使用 Wikipedia 中显示的值，如此处所示。

   例如，要将时区更改为在科罗拉多州丹佛使用的时区，请输入：

   ```
   TZ=America/Denver
   ```

2. 重新启动容器。

修改默认用法

用法指示此版本发布时打算如何将组件包括在项目中。

可能的用法值包括：

- 静态链接。紧密集成的组件，它与项目静态链接并与项目一起分发。
- 动态链接。与 DLL 或 .jar 等文件动态链接的适度集成组件。
- 源代码。源代码，比如 .java 或 .cpp 文件。
- 开发工具/已排除。组件将不包括在发布的项目中。例如，内部用于构建、开发或测试的组件。例如，单元测试、IDE 文件或编译器。

- 单独工作。适用于松散集成的组件。您的工作不是从组件派生的。要被视为单独的工作，您的应用程序需具有自己的可执行文件，并且组件和您的应用程序之间没有链接。一个示例是在您的分发介质中附带免费的 Acrobat PDF Viewer。
- 标准的实施。适用于您按照标准实施的情况。例如，项目附带的 Java 规范请求。
- 仅汇总。适用于您的项目不使用或以任何方式依赖的组件，尽管它们可能位于同一介质上。例如，您的发行版中包含的不相关产品的示例版本。
- 前提条件。适用于您的发行版所需、但未提供的组件。
- 未指定。此组件的用法尚未确定。

在 Black Duck 版本 2020.10.0 中，Synopsys 引入了 UNSPECIFIED 用法值，您可以将该值用作默认选项（而不是现有的默认值），以明确说明对组件是分配了真正的用法值还是默认用法值。例如，如果使用 DYNAMICALLY_LINKED 作为默认用法值，则可能无法区分具有真正的用法值 DYNAMICALLY_LINKED 的组件和默认分配了用法值的组件。通过使用 UNSPECIFIED 作为用法默认值，您知道，当看到分配 UNSPECIFIED 作为用法值时，应采取措施来分配有效的用法值。

默认用法由匹配类型决定：代码段的用法是“源代码”，而所有其他匹配类型都是“动态链接”。

Black Duck 使用以下变量，以便您可以更改类似匹配类型的默认用法：

- BLACKDUCK_HUB_FILE_USAGE_DEFAULT。定义此变量的用法将为以下匹配类型设置默认值：
  - 二进制
  - 精确目录
  - 精确文件
  - 文件已添加/删除
  - 文件已修改
  - 部分
- BLACKDUCK_HUB_DEPENDENCY_USAGE_DEFAULT。定义此变量的用法将为以下匹配类型设置默认值：
  - 文件依赖关系
  - 直接依赖关系
  - 过渡依赖关系
- BLACKDUCK_HUB_SOURCE_USAGE_DEFAULT。定义此变量的用法将为以下匹配类型设置默认值：
  - 代码段
- BLACKDUCK_HUB_MANUAL_USAGE_DEFAULT。定义此变量的用法将为以下匹配类型设置默认值：
  - 手动添加
  - 手动标识
- BLACKDUCK_HUB_SHOW_UNMATCHED：确定是否显示不匹配的组件计数。默认值为 false（不显示）。

bdio2 上传的 jsonld/bdio 文件的匹配类型

这些匹配的默认用法是“动态链接”。

- 直接依赖关系二进制
- 传递依赖关系二进制

这些是与现有二进制匹配相同的完整文件匹配。

单个文件可以有多个“传递依赖关系二进制”匹配，但只有一个“直接依赖关系二进制”匹配。

要配置不同的用法值：

1. 通过移除注释图标 (#) 并输入一个值，将 docker-swarm 目录中的 blackduck-config.env 文件编辑为新的用法值。使用文件中所示的其中一个用法值：SOURCE_CODE、STATICALLY_LINKED、DYNAMICALLY_LINKED、SEPARATE_WORK、IMPLEMENTATION_OF_STA

   例如，要将文件的默认用法更改为“未指定”：

   ```
   BLACKDUCK_HUB_FILE_USAGE_DEFAULT=UNSPECIFIED
   ```

   注：如果输入了错误的用法文本，则仍将应用原始默认值。一条警告消息将出现在 jobrunner 容器的日志文件中。

修改后的用法值将应用于任何新扫描或重新扫描。

自定义 Black Duck 容器的用户 ID

您可能需要更改运行容器时使用的用户 ID (UID)。

每个容器的当前 UID 为：

- 身份验证 (blackduck-authentication)：100
- Binaryscanner (bdba-worker)：0
- CA (blackduck-cfssl)：100
- DB (blackduck-postgres)：1001
- 文档 (blackduck-documentation)：8080
- Job Runner (blackduck-jobrunner)：100
- Logstash (blackduck-logstash)：100
- RabbitMQ (rabbitmq)：100
- 注册 (blackduck-registration)：8080
- 扫描 (blackduck-scan)：8080
- Web 应用程序 (blackduck-webapp)：8080
- Webserver (blackduck-nginx)：100
- Uploadcache (blackduck-uploadcache)：100
- Redis (blackduck-redis)：以任何用户/组的身份运行
- Bomengine (blackduck-bomengine)：100
- Matchengine (blackduck-matchengine：100

更改 UID 包括将容器的新值添加到 docker-swarm 目录中的 docker-compose.local-overrides.yml。在容器部分添加

user:UID_NewValue:root 行。

以下示例将 webapp 容器的 UID 更改为 1001：

webapp：

用户：1001:root

请注意以下事项：

- 无法更改 postgres 容器和 binaryscanner 的 UID。
  - postgres 容器的 UID 必须等于 1001。
  - binaryscanner 容器的 UID 必须等于 0 (root)。
- 虽然某些容器具有相同的 UID 值（例如，“文档”、“注册”、“扫描”和“Web 应用程序”容器都有 UID 8080），但更改一个容器的 UID 值并不会更改具有相同 UID 值的容器的 UID 值。例如，将“Web 应用程序”容器的值从 8080 更改为 1001 不会更改“文档”、“扫描”或“注册”容器的值 - 这些容器的 UID 值仍为 8080。
- 容器期望，无论以什么用户身份运行容器，都必须将该用户指定为 root 组中的用户。

要自定义 UID：

1. 下线 Black Duck。
2. 按以上所述编辑值。
3. 上线 Black Duck。

配置 Web 服务器设置

编辑 hub-webserver.env 文件以：

- 配置主机名。
- 配置主机端口。
- 禁用 IPv6。

配置主机名

编辑 hub-webserver.env 文件以配置主机名，使证书主机名匹配。环境变量将服务名称作为默认值。

当 Web 服务器启动时，如果没有配置证书，它将生成 HTTPS 证书。必须为 PUBLIC_HUB_WEBSERVER_HOST 环境变量指定一个值，以告知 Web 服务器它将侦听的主机名，以便主机名匹配。否则，证书将仅具有用作主机名的服务名称。此值应更改为公开主机名，用户将在浏览器中输入该主机名以访问 Black Duck。例如：

```
PUBLIC_HUB_WEBSERVER_HOST=blackduck-docker01.dc1.lan
```

配置主机端口

您可以为主机端口配置不同的值，默认值为 443。

要配置主机端口：

1. 将新的主机端口值添加到 docker-swarm 目录中的 docker-compose.local-overrides.yml 文件。

   使用 webserver 部分，通过下列格式添加端口信息： ports: ['NewValue:8443'] 例如，要将端口更改为 8443：

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

2. 编辑 hub-webserver.env 文件中的 PUBLIC_HUB_WEBSERVER_PORT 值以添加新的端口值。例如：

   ```
   PUBLIC_HUB_WEBSERVER_PORT=8443
   ```

3. 注释排除 docker-compose.yml 文件中的以下部分以禁止访问主机上的端口 443，否则，用户仍可以使用端口 443 访问主机。

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

禁用 IPv6

默认情况下，NGiNX 侦听 IPv4 和 IPv6。如果在主机上禁用了 IPv6，请将 IPV4_ONLY 环境变量的值更改为 1。

使用 UTF8 字符编码创建 BOM 报告

要在使用非西方字符时启用 BOM 报告中对 UTF8 字符编码的支持，请将以下内容添加到 blackduck-config.env 文件中：

```
USE_CSV_BOM=true
```

扫描监控

为扫描监控端点收集信息的默认时间段为 1 小时。可以使用以下属性将其配置为不同的值：

HUB_SCAN_MONITOR_ROLLUP_WINDOW_SECONDS

仅当系统在上述配置的上一时间段（默认时间段为 1 小时）内至少有特定数量的扫描要分析时，API 才能够提供信息。如果扫描小于阈值，则默认情况下 1 级请求将返回 OK 响应。可以使用以下属性将扫描数配置为不同的值：

HUB_SCAN_MONITOR_MINIMUM_SCAN_COUNT

1 级请求根据失败率阈值返回 “OK” 或 “NOT OK” 响应，其中，大于上限的结果将返回 “NOT OK” 响应。上限（默认值 30%）和下限（默认值 10%）设置为百分比，可使用以下属性进行配置：

HUB_SCAN_MONITOR_ERROR_RATE_LOWER_THRESHOLD_PERCENT

HUB_SCAN_MONITOR_ERROR_RATE_HIGHER_THRESHOLD_PERCENT

有关扫描监控 API 请求的更多信息，请参阅 Black Duck 中的《REST API 开发人员指南》。

配置 HTML 报告下载大小

您可以配置您的环境以允许更大或更小的 HTML 报告下载。默认大小为 3000 KB。超过此限制的报告将返回 “503 服务不可用” 错误消息。

要更改此限制，请更改 blackduck-config.env 文件中 HUB_MAX_HTML_REPORT_SIZE_KB 属性的值。

配置 KB 许可证更新和安全更新作业

要禁用 KB 许可证更新和安全更新作业，请在 blackduck-config.env 文件中添加以下属性：

KB_UPDATE_JOB_ENABLED=FALSE

要更改 KB 许可证更新作业的频率，请在 blackduck-config.env 文件中添加以下属性：

KB_LICENSE_UPDATER_PERIOD_MINUTES=<time in minutes>

在 Black Duck 中配置密钥加密

Black Duck 支持对系统中的关键数据进行静态加密。此加密基于编排环境（Docker Swarm 或 Kubernetes）调配给 Black Duck 安装的密钥。创建和管理此密钥、创建备份密钥以及根据您所在组织的安全策略轮换密钥的过程如下所述。

要加密的关键数据如下：

- SCM 集成 Oauth 令牌
- SCM 集成提供商 OAuth 应用程序客户端密钥
- LDAP 凭据
- SAML 私有签名证书

注：一旦启用了密钥加密，就永远不能禁用它。

什么是加密密钥？

加密密钥是一个随机序列，用于生成内部加密密钥以解锁系统内的资源。Black Duck 中的密钥加密由 3 个对称密钥（根密钥、备份密钥和以前的密钥）控制。这三个密钥通过传递到 Black Duck 的种子，作为 Kubernetes 和 Docker Swarm 密钥生成。这三个密钥被命名：

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

在正常情况下，并非全部三个种子都会被激活使用。除非正在执行轮换操作，否则唯一活动的种子将是根种子。

保护根种子

必须保护根种子。拥有您的根种子以及系统数据副本的用户可以解锁并读取系统的受保护内容。某些 Docker Swarm 或 Kubernetes 系统默认情况下不静态加密密钥。强烈建议将这些编排系统配置为在内部加密，以便以后在系统中创建的密钥能够保持安全。

根种子是从备份重新创建系统状态（作为灾难恢复计划的一部分）所必需的。根种子文件的副本应存储在独立于编排系统的秘密位置，以便种子与备份的组合可以重新创建系统。不建议将根种子存储在与备份文件相同的位置。如果一组文件被泄露或被盗 - 两种情况都会出现，因此，建议为备份数据和种子备份设置单独的位置。

在 Docker Swarm 中启用密钥加密

要在 Docker Swarm 中启用密钥加密：

1. 首先创建初始种子密钥
2. 编辑 blackduck-config.env 以将 SYNOPSYS_CRYPTO_ENABLED 设置为 true
3. 开始使用 docker-compose.encryption.yml 作为 Docker Swarm 部署的一部分

Docker Swarm 部署命令如下所示：

```
docker stack deploy -c docker-compose.yml [-c <other compose yaml files> …] -c docker-
compose.encryption.yml blackduck
```

密钥种子管理脚本

您可以在 Black Duck GitHub 公共存储库中找到示例管理脚本：

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

这些脚本不是用来管理 Black Duck 密钥加密，而是用来说明此处所述的低级 Docker 和 Kubernetes 命令的用法。有两组脚本，每组都在其自己的子目录中，对应于在 Kubernetes 和 Docker Swarm 平台上使用。对于 Kubernetes 和 Docker Swarm，各个脚本之间存在一对一对应关系（如果适用）。例如，两组脚本都包含一个具有如下名称的脚本：

createInitialSeeds.sh

生成种子

在 OpenSSL 中生成种子

可以使用任何机制（生成至少 1024 字节长度的安全随机内容）生成种子的内容。一旦创建种子并将其保存在密钥中，就应将其从文件系统中移除并保存在一个私密的安全位置。

OpenSSL 命令如下所示：

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

在 Docker Swarm 中生成种子

在 Docker Swarm 中，必须停止 Black Duck 才能创建和移除密钥。Docker Swarm 命令如下所示：

```
docker secret create crypto-root-seed ./root_seed
```

在 Docker Swarm 中，编排文件中配置的密钥必须存在，且不能为零长度。为了解决此限制，Black Duck 将 2 个或更少字节的“占位符”加密种子视为不存在。因此，可以使用以下命令在 Docker Swarm 中删除以前的密钥：

```
echo -n "1" | docker secret create crypto-prev-seed –
```

一旦通过编排文件启用加密，在启用 Black Duck 密钥加密之前，必须使用类似于此处所示的命令创建这三个种子密钥；请参阅示例脚本。

配置备份种子

建议备份根种子，以确保系统可以在灾难恢复场景中恢复。备份根种子是一个备用根种子，可用于恢复系统。因此，它必须以与根种子相同的方式安全地存储。

备份根种子具有一些特殊特性，即，一旦它与系统关联，即使在根种子轮换期间，它也仍然可行。一旦系统处理了备份种子，应将其从密钥中移除，以限制其受到攻击和泄漏的可能性。备份根种子可能有不同的（频率较低的）轮换计划，因为系统中的密钥不应在任何时候都处于“活动”状态。

当您需要或想要轮换根种子时，首先需要将当前根种子定义为上一个根种子。然后，您可以生成一个新的根种子并将其放置到位。

当系统处理这些种子时，以前的根密钥将用于轮换资源，以使用新的根种子。完成此处理后，应从密钥中移除之前的根种子，以完成轮换并清理旧资源。

创建备份根种子

初始创建后，备份种子/密钥将 TDEK（租户解密、加密密钥）低级密钥打包。示例脚本 createInitialSeeds.sh 将创建根种子和备份种子。一旦 Black Duck 运行，它使用两个密钥来打包 TDEK。

该操作完成并且根种子和备份种子都安全地存储在其他位置后，应删除备份种子密钥；请参阅示例脚本cleanupBackupSeed.sh。

如果根密钥丢失或泄漏，备份密钥可用于替换根密钥；请参阅示例脚本useRootSeed.sh。

轮换备份种子

与根密钥类似，备份种子应定期轮换。与根种子不同（旧的根种子存储为以前的种子密钥，而新的根种子密钥提供给系统），备份种子只是通过创建新的备份种子来进行轮换。请参阅示例脚本rotateBackupSeed.sh。

轮换完成后，新的备份种子应安全存储并从 Black Duck 主机文件系统中移除。

管理密钥轮换

根据组织的安全策略定期轮换正在使用的根种子是一种好做法。要执行此操作，还需要一个额外的密钥来执行轮换。要轮换根种子，将当前根种子配置为“上一个根种子”，并生成新生成的根种子并将其配置为根种子。一旦系统处理此配置（具体细节如下），密钥将被轮换。

此时，新旧种子都能够解锁系统内容。默认情况下，将使用新的根种子，允许您测试并确保系统按预期工作。一旦所有内容都得到验证，您就可以通过移除“以前的根种子”来完成轮换。

从系统中移除之前的根种子后，就不能再将其用于解锁系统内容，并且可以将其丢弃。新的根种子现在是正确的根种子，应适当地备份和保护。

根密钥用于打包实际加密和解密 Black Duck 密钥的低级 TDEK（租户解密、加密密钥）。应该在方便 Black Duck 管理员并符合用户组织规则时，定期轮换根密钥。

轮换根密钥的过程是使用当前根种子的内容创建以前的种子密钥。然后，应创建一个新的根种子并将其存储在根种子密钥中。

Docker Swarm 中的密钥轮换

对于 Docker Swarm，必须停止 Black Duck，执行三个种子操作，然后 Black Duck 再次启动。从运行中的 Black Duck 实例中提取的根种子作为上一个种子，extractRootAsPreviousSeed.sh。请参阅 Docker Swarm 示例脚本rotateRootSeed.sh。

轮换完成后，应移除上一个种子密钥；请参阅示例脚本cleanupPreviousSeed.sh。在 Docker Swarm 中，必须关闭系统，移除之前的密钥，然后再启动 Black Duck。

在用户界面中，转到“管理”>“系统信息”>“加密”，查看“加密诊断”选项卡即可跟踪轮换状态。

为 Blackduck 存储配置自定义卷

存储容器可配置为最多使用三 (3) 个卷来存储基于文件的对象。此外，可以将配置设置为将对象从一个卷迁移到另一个卷。

为什么使用多个卷？

默认情况下，存储容器使用单个卷来存储所有对象。此卷的大小取决于存储对象的典型客户使用情况。由于每个客户都不同，因此可能需要拥有比卷所能提供的空间更多的可用空间。由于并非所有卷都是可扩展的，因此可能需要添加不同的、更大的卷并将数据迁移到新卷。

可能需要多个卷的另一个原因是：卷托管在远程系统（NAS 或 SAN）上，并且该远程系统将被停用。需要创建托管在新系统上的第二个卷，并将内容移至该卷。

配置多个卷

Swarm 部署文件包含一个名为 docker-compose.storage-overrides.yml 的文件。此文件是一个模板，必须进行自定义并包含在部署中，才能覆盖默认存储配置，以使用更多卷。

此文件包含三个部分：环境、存储卷和服务卷。

环境

环境部分的结构如下：

```
services:
  storage:
    environment:
      # Provider 1 settings
      BLACKDUCK_STORAGE_PROVIDER_ENABLED_1: 'true'
      BLACKDUCK_STORAGE_PROVIDER_PREFERENCE_1: 10
      BLACKDUCK_STORAGE_PROVIDER_READONLY_1: 'false'
      BLACKDUCK_STORAGE_PROVIDER_MIGRATION_MODE_1: 'none'
      # Provider 2 settings
      BLACKDUCK_STORAGE_PROVIDER_ENABLED_2: 'false'
      BLACKDUCK_STORAGE_PROVIDER_PREFERENCE_2: 20
      BLACKDUCK_STORAGE_PROVIDER_READONLY_2: 'false'
      BLACKDUCK_STORAGE_PROVIDER_MIGRATION_MODE_2: 'none'
      # Provider 3 settings
      BLACKDUCK_STORAGE_PROVIDER_ENABLED_3: 'false'
      BLACKDUCK_STORAGE_PROVIDER_PREFERENCE_3: 30
      BLACKDUCK_STORAGE_PROVIDER_READONLY_3: 'false'
      BLACKDUCK_STORAGE_PROVIDER_MIGRATION_MODE_3: 'none'
...
```

这些设置启用或禁用三个存储提供商中的每一个。文件中的配置随附了与默认配置相同的配置。（提供商 1 已启用，2 和 3 已禁用）。

要配置提供商，请更改给定提供商的设置。设置如下：

设置	详细信息
BLACKDUCK_STORAGE_PROVIDER_ENABLED_(N)	默认值：用于提供商 1，true 用于其他 false 有效值：true 或 false 备注：这将启用或禁用提供商。必须至少启用一个提供商，否则系统无法启动。
BLACKDUCK_STORAGE_PROVIDER_PREFERENCE_(N)	默认值：提供商编号乘以 10 有效值： 0-999 备注：此值指示提供商之间的偏好程度。具有最高优先级（最低偏好程度编号）的提供商将用于存储新内容。配置中的其他提供商将继续提供它们的内容和所有其他功能 - 但不会向其添加新内容。 注意：所有活动提供商必须具有唯一的偏好程度编号。两个提供商不能共享相同的值。
BLACKDUCK_STORAGE_PROVIDER_READONLY_(N)	默认值： false 有效值：true 或 false 备注：这表示提供商为只读。最高优先级（最低偏好程度编号）的提供商不能为只读，否则系统无法正常工作。 只读提供商不会因添加数据或移除数据而更改存储卷，但数据库中的元数据将被处理，以记录对象删除和其他更改。
BLACKDUCK_STORAGE_PROVIDER_MIGRATION_MODE_(N)	默认值： none

设置	详细信息
	有效值：none、drain、delete、 duplicate 备注：这将配置提供商的迁移模式，本文档的迁移部分详细介绍了此模式是什么以及如何使用此模式。

存储卷

此部分将分配命名的存储卷（在系统上软件要求的位置放置特定挂载点）。提供商 1 的卷名称为 “storage-volume” ，并在其他位置进行定义。它将始终存在，即使未启用存储提供商 1，也是如此。覆盖文件的存储卷部分如下所示：

```
services:
  storage:
    environment:
      ...
    volumes:
      ## NOTE: File provider 1's volume mount point is always
      ## present since it is defined as the upstream default
      ## It will only be used if the file provider 1 is enabled
      ##
      ## File provider 2's volume mount point
      #- storage-volume2:/opt/blackduck/hub/uploads2
      ##
      ## File provider 3's volume mount point
      #- storage-volume3:/opt/blackduck/hub/uploads3
...
```

对于其他提供商，您只应为已启用的每个提供商取消这些部分的注释。例如，如果启用了提供商二 (2)，则文件的此部分应如下所示：

```
...
##
## File provider 2's volume mount point
- storage-volume2:/opt/blackduck/hub/uploads2
##
...
```

服务卷

此部分按名称及其配置定义卷。提供的配置使用 Docker Swarm 的默认卷驱动程序。与 “服务卷” 部分类似，您只需根据启用的提供商取消注释相应的部分。文件的服务卷部分如下所示：

```
services:
  storage:
    environment:
       ...
    volumes:
      ...
volumes: {
    ## NOTE: File provider 1's storage volume is always present
    ## since it is defined as the upstream default. It will
    ## only be used if the file provider 1 is enabled
    ##
    ## File provider 2's storage volume
    #storage-volume2: null,
    ##
    ## File provider 3's storage volume
    #storage-volume3: null,
}
```

注: 可以在此处进行默认存储驱动程序以外的配置。可以创建一个名为 “storage-volume2” 的存储卷，由 NFS、NAS 或 SAN 提供支持。这些配置需要根据标准 Docker swarm 参数以及适合客户环境的供应商设置或扩展来制定，因此不能在此处记录。

在卷之间迁移

配置多个卷后，可以将内容从一个或多个提供商卷迁移到新的提供商卷。这只能对不是最高优先级（最低偏好程度）的提供商执行。为此，请使用以下迁移模式之一配置卷。配置完成后，需要重新启动 Black Duck 才能启动迁移，迁移由后台作业执行，直至完成。

迁移模式	详细信息
none	目的：表示没有正在进行的迁移。 备注：默认迁移模式。
drain	目的：此模式将内容从配置的提供商移动到最高优先级（最低偏好程度编号）的提供商。内容移动后，将立即从源提供商中移除该内容。 备注：这是一个直接的移动操作 - 将其添加到目标提供商并从来源中移除。
delete	目的：此模式将内容从配置的提供商复制到最高优先级（最低偏好程度编号）的提供商。复制内容后，该内容将在源提供商中标记为删除。应用标准删除保留期 - 在该期限之后，内容将被移除。 备注：此移动允许系统在保留窗口期内从备份中恢复，以便源提供商中的内容仍然保持可行。默认保留期为 6 小时。
duplicate	目的：此模式将内容从配置的提供商复制到最高优先级（最低偏好程度编号）的提供商。复制内容后，来源（包括元数据）将保持不变。 备注：重复迁移后，数据库中将有两个卷，其中包含所有内容和元数据。如果您在“复制和转储”过程中执行下一步骤并取消配置原始卷，则文件将被删除，但元数据将保留在数据库中 - 引用未知卷，并在删减程序作业中生成警告（作业错误）。要解决此错误，请使用以下属性启用孤立元数据记录的删减： `storage.pruner.orphaned.data.pruning.enable=true`

配置 jobrunner 线程池

在 Black Duck 中，有两个作业池 - 一个运行计划作业（称为定期池），另一个运行从某些事件（包括 API 或用户交互）启动的作业（称为按需池）。

每个池都有两个设置：最大线程数和预取。

最大线程数是 jobrunner 容器可以同时运行的最大作业数。将定期和按需最大线程数相加，总和不应大于 32，因为大多数作业使用数据库，并且最多有 32 个连接。jobrunner 内存很容易饱和，因此默认的线程数设置得非常低。

预取是每个 jobrunner 容器在每次往返数据库的过程中将抓取的作业数。该值设置得越高，效率越高，但该值设置得越低，将使负载更均匀地分布在多个 jobrunner 中（但通常情况下，均匀的负载不是 jobrunner 的设计目标）。

在 Docker Swarm 中，将以下设置添加到您的 blackduck-config.env 文件中：

```
org.quartz.scheduler.periodic.maxthreads=2
org.quartz.scheduler.periodic.prefetch=1
org.quartz.scheduler.ondemand.maxthreads=4
org.quartz.scheduler.ondemand.prefetch=2
```

为 BOM 支持配置快速扫描

您可以配置快速扫描以提供完整的结果格式，以包括用于 BOM 支持的数据点。为此，请设置以下环境变量：BLACKDUCK_RAPID_SCAN_EXTENDED_DATA=true。

5. 卸载 Black Duck

按照以下说明卸载 Black Duck。

使用以下任一方法卸载 Black Duck：

- 停止并移除容器，然后移除卷。

  ```
  docker stack rm hub
  ```

- 停止并移除容器，但保留卷。例如：

  ```
  docker volume prune
  ```

  警告: 此命令可移除所有未使用的卷：将移除任何容器未引用的卷。这包括其他应用程序未使用的卷。

请注意，PostgreSQL 数据库不会备份。按照以下说明备份数据库。

6. 升级 Black Duck

Black Duck 支持升级到任何可用版本，使您能够在单个升级中跳过多个版本。

注: 对于从 2019.8.0 之前的版本升级的客户，在启动时运行两个作业（VulnerabilityReprioritizationJob 和 VulnerabilitySummaryFetchJob）以同步漏洞数据。

这些作业可能需要一些时间才能运行，并且在这些作业完成之前，现有 BOM 的整体漏洞评分将不可用。具有“系统管理员”角色的用户可以使用 Black Duck“作业”页面来监控这些作业。

注: 从 2018.12.0 之前的版本升级时，由于需要进行数据迁移以支持此版本中的新功能，因此升级时间将比通常的时间长。升级时间取决于 Black Duck 数据库的大小。如果您要监控升级过程，请与 Synopsys 客户支持联系以获取相关说明。

安装文件

GitHub 上提供了安装文件。

下载编排文件。作为安装/升级过程的一部分，这些编排文件将提取必要的 Docker 映像。

请注意，尽管 tar.gz 文件的文件名因访问文件的方式而异，但内容是相同的。

从 GitHub 页面下载

1. 选择从 GitHub 页面下载 .tar.gz 文件的链接：https://github.com/blackducksoftware/hub。
2. 解压缩 Black Duck .gz 文件：

   ```
   gunzip hub-2023.4.0.tar.gz
   ```
3. 解压缩 Black Duck.tar 文件：

   ```
   tar xvf hub-2023.4.0.tar
   ```

使用 wget 命令下载

1. 运行以下命令：

   ```
   wget https://github.com/blackducksoftware/hub/archive/v2023.4.0.tar.gz
   ```
2. 解压缩 Black Duck .gz 文件：

   ```
   gunzip v2023.4.0.tar.gz
   ```
3. 解压缩 Black Duck.tar 文件：

   ```
   tar xvf v2023.4.0.tar
   ```

用于清除审核事件表中未使用的行的迁移脚本

请注意，本节仅适用于从早于 2019.12.0 的 Black Duck 版本升级。

在升级过程中，将运行迁移脚本，以清除由于报告数据库更改而不再在 audit_event 表中使用的行。此脚本可能需要很长时间来运行，具体取决于 audit_event 表的大小。例如，针对 350 GB audit_event 表运行迁移脚本大约需要 20 分钟。

要确定审核事件表的大小，请执行以下任务之一：

- 从 bds_hub 数据库，运行以下命令：

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

- 以系统管理员身份登录到 Black Duck UI 并执行以下步骤：

  1. 单击“展开菜单”图标 ( ) 并选择管理。
  2. 在“管理”页面上，选择系统信息。

     出现“系统信息”页面。
  3. 在页面左列中选择 db。
  4. 在“表大小”表中查找 audit_event 表名称的 total_tbl_size 值。

```
Table Sizes (100 biggest sorted by size)
 schemaname | tablename                                                  | total_tbl_size_pretty | tbl_size_pretty | total_tbl_size | tbl_size
------------+------------------------------------------------------------+-----------------------+-----------------+----------------+------------
 st         | scan_composite_leaf                                        | 6508 MB               | 4232 MB         | 6823731200     | 4437254144
 st         | audit_event                                                | 2027 MB               | 1577 MB         | 2125168640     | 1653915648
```

升级完成后，强烈建议您在 audit_event 表上运行 VACUUM FULL 命令以优化 PostgreSQL 的性能。

- 根据系统的使用情况，运行 VACUUM FULL 命令可以回收大量 Black Duck 不再使用的磁盘空间。
- 通过运行此命令，将提高查询性能。

注: 如果不运行 VACUUM FULL 命令，可能会降低性能。

重要: 您必须确保有足够的空间来运行 VACUUM FULL 命令，否则，它将由于磁盘空间不足而失败，并可能损坏整个数据库。VACUUM FULL 命令所需的磁盘空间是 audit_event 表当前使用的磁盘空间的两倍。

要对容器化 PostgreSQL 数据库部署运行 VACUUM FULL 命令，请执行以下步骤：

1. 获取 audit_event 表的大小，并确保您有足够的空间运行 VACUUM FULL 命令。
2. 运行 docker ps 命令以获取 PostgreSQL 容器的 ID。
3. 运行以下命令以访问 PostgreSQL 容器。

   ```
   docker exec -it <container_ID> psql bds_hub
   ```
4. 运行以下 VACUUM FULL 命令以回收不再使用的空间。

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

如果您有外部 PostgreSQL 数据库部署，则必须确定 audit_event 表的大小，执行 VACUUM FULL 命令，然后在完成后重新启动部署。

从现有 Docker 架构升级

要从早期版本的 Black Duck 升级：

1. 迁移 PostgreSQL 数据库。
2. 升级 Black Duck。

迁移 PostgreSQL 数据库

Black Duck 2022.10.0 已将 PostgreSQL 映像从版本 11 迁移到版本 13，并支持从使用 PostgreSQL 9.6 容器（版本 4.2 至 2021.10.x）或 PostgreSQL 11 容器（版本 2022.2.0 至 2022.7.x）的版本进行升级。在安装过程中，blackduck-postgres-upgrader 容器将现有数据库迁移到 PostgreSQL 13，然后在完成后退出。

请注意以下事项：

- 强烈建议使用非核心 PG 扩展的客户在迁移前卸载这些扩展，并在迁移成功完成后重新安装；否则，迁移可能会失败。
- 进行复制设置的客户在迁移之前需要遵循 pg_upgrade 文档中的说明。如果没有进行上述的准备工作，迁移可能会成功，但复制设置将会中断。
- 未使用 Synopsys 提供的 PostgreSQL 映像的客户必须确保他们使用的是受支持的版本。Black Duck 2022.10.0 支持 PostgreSQL 版本 13.x 和 14.x，Synopsys 建议使用最新版本 14.x。
  - 社区 PostgreSQL 用户必须按照 PostgreSQL 14 的说明 (https://www.postgresql.org/docs/14/pgupgrade.html) 进行 PostgreSQL 升级。
  - 第三方 PostgreSQL（例如 RDS）的用户必须遵循其提供商的说明。
- 内部 PostgreSQL 容器的用户，PostgreSQL 将在需要时自动升级。
- 使用 4.2.0 之前的 Black Duck 版本的 Black Duck 用户应在升级之前联系 Synopsys 技术支持。

重要：开始迁移之前：

- 确保您有额外的 10% 磁盘空间，以避免由于系统目录的数据复制而导致磁盘使用情况出现意外问题。
- 检查根目录空间和卷安装以避免磁盘空间不足，因为这可能导致 Linux 系统中断。

迁移完全是自动进行的；除了标准的 Black Duck 升级之外，不需要额外的操作。blackduck-postgres-upgrader 容器必须以 root 身份运行才能更改上述布局和 UID。

随后 Black Duck 重新启动时，blackduck-postgres-upgrader 将确定不需要迁移，并立即退出。

升级 Black Duck

要升级 Black Duck：

1. 执行以下操作之一：
   - 如果未使用 docker-compose.local-overrides.yml 修改 .yml 文件，请使用较新版本的 Black Duck 中的 docker-swarm 目录内的文件运行以下命令之一：
     - ```
       docker stack deploy -c sizes-gen03/120sph.yaml -c docker-compose.yml hub
       ```

       （使用 DB 容器）
     - ```
       docker stack deploy -c sizes-gen03/120sph.yaml -c docker-compose.externaldb.yml hub
       ```

       （使用外部 PostgreSQL 实例）
   - 如果要升级 Black Duck - 二进制分析，请使用较新版本的 Black Duck 中的 docker-swarm 目录内的文件运行以下命令之一：
     - ```
       docker stack deploy -c sizes-gen03/120sph.yaml -c docker-compose.yml -c docker-compose.bdba.yml hub
       ```

       （使用带有 Black Duck - 二进制分析 的 DB 容器）
     - ```
       docker stack deploy -c sizes-gen03/120sph.yaml -c docker-compose.externaldb.yml -c docker-compose.bdba.yml hub
       ```

       （使用带有 Black Duck - 二进制分析 的外部数据库）
   - 如果使用了 docker-compose.local-overrides.yml 修改 .yml 文件，请运行以下命令之一。使用包含修改的 docker-compose.local-overrides.yml 文件版本；对于所有其他文件，请使用较新版本的 Black Duck 的 docker-swarm 目录中的文件：

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

  （使用 DB 容器）

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

  （使用外部 PostgreSQL 实例）

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

  （使用带有 Black Duck - 二进制分析 的 DB 容器）

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

  （使用带有 Black Duck - 二进制分析 的外部数据库）

- 要升级具有 Swarm 服务的只读文件系统的 Black Duck，请将 docker-compose.readonly.yml 文件添加到前面的示例中。

  例如，如果您使用了 docker-compose.local-overrides.yml 修改 .yml 文件，并希望升级使用 DB 容器的 Black Duck 安装，请运行以下命令：

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

注：可以更改上述命令中使用的资源定义文件，以更好地适应您的扫描模式和使用情况。有关所有可用选项的列表，请参阅 分发 一节。请注意每个选项的系统要求，因为它们随着资源定义的增加而增加。

注：如果您以前编辑过 hub-proxy.env 文件，则必须将这些编辑复制到 blackduck-config.env 文件。

备份和还原数据

如果数据库是内部的（即使用 postgres 容器作为后端数据库），您可以使用 docker-swarm/bin 文件夹下的脚本进行备份和恢复。

备份数据

1. 以数据库迁移模式启动 HUB：

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

2. 运行以下命令：

   ```
   ./hub_create_data_dump.sh /pathtodbdump
   ```

   脚本完成后，您将在 /pathtodbdump 下获得以下文件：

   - bds_hub.dump
   - globals.sql

恢复您的数据

1. 要还原数据，请以数据库迁移模式启动一个全新的 HUB，然后使用 hub_db_migrate.sh 还原数据：

```
docker stack deploy -c docker-compose.dbmigrate.yml hub
./hub_db_migrate.sh /pathtodbdump
```

2. 最后，正常启动 HUB：

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

   可以更改上面使用的资源定义文件，以更好地适应您的扫描模式和使用情况。有关所有可用选项的列表，请参阅 分发 一节。请注意每个选项的系统要求，因为它们随着资源定义的增加而增加。

如果数据库是外部的，请联系数据库管理员进行数据备份。

7. Docker 容器

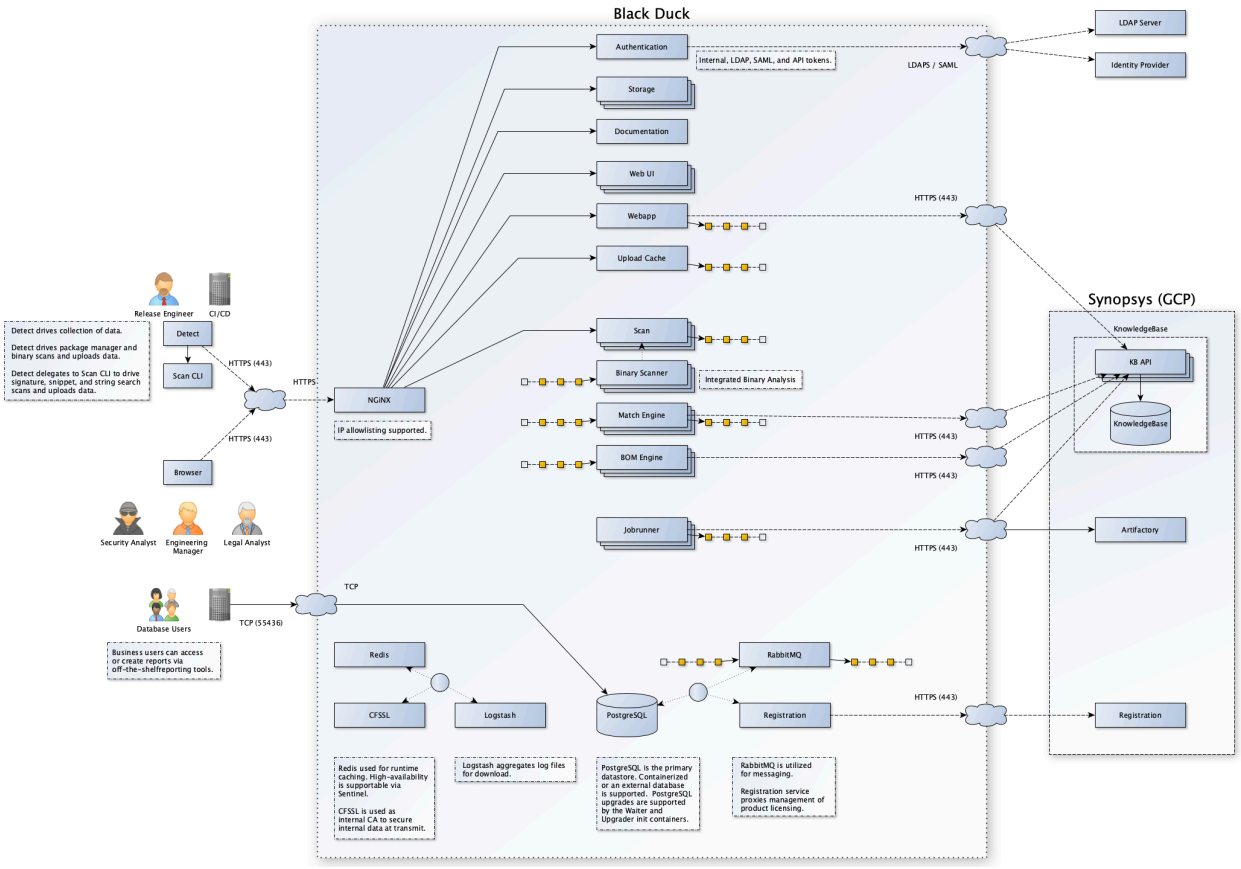
这些是 Docker 网络中组成 Black Duck 应用程序的容器：

1. 身份验证
2. Bomengine
3. CA
4. DB

   注: 如果您使用外部 Postgres 实例，则 Black Duck 应用程序中不包含此容器。
5. 文档
6. Jobrunner
7. Logstash
8. Matchengine
9. Rabbitmq
10. Redis
11. 注册
12. 扫描
13. 存储
14. Uploadcache - 此容器也用于 Black Duck - 二进制分析。
15. Webapp
16. Webserver
17. WebUI

如果启用 Black Duck - 二进制分析，则需要 Binaryscanner 容器。

下图显示了容器之间的基本关系以及哪些端口在 Docker 网络之外公开。



Zookeeper 容器在 Black Duck 版本 2020.4.0 中被移除。您可以手动移除以下 zookeeper 卷，因为它们不再被使用：

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

下表提供了有关每个容器的更多信息。

身份验证容器

容器名称：blackduck-authentication	
映像名称	blackducksoftware/blackduck-authentication：2023.4.0
说明	身份验证服务是所有与身份验证相关的请求所针对的容器。
可扩展性	此容器只能有一个实例。它当前无法扩展。
链接/端口	没有东西在外部（内部 8443）。此容器将需要连接到以下其他容器/服务： • postgres • cfssl • logstash • 注册 • webapp

容器名称：blackduck-authentication	
	该容器需要将 8443 公开给将链接到它的其他容器。
备用主机名环境变量	在其他类型的编排中运行时，有时为这些容器设置主机名（并非 Docker Swarm 使用的默认值）是非常有用的做法。可以设置这些环境变量以覆盖默认主机名： • postgres - $HUB_POSTGRES_HOST • cfssl - $HUB_CFSSL_HOST • logstash - $HUB_LOGSTASH_HOST • registration - $HUB_REGISTRATION_HOST • webapp - $HUB_WEBAPP_HOST
资源/限制	• 默认最大 Java 堆大小: 512 MB • 容器内存：1 GB • 容器 CPU：1 个 CPU
用户/组	此容器作为 UID 100 运行。如果容器作为 UID 0 (root) 启动，则在执行其主进程之前，用户将切换到 UID 100:root。 只要该容器也在 root 组 (GID/fsGroup 0) 中启动，它也可以作为随机 UID 启动。
环境文件	blackduck-config.env

Binaryscanner 容器

只有在您有 Black Duck - 二进制分析 的情况下，才会安装以下容器。

容器名称：bdba-worker	
映像名称	映像：sigsynopsys/bdba-worker:2023.03
说明	此容器可分析二进制文件。 此容器当前仅在启用了 “Black Duck - 二进制分析” 时使用。
可扩展性	可以扩展此容器。
链接/端口	此容器需要连接到以下容器/服务： • cfssl • logstash • rabbitmq • webserver 该容器需要将端口 5671 公开给将链接到它的其他容器。
备用主机名环境变量	在其他类型的编排中运行时，有时为这些容器设置主机名（并非 Docker Swarm 使用的默认值）是非常有用的做法。可以设置这些环境变量以覆盖默认主机名： • cfssl：$HUB_CFSSL_HOST • logstash：$HUB_LOGSTASH_HOST • rabbitmq：$RABBIT_MQ_HOST • webserver：$HUB_WEBSERVER_HOST
资源/限制	• 默认最大 Java 堆大小: 不适用

容器名称：bdba-worker	
	• 容器内存：2 GB • 容器 CPU：1 个 CPU
用户/组	此容器作为 UID 0 运行。
环境文件	hub-bdba.env

Bomengine 容器

容器名称：bomengine	
映像名称	blackducksoftware/blackduck-bomengine：2023.4.0
说明	Bomengine 容器负责构建 BOM 并使其保持最新。
可扩展性	可以扩展此容器
链接/端口	bomengine 容器需要连接到以下容器/服务： • postgres • 注册 • logstash • cfssl
备用主机名环境变量	有时，在其他类型的编排中运行时，任何单个服务名称可能会有所不同。例如，您可能有一个外部 PostgreSQL 端点，该端点通过不同的服务名称进行解析。要支持此类使用案例，可以设置这些环境变量以覆盖默认主机名： • postgres：$HUB_POSTGRES_HOST • 注册：$HUB_REGISTRATION_HOST • logstash：$HUB_LOGSTASH_HOST • cfssl：$HUB_CFSSL_HOST
资源/限制	• 默认最大 Java 堆大小: 4 GB • 容器内存：4.5 GB
用户/组	此容器作为 UID 100 运行 如果容器作为 UID 0 (root) 启动，则在执行其主进程之前，用户将切换到 UID 100:root。只要该容器也在 root 组 (GID/fsGroup 0) 中启动，它也可以作为随机 UID 启动。
环境文件	blackduck-config.env

CA 容器

容器名称：blackduck-cfssl	
映像名称	blackducksoftware/blackduck-cfssl:1.0.2
说明	此容器使用 CFSSL，用于为 PostgreSQL、NGiNX 和需要向 Postgres 进行身份验证的客户端生成证书。此容器还用于为组成应用程序的内部容器生成 TLS 证书。

容器名称：blackduck-cfssl	
可扩展性	此容器只能有一个实例。不应进行扩展。
链接/端口	该容器需要将 Docker 网络中的端口 8888 公开给链接到它的其他容器/服务。
资源/限制	• 默认最大 Java 堆大小: 不适用 • 容器内存：512 MB • 容器 CPU：未指定
用户/组	此容器作为 UID 100 运行。如果容器作为 UID 0 (root) 启动，则在执行其主进程之前，用户将切换到 UID 100:root。 只要该容器也在 root 组 (GID/fsGroup 0) 中启动，它也可以作为随机 UID 启动。
环境文件	blackduck-config.env

DB 容器

注: 如果您使用外部 Postgres 实例，则 Black Duck 应用程序中不包含此容器。

容器名称：blackduck-postgres	
映像名称	blackducksoftware/blackduck-postgres:9.6-1.1
说明	DB 容器包含 PostgreSQL 数据库，这是一个开源对象关系数据库系统。应用程序使用 PostgreSQL 数据库存储数据。 此容器有一个实例。这是存储应用程序所有数据的位置。Postgres 有两组端口。一个端口将公开给 Docker 网络中的容器。这是应用程序将使用的连接。此端口通过证书身份验证进行保护。第二个端口在 Docker 网络之外公开。这允许只读用户通过使用 hub_reportdb_changepassword.sh 脚本设置的密码进行连接。此端口和用户可用于报告和数据提取。 有关报告数据库的详细信息，请参阅报告数据库指南。
可扩展性	此容器只能有一个实例。不应进行扩展。
链接/端口	DB 容器需要连接到以下容器/服务： • logstash • cfssl 该容器需要将端口 5432 公开给 Docker 网络中将链接到它的其他容器。此容器在 Docker 网络外公开端口 55436 以进行数据库报告。
备用主机名环境变量	在其他类型的编排中运行时，有时为这些容器设置主机名（并非 Docker Swarm 使用的默认值）是非常有用的做法。可以设置这些环境变量以覆盖默认主机名： • logstash：$HUB_LOGSTASH_HOST • cfssl：$HUB_CFSSL_HOST
资源/限制	• 默认最大 Java 堆大小: 不适用 • 容器内存：3 GB • 容器 CPU：1 个 CPU

容器名称：blackduck-postgres	
用户/组	此容器作为 UID 1001 运行。如果容器作为 UID 0 (root) 启动，则在执行其主进程之前，用户将切换到 UID 1001:root。 此容器不能以任何其他用户 ID 启动。
环境文件	不适用

文档容器

容器名称：blackduck-documentation	
映像名称	blackducksoftware/blackduck-documentation：2023.4.0
说明	文档容器提供应用程序的文档。
可扩展性	此容器有一个实例。不应进行扩展。
链接/端口	此容器必须连接到以下其他容器/服务： • logstash • cfssl 文档容器必须将端口 8443 公开给链接到它的其他容器。
备用主机名环境变量	在其他类型的编排中运行时，有时为这些容器设置主机名（并非 Docker Swarm 使用的默认值）是非常有用的做法。可以设置这些环境变量以覆盖默认主机名： • logstash：$HUB_LOGSTASH_HOST • cfssl：$HUB_CFSSL_HOST
资源/限制	• 默认最大 Java 堆大小: 512 MB • 容器内存：512 MB • 容器 CPU：未指定
用户/组	此容器作为 UID 8080 运行。如果容器作为 UID 0 (root) 启动，则在执行其主进程之前，用户将切换到 UID 8080:root。 只要该容器也在 root 组 (GID/fsGroup 0) 中启动，它也可以作为随机 UID 启动。
环境文件	blackduck-config.env

Jobrunner 容器

容器名称：blackduck-Jobrunner	
映像名称	blackducksoftware/blackduck-jobrunner：2023.4.0
说明	Jobrunner 容器是负责运行应用程序的所有作业的容器。这包括匹配、BOM 构建、报告、数据更新等。此容器没有任何公开的端口。
可扩展性	可以扩展此容器。
链接/端口	Jobrunner 容器需要连接到以下容器/服务： • postgres

容器名称：blackduck-Jobrunner	
	• 注册 • logstash • cfssl
备用主机名环境变量	有时，在其他类型的编排中运行时，任何单个服务名称可能会有所不同。例如，您可能有一个外部 PostgreSQL 端点，该端点通过不同的服务名称进行解析。要支持此类使用案例，可以设置这些环境变量以覆盖默认主机名： • postgres：$HUB_POSTGRES_HOST • 注册：$HUB_REGISTRATION_HOST • logstash：$HUB_LOGSTASH_HOST • cfssl：$HUB_CFSSL_HOST
资源/限制	• 默认最大 Java 堆大小: 4 GB • 容器内存：4.5 GB • 容器 CPU：1 个 CPU
用户/组	此容器作为 UID 100 运行。如果容器作为 UID 0 (root) 启动，则在执行其主进程之前，用户将切换到 UID 100:root。 只要该容器也在 root 组 (GID/fsGroup 0) 中启动，它也可以作为随机 UID 启动。
环境文件	blackduck-config.env

Logstash 容器

容器名称：blackduck-logstash	
映像名称	blackducksoftware/blackduck-logstash:1.0.10
说明	Logstash 容器收集并存储所有容器的日志。
可扩展性	此容器只能有一个实例。不应进行扩展。
链接/端口	该容器需要将 Docker 网络中的端口 5044 公开给链接到它的其他容器/服务。
资源/限制	• 默认最大 Java 堆大小: 1 GB • 容器内存：1 GB • 容器 CPU：未指定
用户/组	此容器作为 UID 100 运行。如果容器作为 UID 0 (root) 启动，则在执行其主进程之前，用户将切换到 UID 100:root。 只要该容器也在 root 组 (GID/fsGroup 0) 中启动，它也可以作为随机 UID 启动。
环境文件	blackduck-config.env

Matchengine 容器

容器名称：matchengine	
映像名称	blackducksoftware/blackduck-matchengine：2023.4.0
说明	从云知识库中检索组件匹配信息。
可扩展性	此容器可以有多个实例。
链接/端口	• 不公开任何端口。 • 从外部连接到云 KB 服务 内部连接到以下组件： • cffsl • logstash • 注册 • postgres • rabbitmq
环境变量	环境变量如下： • HUB_CFSSL_PORTHUB_MATCHENGINE_HOST • HUB_MAX_MEMORY • HUB_REGISTRATION_HOST • HUB_REGISTRATION_PORTHUB_POSTGRES_HOST • HUB_POSTGRES_PORT • RABBIT_MQ_HOST • RABBIT_MQ_PORT
资源/限制	• 默认最大 Java 堆大小: 1 GB • 容器内存：1GB 预留，1.5GB 限制 • 容器 CPU：0.5 预留，1 限制
用户/组	此容器作为 UID 100 运行 如果容器作为 UID 0 (root) 启动，则在执行其主进程之前，用户将切换到 UID 100:root。只要该容器也在 root 组 (GID/fsGroup 0) 中启动，它也可以作为随机 UID 启动。
环境文件	blackduck-config.env

Rabbitmq 容器

容器名称：rabbitmq	
映像名称	blackducksoftware/rabbitmq:1.2.2
说明	此容器有助于将信息上传到二进制分析工作进程。它还使 bomengine 容器能够接收通知以开始 BOM 计算。它在 Docker 网络中公开端口，但不在 Docker 网络之外公开端口。
可扩展性	此容器只能有一个实例。不应进行扩展。
链接/端口	此容器需要连接到以下其他容器/服务：

容器名称：rabbitmq	
	• cfssl 该容器需要将端口 5671 公开给将链接到它的其他容器。
备用主机名环境变量	在其他类型的编排中运行时，有时为这些容器设置主机名（并非 Docker Swarm 使用的默认值）是非常有用的做法。可以设置这些环境变量以覆盖默认主机名： • cfssl：$HUB_CFSSL_HOST
限制	• 默认最大 Java 堆大小: 不适用 • 容器内存：1 GB • 容器 CPU：未指定
用户/组	此容器作为 UID 100 运行。如果容器作为 UID 0 (root) 启动，则在执行其主进程之前，用户将切换到 UID 100:root。 只要该容器也在 root 组 (GID/fsGroup 0) 中启动，它也可以作为随机 UID 启动。
环境文件	blackduck-config.env

Redis 容器

容器名称：blackduck-redis	
映像名称	redis:7.0.9-alpine3.17
说明	此容器将在 Black Duck 中实现更一致的缓存功能，并将用于提高应用程序性能。 默认情况下，Redis 作为主缓存机制启用。

配置	要配置 Redis，请使用以下设置：
	• blackduck-config.env 环境文件，用于配置 redis 设置
	  • Redis 设置：
	  • BLACKDUCK_REDIS_MODE：
	Redis 模式可以是独立或 sentinel
	  • BLACKDUCK_REDIS_TLS_ENABLED：
	是否在 Redis 客户端和服务器之间强制实施 TLS/SSL 连接。
	• 对于 Redis 独立模式，使用 docker-compose.yml，该模式需要 1024 MB 额外内存
	• 对于 redis sentinel 模式，使用 docker-compose.yml 和 docker-compose.redis.sentinel.yml，该模式可提供高可用性，但也需要 3168 MB 额外内存。
	• 就像其他服务一样，Redis 容器与 logstash 和 filebeat 集成。
	• Redis 容器具有运行状况检查。
	• 在 Black Duck Hub 系统信息页面中，有一个 redis-cache 选项卡，其中显示 Redis 调试信息。

可扩展性	该容器不应扩展。
链接/端口	Redis 容器需要连接到以下容器/服务： • logstash

容器名称：blackduck-redis	
	• cfssl
备用主机名环境变量	不适用

资源/限制	• 默认最大 Java 堆大小: 不适用
	• 容器内存：
	  • 独立 1024 MB
	  • Sentinel 模式 3168 MB
	• 容器 CPU：1

用户/组	可以作为任何用户/组运行，例如。{"runAsUser": 4567, "runAsGroup": 4567}
环境文件	blackduck-config.env

注册容器

容器名称：blackduck-registration	
映像名称	blackducksoftware/blackduck-registration：2023.4.0
说明	该容器是一种小型服务，用于处理来自其他容器的注册请求。此容器定期连接到 Black Duck 注册服务并获取注册更新。
可扩展性	该容器不应扩展。
链接/端口	注册容器需要连接到此容器/服务： • logstash • cfssl 该容器需要将端口 8443 公开给将链接到它的其他容器。
备用主机名环境变量	在其他类型的编排中运行时，有时为这些容器设置主机名（并非 Docker Swarm 使用的默认值）是非常有用的做法。可以设置这些环境变量以覆盖默认主机名： • logstash：$HUB_LOGSTASH_HOST • cfssl：$HUB_CFSSL_HOST
资源/限制	• 默认最大 Java 堆大小: 512 MB • 容器内存：640 MB • 容器 CPU：未指定
用户/组	此容器作为 UID 8080 运行。如果容器作为 UID 0 (root) 启动，则在执行其主进程之前，用户将切换到 UID 8080:root。只要该容器也在 root 组 (GID/fsGroup 0) 中启动，它也可以作为随机 UID 启动。
环境文件	blackduck-config.env

扫描容器

容器名称：blackduck-scan	
映像名称	blackducksoftware/blackduck-scan：2023.4.0
说明	扫描服务是所有扫描数据请求所针对的容器。
可扩展性	可以扩展此容器。
链接/端口	此容器需要连接到以下容器/服务： • postgres • 注册 • logstash • cfssl 该容器需要将端口 8443 公开给将链接到它的其他容器。
备用主机名环境变量	在其他类型的编排中运行时，有时为这些容器设置主机名（并非 Docker Swarm 使用的默认值）是非常有用的做法。可以设置这些环境变量以覆盖默认主机名： • postgres：$HUB_POSTGRES_HOST • 注册：$HUB_REGISTRATION_HOST • logstash：$HUB_LOGSTASH_HOST • cfssl：$HUB_CFSSL_HOST
资源/限制	• 默认最大 Java 堆大小: 2 GB • 容器内存：2.5 GB • 容器 CPU：1 个 CPU
用户/组	此容器作为 UID 8080 运行。如果容器作为 UID 0 (root) 启动，则在执行其主进程之前，用户将切换到 UID 8080:root。 只要该容器也在 root 组 (GID/fsGroup 0) 中启动，它也可以作为随机 UID 启动。
环境文件	blackduck-config.env

存储容器

容器名称：blackduck-storage	
映像名称	blackducksoftware/blackduck-storage：2023.4.0
说明	存储服务为许多用户提供了功能，可以在文件具有多个可用版本时上传文件，下载文件和定义默认文件。
可扩展性	可以扩展此容器。
链接/端口	此容器需要连接到以下容器/服务： • postgres • 注册 • rabbitmq • logstash

容器名称：blackduck-storage	
	• cfssl
备用主机名环境变量	有时，在其他类型的编排中运行时，任何单个服务名称可能会有所不同。例如，您可能有一个外部 PostgreSQL 端点，该端点通过不同的服务名称进行解析。要支持此类使用案例，可以设置这些环境变量以覆盖默认主机名： • postgres：$HUB_POSTGRES_HOST • 注册：$HUB_REGISTRATION_HOST • rabbitmq：$RABBIT_MQ_HOST • logstash：$HUB_LOGSTASH_HOST • cfssl：$HUB_CFSSL_HOST
资源/限制	• 默认最大 Java 堆大小: 512 MB • 容器内存：1 GB
用户/组	如果容器作为 UID 0 (root) 启动，则在执行其主进程之前，用户将切换到 UID 100:root。只要该容器也在 root 组 (GID/fsGroup 0) 中启动，它也可以作为随机 UID 启动。
环境文件	blackduck-config.env

Uploadcache 容器

容器名称：Uploadcache	
映像名称	blackducksoftware/blackduck-upload-cache:1.0.17
说明	此容器将用于临时存储上传内容以进行二进制分析，以及在上传源文件以进行代码段匹配时使用。
可扩展性	此容器只能有一个实例。不应进行扩展。
链接/端口	此容器需要连接到以下容器/服务： • cfssl • logstash • rabbitmq（如果 Black Duck - 二进制分析 已启用） 该容器将端口 9443 和 9444 公开给链接到它的其他容器。
备用主机名环境变量	在其他类型的编排中运行时，有时为这些容器设置主机名（并非 Docker Swarm 使用的默认值）是非常有用的做法。可以设置这些环境变量以覆盖默认主机名： • cfssl：$HUB_CFSSL_HOST • logstash：$HUB_LOGSTASH_HOST • rabbitmq：$RABBIT_MQ_HOST
限制	• 默认最大 Java 堆大小: 不适用 • 容器内存：512 MB • 容器 CPU：未指定

容器名称：Uploadcache	
用户/组	此容器作为 UID 100 运行。如果容器作为 UID 0 (root) 启动，则在执行其主进程之前，用户将切换到 UID 100:root。只要该容器也在 root 组 (GID/fsGroup 0) 中启动，它也可以作为随机 UID 启动。
环境文件	hub-bdba.env `blackduck-config.env`

Webapp 容器

容器名称：blackduck-webapp	
映像名称	blackducksoftware/blackduck-webapp：2023.4.0
说明	webapp 容器是所有 Web/UI/API 请求所针对的容器。它还处理任何 UI 请求。在图中，webapp 的端口不会在 Docker 网络之外公开。有一个 NGiNX 反向代理（如 WebServer 容器中所述）在 Docker 网络之外公开。
可扩展性	此容器只能有一个实例。不应进行扩展。
链接/端口	webapp 容器需要连接到以下容器/服务： • postgres • 注册 • logstash • cfssl 该容器需要将端口 8443 公开给将链接到它的其他容器。
备用主机名环境变量	在其他类型的编排中运行时，有时为这些容器设置主机名（并非 Docker Swarm 使用的默认值）是非常有用的做法。可以设置这些环境变量以覆盖默认主机名： • postgres：$HUB_POSTGRES_HOST • 注册：$HUB_REGISTRATION_HOST • logstash：$HUB_LOGSTASH_HOST • cfssl：$HUB_CFSSL_HOST
资源/限制	• 默认最大 Java 堆大小: 2 GB • 容器内存：2.5 GB • 容器 CPU：1 个 CPU
用户/组	此容器作为 UID 8080 运行。如果容器作为 UID 0 (root) 启动，则在执行其主进程之前，用户将切换到 UID 8080:root。 只要该容器也在 root 组 (GID/fsGroup 0) 中启动，它也可以作为随机 UID 启动。
环境文件	blackduck-config.env

Webserver 容器

容器名称：blackduck-webserver	
映像名称	blackducksoftware/blackduck-nginx:1.0.32
说明	WebServer 容器是应用程序容器的反向代理。它有一个端口在 Docker 网络之外公开。这是为 HTTPS 配置的容器。这里有配置卷，允许配置 HTTPS。 支持 HTTP/2 和 TLS 1.3
可扩展性	该容器不应扩展。
链接/端口	Web App 容器需要连接到以下容器/服务： • webapp • cfssl • 文档 • 扫描 • 身份验证 • 上传缓存（如果 Black Duck - 二进制分析 已启用） 此容器在 Docker 网络外公开端口 443。
备用主机名环境变量	在其他类型的编排中运行时，有时为这些容器设置主机名（并非 Docker Swarm 使用的默认值）是非常有用的做法。可以设置这些环境变量以覆盖默认主机名： • webapp：$HUB_WEBAPP_HOST • cfssl：$HUB_CFSSL_HOST • 扫描：$HUB_SCAN_HOST • 文档：$HUB_DOC_HOST • 身份验证：$HUB_AUTHENTICATION_HOST • 上传缓存：$HUB_UPLOAD_CACHE_HOST
资源/限制	• 默认最大 Java 堆大小: 不适用 • 容器内存：512 MB • 容器 CPU：未指定
用户/组	此容器作为 UID 100 运行。如果容器作为 UID 0 (root) 启动，则在执行其主进程之前，用户将切换到 UID 100:root。 只要该容器也在 root 组 (GID/fsGroup 0) 中启动，它也可以作为随机 UID 启动。
环境文件	hub-webserver.env， blackduck-config.env

WebUI 容器

容器名称：webui	
映像名称	映像名称：blackducksoftware/blackduck-webui:2023.4.0
说明	一个 Node.js 容器，用于管理用户界面的所有资源。Node.js 是在 Chrome 的 V8 JavaScript 引擎之上构建的开源应用程序服务器。
可扩展性	此容器只能有一个实例。

容器名称：webui	
链接/端口	此容器需要连接到以下容器/服务： • cfssl • logstash 此容器为来自 nginx 容器的代理请求公开端口 8443。Docker 网络外没有公开端口。
资源/限制	• 默认最大 Java 堆大小: 不适用 • 容器内存：640 MB • 容器 CPU：未指定